

eFax Developer™

Java User Guide for Outbound Processing

Version 2.0

Revision Date: 05/01/2014

Contents

WELCOME	5
CONTENTS AT A GLANCE	5
PRODUCT INTRODUCTION	6
eFAX DEVELOPER™ OUTBOUND OVERVIEW	6
COMPONENTS	6
<i>Software Developers Kit (SDK)</i>	6
<i>com.efaxdeveloper.util-2.0.x</i>	6
<i>commons-codec-1.6</i>	6
<i>OutboundRequest and OutboundResponse objects</i>	6
<i>DocumentBundler object</i>	6
<i>EmailBundler object</i>	7
<i>DispositionCatcher object</i>	7
<i>StatusRequest and StatusResponse objects</i>	7
<i>Security</i>	7
<i>eFax Developer™ Online Interface</i>	7
<i>Client Profile</i>	7
<i>XSD (XML Schema Definition)</i>	7
<i>Account Identifier</i>	7
PRODUCT INTEGRATION.....	8
INTEGRATION REQUIREMENTS	8
INTEGRATION PLANNING.....	8
PROGRAMMING.....	9
THE “OUTBOUND REQUEST”	9
THE “OUTBOUND RESPONSE”	9
THE “STATUS REQUEST”	10
THE “STATUS RESPONSE”	10
THE “FINAL DISPOSITION”	11
DYNAMIC FAX HEADERS.....	12
<i>Overview</i>	12
<i>@Variables</i>	12
<i>Font Control</i>	13
ARGUMENT MAPPING.....	14
<i>Outbound Request Mutator Methods (OutboundRequest)</i>	14
<i>Outbound Response Accessor Methods (OutboundResponse)</i>	20
<i>Status Request Mutator Methods (StatusRequest)</i>	21
<i>Status Response Accessor Methods (StatusResponse)</i>	22
<i>Final Disposition Accessor Methods (DispositionCatcher)</i>	25
HTML RESPONSE.....	26
OUTBOUND RESPONSE HTML	26
STATUS RESPONSE HTML	27
CODE SAMPLES.....	29
OUTBOUND REQUEST	29
STATUS REQUEST	30
FINAL DISPOSITION (WITH DISPOSITIONCATCHER)	31
USER LEVEL ERROR MESSAGES.....	32

ERROR MESSAGES RETURNED BY eFAX DEVELOPER™ AUTHENTICATION.....	32
ERROR MESSAGES RETURNED BY XSD VALIDATION.....	35

Notice

In all communications concerning this documentation, please refer to the revision date displayed on the front cover of this user's guide.

Copyright

The use, disclosure, reproduction, modification, transfer, or transmittal of this work for any purpose in any form or by any means without the written permission of j2 Global, Inc. is strictly prohibited.

Welcome

Welcome to *eFax Developer's Java User Guide for Outbound Processing*. This guide was designed to assist you in integrating *eFax Developer™* into your current outbound processing.

Users who wish to “cut to the chase” can skip directly to Product Integration as a quick-start.

Contents at a glance

- **Product Introduction** will provide an introduction to *eFax Developer™ Outbound*
- **Product Integration** will provide information on using *eFax Developer™ SDKs*
- **Programming** will provide essential information for the client-side programmer
- **HTML Response** will provide screen print examples of various HTML responses
- **Code Samples** will provide basic examples of how the objects should be used
- **User Level Error Messages** will cover common error messages and solutions

Product Introduction

eFax Developer™ Outbound Overview

eFax Developer™ is an online interface used in conjunction with *eFax Developer™* client-side APIs. The client-side process is divided into inbound and outbound processing. This documentation will cover client-side outbound processing in its entirety.

Based on XML technology, developers POST outbound requests to *eFax Developer™* through components found within our client-side Software Developers Kits (SDKs).

eFax Developer™ will authenticate the request, then immediately notify the client process whether or not the outbound request was successfully received via an HTTP response.

Once validated, a fax transmission will be created and launched. Clients can be notified of a transmission's final disposition. Clients can control this notification through elements passed in the outbound request. A client can suppress notification, or generate notification upon success, failure, or both.

Components

Software Developers Kit (SDK)

The *eFax Developer™* SDK contains everything the client programmer will need to create a seamless easy to use interface to *eFax Developer™*.

Components found within the SDK are separated into inbound and outbound functionality. This document will focus on outbound processing only.

com.efaxdeveloper.util-2.0.x

The *com.efaxdeveloper.util-2.0.x.jar* file is the client-side API library. This Java™ Archive (JAR) file contains the core classes required to interface with *eFax Developer™*.

commons-codec-1.6

The *commons-codec-1.6.jar* file contains the Apache Commons Codec, a required component to successfully interface with *eFax Developer™*.

OutboundRequest and OutboundResponse objects

The *OutboundRequest* and *OutboundResponse* objects are used by the client-side process to submit fax transmission to and manage responses from *eFax Developer™*.

DocumentBundler object

The *DocumentBundler* is used by the client-side process to bundle all documents for transmission. Once loaded, the *DocumentBundler* is passed to the *OutboundRequest* instance via the *setDocuments()* method.

EmailBundler object

Similar in concept to the *DocumentBundler*, the *EmailBundler* is used by the client-side process to bundle email addresses for final disposition notification processing. Once loaded, the *EmailBundler* is passed to the *OutboundRequest* instance via the `setDispositionsTo(EmailBundler)` method.

DispositionCatcher object

If desired, *eFax Developer™* can generate final disposition notifications to a client specified URL. The *DispositionCatcher* is the interface between the client's disposition endpoint and the *eFax Developer™* outbound dispositions.

StatusRequest and StatusResponse objects

The *StatusRequest* and *StatusResponse* objects are used by the client-side process to submit status requests to and manage responses from *eFax Developer™*.

Security

eFax Developer™ is designed with data security in mind. Requests are made using secure (HTTPS) protocol ensuring sender and receiver are the only parties able to decode the request. Additionally, each request will be authenticated against the requesting client's profile and XSD (schemata).

eFax Developer™ Online Interface

Clients can change various outbound settings through *eFax Developer™* online by clicking the "Settings" icon located on any page of the interface. Go to <https://secure.efaxdeveloper.com> and log into the *eFax Developer™* online interface.

Client Profile

eFax Developer™ client profile will consist of a user name, password and a unique Account Identifier. If desired, clients may choose to have all submitting IP Address' validated as an additional layer of security. Clients can access their profile via the *eFax Developer™* online interface.

XSD (XML Schema Definition)

To ensure data integrity, *eFax Developer™* will validate all requests against the client's XSD file. XSD files are client specific and generated at account setup time.

Account Identifier

Each client will be provided a unique Account Identifier. The Account Identifier is passed as part of the request. *eFax Developer™* will authenticate the Account Identifier against the requesting client's profile.

Product Integration

Integration Requirements

Applications that use the *com.efaxdeveloper.util-2.0.x* API must meet the following requirements.

- JDK 1.5 or greater is required
- Apache Commons Codec 1.6 or later (*commons-codec-1.6.jar* is included with our SDK package)

Integration Planning

To successfully integrate with *eFax Developer™*, it is important to have the appropriate tools as well as a project plan prior to attempting integration. The following is provided to assist you in the integration planning process.

- Contact *eFax Developer™* Support to set up a client profile and receive a copy of the SDK
- Log into the *eFax Developer™* online interface to set your outbound settings as desired
- Unzip the SDK into a folder of your choosing
- Add the *com.efaxdeveloper.util-2.0.x.jar* and *commons-codec-1.6.jar* to your CLASSPATH
- Carefully review this documentation, sample code and Javadoc found within the SDK folder
- Develop your outbound request processing
- Develop your “final disposition” processing (separate application flow if used)
- Test your application’s integration with *eFax Developer™*
- Make your application live

Programming

The “Outbound Request”

The *OutboundRequest* object is responsible for submitting fax requests to eFax Developer™ Outbound Services. Clients can specify a number of varying parameters on a request-by-request basis. Clients can specify final disposition notification be sent via email, or be posted to a client defined endpoint as desired.

It is recommended that clients review the *OutboundRequestServlet* sample code contained within the Samples folder for a detailed example on setting up and submitting an “Outbound Request.”

The following is provided as a high-level overview of expected client-side “Outbound Request” processing.

For each outbound request:

- Instantiate an *OutboundRequest* instance
- Instantiate a *DocumentBundler* instance
- Add all documents to be faxed to the *DocumentBundler*
- Set the *DocumentBundler* to the *OutboundRequest* instance
- Set the request arguments via the *OutboundRequest* object’s mutator methods
- Submit the outbound request using the object’s *sendFax()* method
- Process the *OutboundResponse* object returned by the *sendFax()* method

The “Outbound Response”

The *OutboundResponse* object is used to retrieve data elements returned in the HTTP response to the *OutboundRequest.sendFax()* method call. The client process can easily retrieve data elements via the *OutboundResponse* object’s accessor methods.

It is recommended that clients review the *OutboundRequestServlet* sample code contained within the Samples folder for a detailed example on processing the HTTP response returned by eFax Developer™.

The following is provided as a high-level overview of expected client-side “Outbound Response” processing.

For each outbound request:

- Process the *OutboundResponse* object returned by *OutboundRequest.sendFax()*
- If an HTML response was requested, retrieve the HTML from the *getRawResponse()* method otherwise retrieve the data elements via the *OutboundResponse* object’s accessor methods.
- If the *OutboundRequest* was rejected, where *OutboundResponse.isApproved()* returns false, it is the client’s responsibility to “fix” user level errors before resubmitting fax requests.

The “Status Request”

The *StatusRequest* object is responsible for submitting status requests to eFax Developer™ Outbound Services. Clients can provide their client-defined transmission identifier or the eFax Developer™ DOC identifier to request the current status of a transmission.

It is recommended that clients review the *StatusRequestServlet* sample code contained within the Samples folder for a detailed example on setting up and submitting a “Status Request.”

The following is provided as a high-level overview of expected client-side “Status Request” processing.

- Instantiate a *StatusRequest* instance
- Set the request arguments as required via the objects mutator methods
- Submit the status request using the object’s *getStatus()* method
- Process the *StatusResponse* object returned by the *getStatus()* method

The “Status Response”

The *StatusResponse* object is used to retrieve data elements returned in the HTTP response to the *StatusRequest.getStatus()* method call. The client process can easily retrieve data elements via the *StatusResponse* object’s accessor methods.

It is recommended that clients review the *StatusRequestServlet* sample code contained within the Samples folder for a detailed example on processing the HTTP response returned by eFax Developer™.

The following is provided as a high-level overview of expected client-side “Status Response” processing.

For each status request:

- Process the *StatusResponse* object returned by *StatusRequest.getStatus()*
- If an HTML response was requested, retrieve the HTML from the *getRawResponse()* method otherwise retrieve the data elements via the *StatusResponse* object’s accessor methods.
- If the *StatusRequest* was rejected, where *StatusResponse.isApproved()* returns false, it is the client’s responsibility to “fix” user level errors before resubmitting status requests.

The “Final Disposition”

If desired, *eFax Developer™* can generate “final disposition” notifications via HTTP POST to a designated URL as specified via the `setDispositionsTo()` method of the initial outbound request. *eFax Developer™* will make an initial attempt to deliver this disposition. If delivery is not successful, *eFax Developer™* will retry the disposition until a total of 4 attempts have been made. With each attempt, an alert email will be sent to the client’s primary email address until a confirmation (“Post Successful”) is received back from the client-side “disposition response” process, or 4 total attempts have been tried.

If desired, *eFax Developer™* can generate “final disposition” notifications to one-or-many email addresses as indicated in the initial outbound request. If the email option is selected, the HTTP POST outbound disposition will not be generated.

It is recommended that clients review the *DispositionCatcherServlet* sample code contained within the Samples folder for a detailed example on setting up “Final Disposition” processing.

The following is provided as a high-level overview of expected client-side “Final Disposition” processing.

- Retrieve the “xml” argument from the “final disposition” HTTP POST.
- Instantiate an instance of the *DispositionCatcher* passing to it the XML string just retrieved.
- Retrieve the disposition values via the *DispositionCatcher* accessor methods.
- If desired, validate the final disposition’s username and password for security.
- Return an HTML response of “Post Successful” back to *eFax Developer™*.

Dynamic Fax Headers

Overview

eFax Developer™ allows clients the ability to *override* the application generated fax header. This section is provided as a programmer's reference to this optional feature.

The following line is an example of a dynamic fax header line with a static company name inserted:

"@DATE1 @TIME3 **My Company Name** @ROUTETO{26} @RCVRFAX **Pg**%P/@SPAGES"

That dynamic header line once converted will display as follows:

10/15/03 11:11AM **My Company Name** Recipient Name Company 8581234567 **Pg** 1/2

Dynamic Fax Headers are freeform; the client is free to format the line as desired. The maximum length of this line is 80 characters for a single line. *@Variables* can be inserted in no particular order or removed completely. Static text can be added as desired, so "Pg 1/2" could have just as easily been made to display as "Page 1 of 2" or just "Page 1." *In the above example, bolded text represents static text.*

@Variables

The following describes variables available for use in the dynamic fax header string:

- @DATEx - from our server system date (Pacific Time)
 - @DATE0 yyyyymmdd (example: 20031015)
 - @DATE1 mm/dd/yy (example: 10/15/03)
 - @DATE2 dd/mm/yy (example: 15/10/03)
 - @DATE3 dd/xx/yy (example: 15/OC/03)
 - @DATE4 mm/dd/yyyy (example: 10/15/2003)
 - @DATE5 dd mon yyyy (example: 15 Oct 2003)
 - @DATE6 xxxxx dd, yyyy (example: October 15, 2003)
 - @DATE7 yy mm dd (example: 03 10 15)
 - @DATE8 yy-mm-dd (example: 03-10-15)
 - @DATE9 yymmdd (example: 031015)
- @TIMEx – from our server system time (Pacific Time)
 - @TIME1 hh:mm (example: 17:30)
 - @TIME2 hh:mm:ss (example: 17:30:00)
 - @TIME3 hh:mmxx (example: 05:30PM)
 - @TIME4 hhmm (example: 1730)
- @ROUTETO{n} – “Recipient Name Recipient Company” passed within the XML-formatted argument. *The number in brackets indicates the maximum width (characters) of this area within the header.*
- @RCVRFAX – “Recipient Fax” number passed within the XML-formatted argument.
- %P – Contains the current page number.
- @SPAGES – Contains the number of pages; including the cover sheet; to be sent.

Font Control

It is possible to control the font size within Dynamic Fax Headers through the use of one or many font control variables. The following describes this feature.

- `%nf` – “n” contains a number (0 through 3) that indicates the font size. “0” is the default value with each subsequent number indicating a slightly smaller font. Font control variables can be placed anywhere within the fax header line. Any text following a font control variable will be affected until another font control variable is encountered.

Argument Mapping

Outbound Request Mutator Methods (*OutboundRequest*)

Method	Required/Optional	Type	Length	Description/Values
setAccountID	Required	Alphanumeric	10	Method used to set the client's account identifier. This value is provided at setup time and is required for authentication.
setConnectionTimeout	Optional	Numeric		Method used to control the connection timeout for the request in milliseconds. A zero value is interpreted as an infinite timeout.
setCustomerID	Optional	Alphanumeric	50	Method used to set a client specified customer identifier. The CustomerID is optional. It is not required or validated by <i>eFax Developer™</i> .
setDispositionsTo	Optional	EmailBundler	N/A	Method used to set the disposition emails. The EmailBundler object contains all email addresses chosen to receive final disposition notifications.
setDispositionsTo	Optional	Alphanumeric	100	Method used to set the disposition URL address. The URL pointing to the client endpoint that will receive final disposition POST requests from <i>eFax Developer™</i> .

Method	Required/Optional	Type	Length	Description/Values
setDispositionLanguage	Optional	Alphanumeric	2	<p>Method used to set the desired disposition language.</p> <p>Set one of the following ISO 639-1 language codes to control final disposition notification emails:</p> <p>“en” English (default) “de” German “es” Spanish “fr” French “it” Italian “nl” Dutch “pl” Polish “pt” Portuguese</p>
setDispositionLevel	Optional	Alphanumeric	7	<p>Method used to set the disposition level.</p> <p>“ERROR” “SUCCESS” “BOTH” “NONE” (default)</p> <p>Defines the level at which final disposition notifications are received. Clients can suppress the notification, or generate the notification upon success, failure or both.</p>
setDocuments	Required	DocumentBundler	N/A	<p>Method used to set the documents to be faxed.</p> <p>The DocumentBundler contains all documents chosen for this transmission.</p> <p><i>eFax Developer™</i> accepts the following file extensions:</p> <p>doc docx xls xlsx ppt pptx html/htm tif/tiff jpg/jpeg txt pdf rtf snp png gif</p>

Method	Required/Optional	Type	Length	Description/Values
setFaxHeader	Optional	Alphanumeric	80	<p>Method used to override the default fax header.</p> <p>Client defined fax header to be used during this transmission.</p> <p><u>Please review the section on “Dynamic Fax Headers” for more information.</u></p>
setFineResolution	Optional	Boolean	N/A	<p>Method to override the default “standard” resolution.</p> <p>Passing true to this method will override the default resolution and cause <i>eFax Developer</i>™ to set the transmission resolution to “fine” instead of “standard.”</p> <p><u>Transmissions requested with "fine" resolution will incur a price premium.</u></p>
setHighPriority	Optional	Boolean	N/A	<p>Method used to override the default ("normal") priority.</p> <p>Passing true to this method will override the default priority and cause <i>eFax Developer</i>™ to set the transmission priority to "high" instead of "normal."</p> <p><u>Transmissions requested with "high" priority will incur a price premium.</u></p>

Method	Required/Optional	Type	Length	Description/Values
setHTMLResponse	Optional	Boolean	N/A	<p>Method used to indicate an HTML response is desired.</p> <p>Passing true to this method will override the default setting and cause eFax Developer™ to respond to client requests with formatted HTML instead of XML.</p>
setNoDuplicates	Optional	Boolean	N/A	<p>Method used to prevent duplicate transmission identifiers from being submitted.</p> <p>Passing true to this method will override the default setting and cause eFax Developer™ to verify that the transmission identifier has not already been used by a previous transmission.</p> <p>When enabled, eFax Developer™ will fail the transmission if a duplicate transmission identifier exists.</p>
setPassword	Required	Alphanumeric	20	<p>Method used to set the client's password.</p> <p>This value is provided at setup time and is required for authentication.</p>
setReadTimeout	Optional	Numeric		<p>Method used to control the read timeout for the response in milliseconds.</p> <p>A zero value is interpreted as an infinite timeout.</p>
setRecipientCompany	Optional	Alphanumeric	50	<p>The recipient's company name when supplied will be merged into the fax header line.</p> <p><u>Please review the section on "Dynamic Fax Headers" when using that option.</u></p>

Method	Required/Optional	Type	Length	Description/Values
setRecipientFax	Required	Alphanumeric	25	<p>The recipient's fax number.</p> <p>Designate international fax numbers by using the "011" international dialing prefix or start the fax number with a plus sign ("+").</p>
setRecipientName	Optional	Alphanumeric	50	<p>The recipient's name when supplied will be merged into the fax header line.</p> <p><u>Please review the section on "Dynamic Fax Headers" when using that option.</u></p>
setSelfBusy	Optional	Boolean	N/A	<p>Method used to change the "self-busy" option.</p> <p>Passing false to this method will override the default "self-busy" option and cause <i>eFax Developer</i>TM to set the transmission "self-busy" option to "disable" instead of "enable."</p> <p><i>eFax Developer</i>TM prevents multiple fax channels from simultaneously dialing the same fax number. Disabling this option will allow a single fax number to be dialed simultaneously by multiple fax channels.</p>
setTransmissionID	Optional	Alphanumeric	15	<p>Method used to set the client generated transmission identifier.</p> <p>This value is a unique client specified number used to identify a transmission. This optional value can be searched for during status request processing.</p> <p>This value should be a unique alphanumeric value when used.</p>

Method	Required/Optional	Type	Length	Description/Values
setTSID	Optional	Alphanumeric	20	<p>Method used to set the Transmitting Subscriber Identification or TSID.</p> <p>eFax Developer™ will transmit the account's fax number as the TSID.</p> <p>The value passed to this method overrides the default by substituting the value provided.</p>
setUserName	Required	Alphanumeric	20	<p>Method used to set the client's user name.</p> <p>This value is provided at setup time and is required for authentication.</p>

Outbound Response Accessor Methods (*OutboundResponse*)

Method	Always filled	Type	Length	Description/Values
getDOCID	No	Alphanumeric	8	<i>eFax Developer</i> TM transmission identifier.
getErrorLevel	Upon error	Alphanumeric	6	“User” “System” Defines the level of error. A “User” level can be handled by the client, while “System” generated errors will require <i>eFax Developer</i> TM Support intervention.
getErrorMessage	Upon error	Alphanumeric	ANY	The generated error message.
getRawResponse	Yes	Alphanumeric	ANY	The raw response returned by <i>eFax Developer</i> TM .
getTransmissionID	No	Alphanumeric	15	Client transmission identifier
isHTMLResponse	Yes	Boolean	N/A	Indication whether or not the response was returned as HTML.
isApproved	Yes	Boolean	N/A	Indication whether or not the request was approved by <i>eFax Developer</i> TM .

Status Request Mutator Methods (*StatusRequest*)

Method	Required/Optional	Type	Length	Description/Values
setAccountID	Required	Alphanumeric	10	Method used to set the client's account identifier. This value is provided at setup time and is required for authentication.
setConnectionTimeout	Optional	Numeric		Method used to control the connection timeout for the request in milliseconds. A zero value is interpreted as an infinite timeout.
setDOCID	Optional	Alphanumeric	8	<i>eFax Developer™</i> transmission identifier.
setHTMLResponse	Optional	Boolean	N/A	Method used to indicate an HTML response is desired. Passing true to this method will override the default setting and cause eFax Developer™ to respond to client requests with formatted HTML instead of XML.
setPassword	Required	Alphanumeric	20	Method used to set the client's password. This value is provided at setup time and is required for authentication.
setReadTimeout	Optional	Numeric		Method used to control the read timeout for the response in milliseconds. A zero value is interpreted as an infinite timeout.
setTransmissionID	Optional	Alphanumeric	15	Client transmission identifier
setUserName	Required	Alphanumeric	20	Method used to set the client's user name. This value is provided at setup time and is required for authentication.

Status Response Accessor Methods (*StatusResponse*)

Method	Always Filled	Type	Length	Description/Values
getBaudRate	No	Alphanumeric	8	Baud Rate used for this fax transmission. May or may not exist depending on the current status of the transmission.
getClassification	No	Alphanumeric	ANY	The status classification. May or may not exist depending on the current status of the transmission.
getCustomerID	No	Alphanumeric	50	The client specified customer identifier.
getDOCID	No	Alphanumeric	8	<i>eFax Developer™</i> transmission identifier.
getDuration	No	Alphanumeric	10	Actual duration in minutes for this fax transmission. May or may not exist depending on the current status of the transmission.
getErrorLevel	Upon error	Alphanumeric	6	“User” “System” Defines the level of error. A “User” level can be handled by the client, while “System” generated errors will require <i>eFax Developer™</i> Support intervention.
getErrorMessage	Upon error	Alphanumeric	ANY	The generated error message.

Method	Always Filled	Type	Length	Description/Values
getLastDate	No	Alphanumeric	10	<p>Last attempt date for this fax.</p> <p>May or may not exist depending on the current status of the transmission.</p> <p>Format is mm/dd/yyyy.</p>
getLastTime	No	Alphanumeric	8	<p>Last attempt time for this fax.</p> <p>May or may not exist depending on the current status of the transmission.</p> <p>Time zone is PST. Format is hh:mm:ss (24 hour).</p>
getMessage	Yes	Alphanumeric	100	The status message for this fax transmission.
getNextDate	No	Alphanumeric	10	<p>Next attempt date for this fax.</p> <p>May or may not exist depending on the current status of the transmission.</p> <p>Format is mm/dd/yyyy.</p>
getNextTime	No	Alphanumeric	8	<p>Next attempt time for this fax.</p> <p>May or may not exist depending on the current status of the transmission.</p> <p>Time zone is PST. Format is hh:mm:ss (24 hour).</p>
getOutcome	No	Alphanumeric	100	<p>The outcome text message.</p> <p>May or may not exist depending on the current status of the transmission.</p>

Method	Always Filled	Type	Length	Description/Values
getPagesScheduled	No	Alphanumeric	ANY	Scheduled number of pages for this fax transmission. May or may not exist depending on the current status of the transmission.
getPagesSent	No	Alphanumeric	ANY	Number of pages successfully sent for this fax transmission. May or may not exist depending on the current status of the transmission.
getRawResponse	Yes	Alphanumeric	ANY	The raw response returned by eFax Developer™.
getRecipientCompany	No	Alphanumeric	20	The recipient's company name.
getRecipientFax	No	Alphanumeric	25	The recipient's fax number.
getRecipientName	No	Alphanumeric	50	The recipient's name.
getRemoteCSID	No	Alphanumeric	ANY	The CSID transmitted to. May or may not exist depending on the current status of the transmission.
getRetries	No	Alphanumeric	3	Actual number of retry attempts for this fax transmission. May or may not exist depending on the current status of the transmission.
getTransmissionID	No	Alphanumeric	15	Client transmission identifier
isHTMLResponse	Yes	Boolean	N/A	Indication whether or not the response was returned as HTML.
isApproved	Yes	Boolean	N/A	Indication whether or not the request was approved by eFax Developer™.

Final Disposition Accessor Methods (*DispositionCatcher*)

Method	Always filled	Type	Length	Description/Values
getCompletionDate	Yes	Alphanumeric	20	Fax completion date and time. Time zone is PST. Format is yyyy-mm-dd hh:mm:ss (24 hour).
getDOCID	Yes	Alphanumeric	8	<i>eFax Developer</i> TM transmission identifier.
getDuration	No	Alphanumeric	5	Transmission time in minutes.
getFaxNumber	Yes	Alphanumeric	25	Recipient's fax number.
getFaxStatus	Yes	Numeric	5	Numeric field indicating the fax status. "0" indicates a successful transmission while all other values indicate an error code which can be cross-referenced with an <i>eFax Developer</i> TM supplied table.
getNumberOfRetries	No	Numeric	2	Number of times the fax was attempted before success or failure.
getPagesSent	No	Numeric	3	The number of pages sent.
getPassword	Yes	Alphanumeric	20	User Password
getRecipientCSID	No	Alphanumeric	20	The station identifier, when supplied by the receiving fax machine upon successful transmission.
getTransmissionID	No	Alphanumeric	15	Client transmission identifier
getUserName	Yes	Alphanumeric	20	User Name

HTML Response

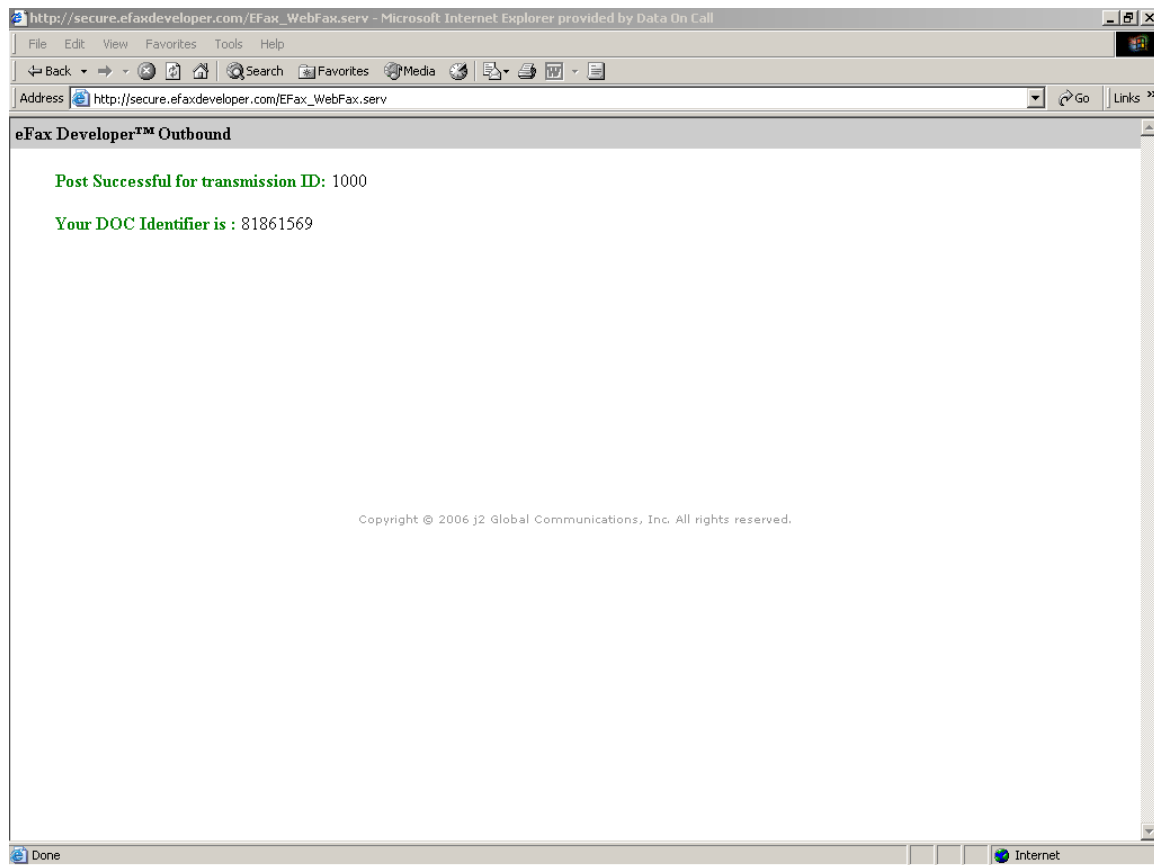
Outbound Response HTML

By default, eFax Developer™ responds to all requests it receives with an XML-formatted response. If an HTML-formatted response is desired, set the `OutboundRequest.setHTMLResponse(boolean)` to true.

When the `OutboundRequest.setHTMLResponse(boolean)` method is set to true, eFax Developer™ will respond back to the `sendFax()` request with an HTML-formatted response. Client processing can retrieve the HTML response from the `OutboundResponse` object via the `getRawResponse()` accessor method.

When an HTML-formatted response is requested, client processing will be limited to the HTML response returned by the `getRawResponse()` method. With the exception of Boolean accessors `isHTMLResponse()` or `isApproved()`, attempting to retrieve response data via an accessor method will result in an empty string being returned when an HTML response is indicated.

The following example shows an HTML-formatted response returned on a successful `sendFax()` request.



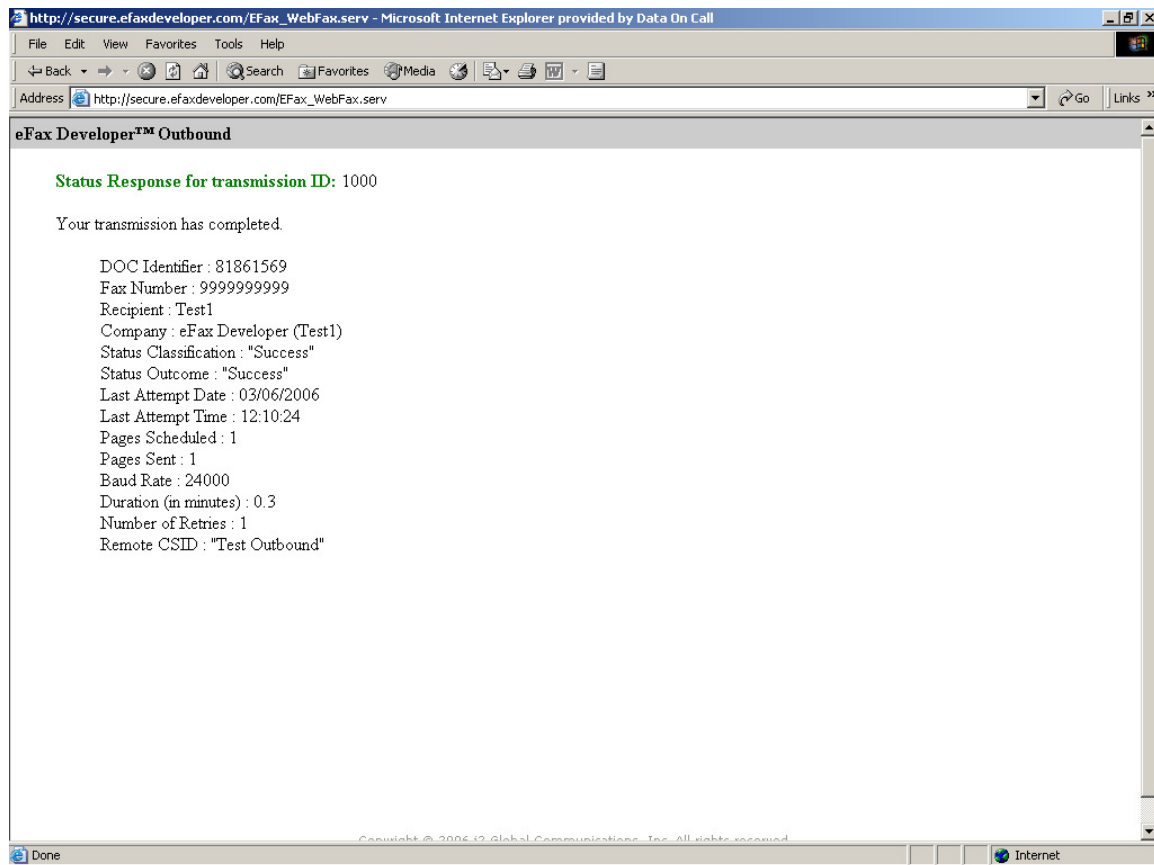
Status Response HTML

By default, eFax Developer™ responds to all requests it receives with an XML-formatted response. If an HTML-formatted response is desired, set the `StatusRequest.setHTMLResponse(boolean)` to true.

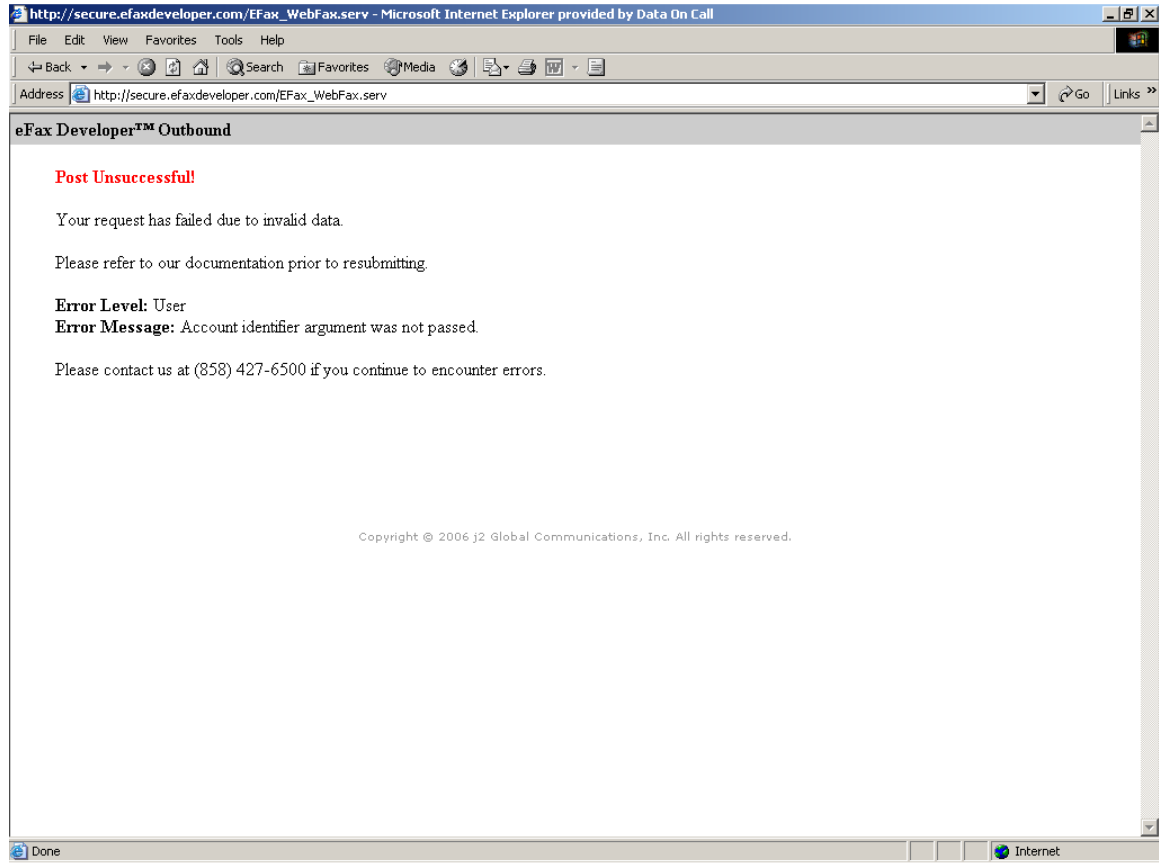
When the `StatusRequest.setHTMLResponse(boolean)` method is set to true, eFax Developer™ will respond back to the `getStatus()` request with an HTML-formatted response. Client processing can retrieve the HTML-formatted response from the `StatusResponse` object via the `getRawResponse()` accessor method.

When an HTML-formatted response is requested, client processing will be limited to the HTML response returned by the `getRawResponse()` method. With the exception of Boolean accessors `isHTMLResponse()` or `isApproved()`, attempting to retrieve response data via an accessor method will result in an empty string being returned when an HTML response is indicated.

The following example shows an HTML-formatted response returned on a successful `getStatus()` request.



The following example shows an HTML-formatted response returned on a failed sendFax() request.



Code Samples

Outbound Request

```
// Instantiate a new OutboundRequest object
OutboundRequest req = new OutboundRequest();

// Set your eFax Developer™ outbound account identifier (required)
req.setAccountID("1234567890");
// Set your eFax Developer™ outbound user name (required)
req.setUserName("abcdefg");
// Set your eFax Developer™ outbound password (required)
req.setPassword("hijklmn");

// Set the recipient fax number for this transmission (required)
req.setRecipientFax("8001234567");

// Set your desired disposition level
req.setDispositionLevel(OutboundRequest.DL_BOTH);
// Set your desired endpoint for final disposition POST
req.setDispositionsTo("https://your.endpoint.com/dispo.jsp");

// Instantiate a DocumentBundler object
DocumentBundler docs = new DocumentBundler();

try {
    // Add document(s) to the DocumentBundler object by path
    docs.add("C:\\your\\document\\path\\doc1.doc");
    docs.add("C:\\your\\document\\path\\doc2.doc");

    // Set the documents to be faxed for this transmission (required)
    req.setDocuments(docs);

    // POST the outbound transmission request to eFax Developer™
    OutboundResponse resp = req.sendFax();

    // When eFax Developer™ approved the outbound request
    if (resp.isApproved()) {
        // Parse the response
        System.out.println("POST received for DOCID: " + resp.getDocID());
    }
    // Otherwise, eFax Developer™ rejected the outbound request
    else {
        // Parse the error response
        System.out.println("POST failed: " + resp.getErrorMessage());
    }
}
catch (Exception e) {}
catch (Error err) {}
```

Status Request

```
// Instantiate a new StatusRequest object
StatusRequest req = new StatusRequest();

// Set your eFax Developer™ outbound account identifier (required)
req.setAccountID("1234567890");
// Set your eFax Developer™ outbound user name (required)
req.setUsername("abcdefg");
// Set your eFax Developer™ outbound password (required)
req.setPassword("hijklmn");

// Search by your transmission identifier
req.setTransmissionID("1234567890");

try {

    // Retrieve the transmission status from eFax Developer™
    StatusResponse resp = req.getStatus();

    // When eFax Developer™ approved the status request
    if (resp.isApproved()) {
        // Parse the response
        System.out.println("POST received for DOCID: " + resp.getDocID());
    }
    // Otherwise, eFax Developer™ rejected the status request
    else {
        // Parse the error response
        System.out.println("POST failed: " + resp.getErrorMessage());
    }
}
catch (Exception e) {}
catch (Error err) {}
```

Final Disposition (with DispositionCatcher)

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

    response.setContentType("text/html");
    PrintWriter out = response.getWriter();

    out.println("<html>");
    out.println("<head><title>DispositionCatcherServlet</title></head>");
    out.println("<body>");

    // Retrieve the disposition "xml" parameter value from the request
    String xml = request.getParameter("xml");

    try {

        // Establish a DispositionCatcher object
        DispositionCatcher dc = new DispositionCatcher(xml);

        // Parse the DispositionCatcher
        System.out.println("");
        System.out.println("-----");
        System.out.println("UserName: "          + dc.getUserName());
        System.out.println("Password: "          + dc.getPassword());
        System.out.println("TransmissionID: "    + dc.getTransmissionID());
        System.out.println("DOCID: "            + dc.getDOCID());
        System.out.println("FaxNumber: "         + dc.getFaxNumber());
        System.out.println("CompletionDate: "    + dc.getCompletionDate());
        System.out.println("FaxStatus: "         + dc.getFaxStatus());
        System.out.println("RecipientCSID: "     + dc.getRecipientCSID());
        System.out.println("Duration: "          + dc.getDuration());
        System.out.println("PagesSent: "         + dc.getPagesSent());
        System.out.println("NumberOfRetries: "   + dc.getNumberOfRetries());
        System.out.println("-----");
        // Reply back to eFax Developer™ indicating that the final disposition
        // has been successfully received by your processing.
        out.println("Post Successful");

    }
    catch (Exception e) {
        e.printStackTrace();
        e.printStackTrace(out);
    }
    catch (Error err) {
        err.printStackTrace();
        err.printStackTrace(out);
    }
    finally {
        out.println("</body></html>");
        out.flush();
        out.close();
    }
}
```

User Level Error Messages

Requests sent to *eFax Developer™* will be validated against an XSD before being authenticated by *eFax Developer™* processing. If the request fails validation or authentication, a response will be returned back to the client indicating why the request failed. It is up to the client to decide how these error messages should be handled.

Error Messages Returned by eFax Developer™ Authentication

The following error messages are returned by *eFax Developer™* authentication. When a request is received by *eFax Developer™*, the request will be authenticated against the client's online profile. If the request fails authentication for any reason, an error message will be returned to the client.

Error Message: Account identifier argument was not passed.

Problem:

The account identifier was not included as part of the request.

Solution:

Be sure that the account identifier is passed to the *OutboundRequest* or *StatusRequest* object's `setAccountID()` method.

Error Message: Login not successful. UserName, Password or IP Address failed login validation.

Problem:

The account could not be validated using the username, password or IP address provided.

Solution:

Log in at <https://secure.efaxdeveloper.com>.

From the "Services" page, click on the [eFax Developer](#) link.

From the "Inbox" page, click the "Settings" icon.

From the "Outbound Settings" tab, make sure the username and password is in sync with the values passed to the *OutboundRequest* or *StatusRequest* object's `setUserName()` and `setPassword()` methods.

If the account is configured to validate sending IP addresses, make sure the sending IP is included in the list of Approved IP Addresses displayed on the "Outbound Settings" tab.

Error Message: Request content length (nnnnnnnn) has exceeded our 30MB ceiling.

Problem:

The total content length of the POST request exceeded 30MB and was rejected.

Solution:

Break up the transmission into separate requests if possible. Please contact *eFax Developer*TM Support if you continue to encounter this exception.

Error Message: This account is no longer active. Please contact eFax Developer Customer Support.

Problem:

The account is no longer active or closed.

Solution:

Contact *eFax Developer*TM Customer Support for further assistance.

Error Message: Demo account has reached its maximum allotment. Please contact eFax Developer Customer Support.

Problem:

A demo account is expired or reached the maximum number of allowed faxes for the demo.

Solution:

Contact *eFax Developer*TM Customer Support for further assistance.

Error Message: This account has been placed on hold. Please contact eFax Developer Customer Support.

Problem:

The account is on hold.

Solution:

Contact *eFax Developer*TM Customer Support for further assistance.

Error Message: Requests from this account are prohibited. Please contact eFax Developer Customer Support.

Problem:

The account has been blocked from submitting requests.

The account is continuously submitting bad requests or bad attachments, or is misusing the service in some manner. This is an extreme measure imposed by eFax Developer™ Support after all other avenues of communication with the primary account holder have been exhausted.

Solution:

Contact *eFax Developer*™ Customer Support for further assistance.

Error Message: TransmissionID required when NoDuplicates option is enabled.

Problem:

The *OutboundRequest* object's `setNoDuplicates()` method was set to true but a transmission identifier was not passed to the object's `setTransmissionID()` method.

Solution:

Set the *OutboundRequest* object's `setNoDuplicates()` method to false, or set a non-empty transmission identifier to the object's `setTransmissionID()` method.

Error Message: Duplicate TransmissionID not allowed when NoDuplicates option is enabled.

Problem:

The *OutboundRequest* object's `setNoDuplicates()` method was set to true and the transmission identifier passed to the object's `setTransmissionID()` method was already submitted.

Solution:

Handle as desired in the client-side process.

Error Message: A transmission or DOC identifier is required for a status request.

Problem:

The status request did not include a transmission or DOC identifier.

Solution:

Set a non-empty transmission or DOC identifier via the *OutboundRequest* object's `setTransmissionID()` or `setDOCID()` methods prior to submitting the status request.

Error Messages Returned by XSD Validation

The following error messages are returned by *eFax Developer*™ XSD validation. XML that contains unexpected or unrecognized parameters, or contains values that fail to meet criteria defined in this document will be rejected.

Error Message: the value is not a member of the enumeration: (“XXX/XXX/XXX”)

Problem:

An unrecognized parameter value was passed in the request.

Solution:

Make sure parameters passed to the *OutboundRequest* object’s `setDispositionLevel()` method meet specifications outlined in this document.

Consider using predefined *OutboundRequest.DL_XXXX* variables when specifying disposition levels. For example, `setDispositionLevel(OutboundRequest.DL_Error)` can be used to return disposition notifications on transmission errors only.

Error Message: the value is not a member of the enumeration.

Problem:

An unrecognized parameter value was passed in the request.

Solution:

Make sure known file types are being passed to the *DocumentBundler* object’s `add()` method and that parameters passed to the *OutboundRequest* object’s `setDispositionLanguage()` method meet specifications outlined in this document.

Consider using predefined *OutboundRequest.ISO_XXXX* variables when specifying a disposition language. For example, `setDispositionLanguage(OutboundRequest.ISO_ITALIAN)` can be used to return final disposition emails in Italian if desired.

Error Message: the length of the value is *nnn*, but the required minimum is *nnn*.

Problem:

The *OutboundRequest* object’s `setRecipientFax()` value did not meet minimum length requirements.

Solution:

Make sure the value passed to the *OutboundRequest* object’s `setRecipientFax()` method meets the minimum required length outlined in this document.

Error Message: the length of the value is *nnn*, but the required maximum is *nnn*.

Problem:

A value was passed that exceeded a maximum length requirement.

Solution:

Make sure values passed to these methods meet the specifications outlined in this document.

```
setCustomerID()  
setTransmissionID()  
setDispositionsTo(String)  
setRecipientFax()
```

Error Message: Content of element “XXXX” is incomplete

Problem:

The element specified in the error message did not contain a value.

Solution:

Make sure a value is being passed via the referenced setter method of the *OutboundRequest* object.