# CFX_AIM_JAVA ™

## A payment card gateway solutions for ColdFusion ® users of Authorize.Net's Advanced Integration Method (AIM)


# Installation & User Guide

| Software Information | Installation & User Guide Information |
|---|---|
| Software Version: 1.0<br>Published: 01/25/2012 | Word Document: CFX_AIM_JAVA.doc<br>Published: 01/25/2012 |

CFXWorks, Inc.
303 Arbor Green Lane, Alpharetta, GA 30004

Email: sales@CFXWorks.com http://www.CFXWorks-Coldfusion.com

Printed in the United States of America.

 

 

# 1. INTRODUCTION

## 1.1. Thank You

Thank you for purchasing or downloading our CFX_AIM_JAVA tag. It is our intention to provide a tag that adds value to your ColdFusion efforts by improving your ability to quickly develop a secure payment card solution. If you feel a need to contact me please send me an email on contact me directly at:

Al Nickles
anickles@cfxworks.com
678-455-0952

## 1.2. Definitions

Authorize.Net is sometimes referred to in this document as “AN”. Also, Authorize.Net’s Advanced Integration Method is referred to as “AIM”.

## 1.3. CFXWorks

CFX_AIM_JAVA is an updated version of our original CFX_AIM tag that was written in C++ and packaged as a ColdFusion tag. The original tag was written in C++ and compiled to a 32-bit Windows DLL. Unfortunately Microsoft decided not to support 32-bit DLLs on several of their new 64-bit platforms and provided no clear migration path from 32-bit to 64-bit platforms. We also got fed up with working with proprietary programming languages and compiles that only ran on specific operating system environments. Therefore, we choose to rewrite our original tag in Java and call it CFX_AIM_JAVA. The new tag features the following improvements:

- The new tag is provided at “no charge” to users who register their Authorize.Net account as described in Section 1.5 paragraph 1.
- The new tag will run on any ColdFusion platform running Java 1.5 or Java 1.6. We believe that it will also run on Java 1.7, but have not tested it on 1.7.
- The new tag will run on both 32-bit and 64-bit platforms.

 

- The new tag has been updated to support Authorize.Net's Version 2.0 Developers Guide. As of 01/25/12 this is the current version of the AIM specification.
- New input and output parameters are supported. See ***Sections 3.1 and 3.2*** of this guide.
- The tag has been updated to support the security requirements defined by PCI-DSS that require one to encrypt using strong encryption, or mask, sensitive card data.

CFX_AIM_JAVA is one of several ColdFusion custom tags that offer to assist ColdFusion programmers develop eCommerce solutions. Specifically, our CryptoXpress ColdFusion tag is used by many merchants requiring strong encryption capabilities. This tag is also 32-bit and 64-bit and will run on all the platforms supported by ColdFusion. You can view this offering on our web site at:

http://www.cfxworks-coldfusion.com/index.cfm?fuseaction=cEncryption.overviewEncryption

## 1.4. New Verses Changed Features

For past users of the CFX_AIM tag, we have attempted to identify in this document what is "new" of "changed" from the original tag. We did this using the following color codes in this document:

NEW is color coded in GREEN

CHANGED is color coded in YELLOW

## 1.5. Why Purchase CFX_AIM_JAVA

Payment gateways facilitate electronic commerce by enabling merchants to accept payment cards and electronic checks as methods of payment for goods and services sold online. The CFX_AIM_JAVA tag acts as a bridge between the merchant's Website and the organization that processes the payment transactions. Typically, payment data is collected online from the shopper and submitted to the gateway for real-time authorization.

Authorization is the process of checking the validity, and in some cases, the availability of funds required to complete a transaction. To authorize a given payment card transaction, the gateway transmits the required transaction information to the appropriate financial institutions for validation, then returns the response (approved or declined) from the institution to the merchant or customer.

 

Our tag supports Authorize.Net's Advanced Integration Method (AIM). AIM targets merchants who process Card-Not-Present transactions. In a Card-Not-Present transaction, the merchant and the shopper are not in the same physical location and the customer usually calls in the payment data or keys in the details of the credit card on a Website. All e-commerce and mail order/telephone order (MOTO) transactions are Card-Not-Present transactions.

## 1.6. Tag Availability and Pricing

**CFX_AIM_JAVA is available as a "NO CHARGE" upgrade to current licensees of our CFX_AIM tag. It is also available at "NO CHARGE" to new users who register CFXWorks as their Affiliate Reseller with Authorize.Net. Please use the following link to complete registration and apply for a new Authorize.Net merchant account:**

https://ems.authorize.net/oap/home.aspx?SalesRepID=98&ResellerID=6199.

If you need assistance please call the Authorize.Net support line at 1-877-447-3938.

If you want to use the CFX_AIM_JAVA tag but already have an existing Authorize.Net merchant account not registered to CFXWorks, you must close your existing account and register a new one using the above link.

CFXWorks provides developer copies of our tag to merchants wanting to "kick the tires". The developer copy allows one to build and test their payment solution prior to completing the above registration process or purchasing an Authorize.Net merchant account. The developer copy of the tag runs only test transactions. It will not run production transactions.

When you are ready for your production version of the tag, CFXWorks needs to know your (X_LOGIN) value to embed it in the tag. Once we have received this value and verify that your account has been activated by Authorize.Net, we will build you a production copy of the tag and email it to you.

At any time you can contact CFXWorks via email at support@cfxworks.com or by phone at 678-455-0952.

 

# 2. TAG DETAILS

## 2.1. Fast Path Implementation Process

If you want to get up and running quickly, CFXWorks recommends that you do the following:

- Step 1 - Download and print a copy of the Authorize.NET AIM Developer Guide. You can download a copy at the following link:

  http://www.authorize.net/support/AIM_guide.pdf

  This guide explains in detail, how to use the AIM method, and it documents the syntax and content of the response you receive from Authorize.Net when you execute a transaction. This guide is over 80 pages long, but fortunately if you are using our tag you can ignore all but a few pages of this document. The two sections of this document that you will find interesting are:

  1. Sections 2 & 3 – Transaction Data Requirements – In this section you will find detailed descriptions of all the parameters supported by Authprize.Net. Our tag supports a subset of these parameters (See Section 3.1 of this Guide)

  2. Section 4 – Transaction Response –In this section you will find a detailed description of the Authorize.Net response parameters.

- Step 2 - If you need to install a production copy of the tag, please provide us your Login ID so we can build you the production version of the tag.

- Step 3 - Install the CFX_AIM_JAVA tag as described **in *Section 3.3 Installation*** of this guide.

- Step 4 - Modify one of the sample ColdFusion programs provided with this tag to contain your x_Login and x_Tran_Key values. Experiment with this program until you understand exactly how the CFX_AIM_JAVA tag functions.

- Step 5 - Once you fully understand how the demo program works, you can integrate the code into your shopping cart application!

 

## 2.2. What Specifically Does the Tag Do?

Our tag performs the following tasks:

- It parses the request parameters passed from your ColdFusion program. In the original tag we validated the syntax of each parameter and if we discovered an error, created or own error code. Our error codes on top of the Authorize.Net error codes created much confusion. Therefore we decided in the new tag not to duplicate Authorize.Net's validation and to in nearly every case simply rely on them to validate the syntax of the parameters. Please refer to the AN Developer Guide for a definition of these error codes.

- It validates that the credit card number provided is valid based on LUHN Formula (Mod 10) for validation of credit card account numbers. Please see ***Appendix B*** for an explanation of this formula. If an error is found, the tag posts an error in the ColdFusion variable "RC" and immediately returns control to your ColdFusion application.

- It formats and assembles the request as per the Authorize.Net specification.

- It sends this request string directly to Authorize.Net using a secure sockets (SSL) connection. The SSL protocol encrypts the data.

- It reads the response from Authorize.Net. The SSL connection decrypts the data and performs authentication.

- It checks the syntax of the response to assure that it is valid.

- It supports an option to create a log of the request sent and response received from Authorize.Net. If a log is created, it is written in a format consistent with PCI-DSS.

- It extracts the following data from the response and populates the following ColdFusion variables:

| | |
|---|---|
| RC | CFXWorks return code |
| X_RESPONSE | AN's exact response |
| X_RESPONSE_CODE | Response Code |
| X_RESPONSE_SUBCODE | Response subcode |
| X_RESPONSE_REASON_CODE | Response Reason Code |
| X_RESPONSE_REASON_TEXT | Response Reason Text |

| | |
|---|---|
| X_APPROVAL_CODE | Approval Code |
| X_AVS_RESULT_CODE | AVS Result Code |
| X_TRANS_ID | Transaction ID |
| X_HASH | MD5 Hash |
| X_RESPONSE_CARD_CODE | CVV2 Response Code |
| X_CAVV_RESPONSE_CODE | CAVV Verification Response |
| X_ACCOUNT_NUMBER | Masked Card Number |
| X_CARD_TYPE | Visa, MasterCard, etc. |

## 2.3. What Specifically Does Our Tag Not Do?

**Our tag does not route your transaction through another gateway** before connecting to Authorize.Net. We connect your ColdFusion application directly to Authorize.Net. This direct connection should improve your response time, it should improve the security of your connection, and it should improve the reliability of your connection. Why? Because there are no added gateways in the mix to slow you down, to introduce security flaws, or to introduce errors.

**Our tag is not a shopping cart application**. It is a secure credit card processing gateway that connects you directly to Authorize.Net. You must write your own shopping cart solution.

**Our tag does not log transactions to a database**. You may want to save your transaction data in a database. As an alternative you can use the tags log as your audit trail. If you create a database, make sure that you encrypt all sensitive card data before it is stored in the database using strong encryption. The encryption algorithm you use should support a least 128-bit encryption keys. Our CryptoXpress tag is used by many merchants to perform the required encryption.

 

## 2.4. X_LOGIN, X_TRAN_KEY & X_TEXT_REQUEST Parameters

**CFX_AIM_JAVA Test Tag**

- X_LOGIN - The X_LOGIN parameter must be coded. You can use a production X_LOGIN value or that of a valid Authorize.Net test account.
- X_TRAN_KEY – This value must be coded using the value set in the Authorize.Net Merchant Interface for the account being used.
- X_TEST_REQUEST – this value is by default set to "TRUE" by the tag.

**CFX_AIM_JAVA Production Tag**

Production copies of the tag do not require the X_LOGIN parameter to be coded. The X_LOGIN production value is hardcoded in the tag. Transactions are routed to the production server if "X_TEST_REQUEST=FALSE" is coded. Transactions are routed to the test server if "X_TEST_REQUEST=TRUE" is coded. The test server enables users to test applications without running live transactions.

The following figure illustrates how you can direct transactions to either the TEST or PRODUCTION server.

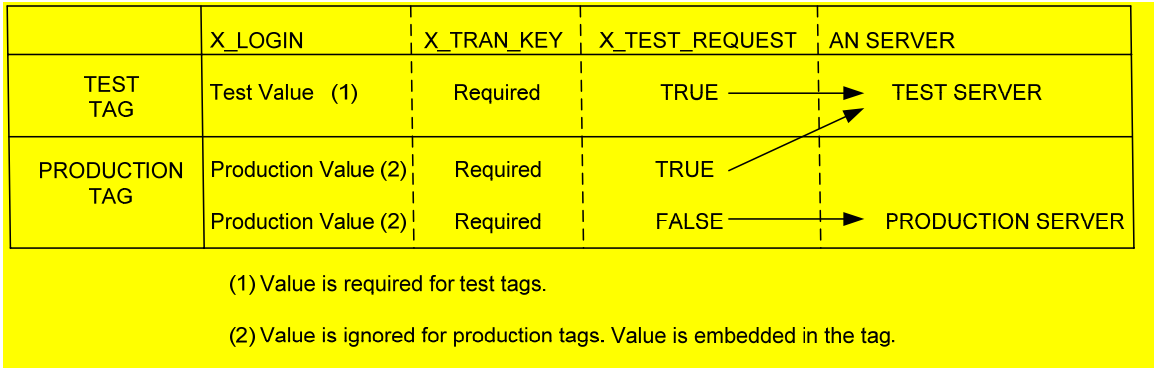


***Figure 1 – Test Verses Production Transactions***

## 2.5. X_PASSWORD

The X_PASSWORD parameter in no longer supported by Authorize.Net. Use X_TRAN_KEY instead.

## 2.6. X_DELIM_DATA & X_DELIM_CHAR Parameters

We strongly suggest coding the following parameters:

X_DELIM_DATA=TRUE
X_DELIM_CHAR="|"

If the above parameters are not coded, we will set them to these values by default. Unless these values are set as above the tag may not be able to parse the correct values from the Authorize.Net response and return them to the program calling the tag.

## 2.7. X_ENCAP_CHAR Parameter

The X_ENCAP_CHAR is used to tailor the format of the Authorize.Net response to a user-defined format. If this parameter is coded, only the X_RESPONSE and RC parameters of the tag, documented in ***Appendix A***, are returned by the tag. This parameter should only be used by ColdFusion programmers who want do write their own parsing routines to extract the Authorize.Net response values from the response.

## 2.8. Transaction Types (X_TYPE)

The CFX_AIM_JAVA tag supports the following credit card transaction types:

**AUTH_CAPTURE** – Transactions of this type will be sent for authorization. The transaction will be automatically processed and sent to settlement if approved.

**AUTH_ONLY** – Transactions of this type are submitted if the merchant wishes to validate the credit card for the amount of the goods sold. Authorize.Net will send this type of transaction to the financial institution for approval. However, it will not be sent to settlement. It the merchant does not act on the transaction within 30 days, the transaction will no longer be available for capture.

**PRIOR_AUTH_CAPTURE** – This transaction is used to request settlement of a transaction that was previously submitted as an AUTH_ONLY. The gateway will accept this transaction and initiate settlement if the following conditions are met:

 

- The transaction is submitted with the transaction ID, X_TRANS_ID, of the original authorization-only transaction, which needs to be settled.
- The transaction ID is valid and the system has a record of the original authorization-only transaction being submitted.
- The original transaction referred to is not already settled or expired or in error.
- The amount being requested for settlement in this transaction is less than or equal to the original authorized amount.

In no amount is submitted in this transaction, the gateway will initiate settlement for the amount of the originally authorized transaction.

**CREDIT** – This transaction is also referred to as a "Refund" and indicates to Authorize.Net that money should flow from the merchant to the customer. The gateway will accept a credit or a refund request if the transaction submitted meets the following conditions:

- The transaction is submitted with the transaction ID, X_TRANS_ID, of the original transaction, against which the credit is being issued.
- Authorize.Net has a record of the original transaction.
- The original transaction has been settled.
- The sum amount being submitted in the Credit transaction and all the credits submitted against the original transaction is less than the original transaction amount.
- The first and last four digits of the credit card number submitted with the credit transaction match the first and last four digits of the credit card number used in the original transaction.

If no credit card number is submitted with the transaction and all the checks pass, the Credit will be issued against the credit card used in the original transaction.

**CAPTURE_ONLY** – This is a request to capture a transaction that was not submitted for authorization through a payment gateway. The gateway will accept this transaction if an authorization code is submitted. The X_AUTH_CODE is a required field for CAPTURE_ONLY transactions.

**VOID** – This transaction is an action on a previous transaction and is used to cancel the previous transaction and ensure it does not get sent for settlement. It can be done on any type of transaction. The transaction will be accepted by the gateway if the following conditions are met:

 

- The transaction is submitted with the transaction ID, X_TRANS_ID, of the original transaction that is to be voided.
- Authorize.Net has a record of the original transaction.
- The original transaction has not been settled.

# 3. TECHNICAL ISSUES

## 3.1. Tag Input Parameters

The input parameters supported by CFX_AIM_JAVA tag include:

| Input Parameters | Description |
|---|---|
| LOGFILE | Fully qualified logfile name.<br>Example: "c:/aim.log" |
| DISPLAY | 0 - No log written<br>1 - display debug and license information but don't send transaction to AN.<br>2 - display license information but don't send transaction to AN.<br>3 – Send transaction to AN and log AN's response.<br>4 – Send transaction to AN and log both the request and response. |
| X_LOGIN | See Section 2.4 of this document.<br>Please see the Authorize.Net AIM Developer Guide. |
| X_TRAN_KEY | See Section 2.4 of this document.<br>Please see the Authorize.Net AIM Developer Guide. |
| X_VERSION | Please see the Authorize.Net AIM Developer Guide. |
| X_TYPE | See Section 2.8 of this document.<br>Please see the Authorize.Net AIM Developer Guide. |
| X_METHOD | Only "CC" currently supported. Defaults to "CC".<br>Please see the Authorize.Net AIM Developer Guide. |
| X_CARD_NUM | Please see the Authorize.Net AIM Developer Guide. |
| X_EXP_DATE | Please see the Authorize.Net AIM |

 

| | |
|---|---|
| | Developer Guide. |
| X_AMOUNT | Please see the Authorize.Net AIM Developer Guide. |
| X_CARD_CODE | Please see the Authorize.Net AIM Developer Guide. |
| X_TRANS_ID | Please see the Authorize.Net AIM Developer Guide. |
| X_AUTH_CODE | Please see the Authorize.Net AIM Developer Guide. |
| X_TEST_REQUEST | See Section 2.4 of this document. Please see the Authorize.Net AIM Developer Guide. |
| X_DUPLICATE_WINDOW | Please see the Authorize.Net AIM Developer Guide. |
| X_INVOICE_NUM | Please see the Authorize.Net AIM Developer Guide. |
| X_DESCRIPTION | Please see the Authorize.Net AIM Developer Guide. |
| X_FIRST_NAME | Please see the Authorize.Net AIM Developer Guide. |
| X_LAST_NAME | Please see the Authorize.Net AIM Developer Guide. |
| X_COMPANY | Please see the Authorize.Net AIM Developer Guide. |
| X_ADDRESS | Please see the Authorize.Net AIM Developer Guide. |
| X_CITY | Please see the Authorize.Net AIM Developer Guide. |
| X_STATE | Please see the Authorize.Net AIM Developer Guide. |
| X_ZIP | Please see the Authorize.Net AIM Developer Guide. |
| X_COUNTRY | Please see the Authorize.Net AIM Developer Guide. |
| X_PHONE | Please see the Authorize.Net AIM Developer Guide. |
| X_SHIP_TO_FIRST_NAME | Please see the Authorize.Net AIM Developer Guide. |
| X_SHIP_TO_LAST_NAME | Please see the Authorize.Net AIM Developer Guide. |
| X_SHIP_TO_COMPANY | Please see the Authorize.Net AIM Developer Guide. |
| X_SHIP_TO_ADDRESS | Please see the Authorize.Net AIM Developer Guide. |
| X_SHIP_TO_CITY | Please see the Authorize.Net AIM |

| | |
|---|---|
| | Developer Guide. |
| X_SHIP_TO_STATE | Please see the Authorize.Net AIM Developer Guide. |
| X_SHIP_TO_PHONE | Please see the Authorize.Net AIM Developer Guide. |
| X_SHIP_TO_ZIP | Please see the Authorize.Net AIM Developer Guide. |
| X_SHIP_TO_COUNTRY | Please see the Authorize.Net AIM Developer Guide. |
| X_TAX | Please see the Authorize.Net AIM Developer Guide. |
| X_FREIGHT | Please see the Authorize.Net AIM Developer Guide. |
| X_DUTY | Please see the Authorize.Net AIM Developer Guide. |
| X_TAX_EXEMPT | Please see the Authorize.Net AIM Developer Guide. |
| X_PO_NUM | Please see the Authorize.Net AIM Developer Guide. |
| X_DELIM_DATA | See Section 2.6 of this document.<br>Please see the Authorize.Net AIM Developer Guide.<br>By default set to "TRUE\|" |
| X_DELIM_CHAR | See Section 2.6 of this document.<br>Please see the Authorize.Net AIM Developer Guide.<br>By default set to "\|" |
| X_ENCAP_CHAR | See Section 2.7 of this document.<br>Please see the Authorize.Net AIM Developer Guide. |
| X_AUTHENTICATION_INDICATOR | Please see the Authorize.Net AIM Developer Guide. |
| X_CARDHOLDER_AUTHENTICATION_VALUE | Please see the Authorize.Net AIM Developer Guide. |
| X_EMAIL_CUSTOMER | Please see the Authorize.Net AIM Developer Guide. |
| X_EMAIL | Please see the Authorize.Net AIM Developer Guide. |

***Table 1 – CFX_AIM_JAVA Input Parameters***

 

## 3.2. Tag Output Parameters

The input parameters supported by CFX_AIM tag are as follows:

| Reply Parameters | Description |
|---|---|
| RC | See Appendix A |
| SEQUENCE_NBR | A number that can be used to cross reference a transaction to the log entry. |
| X_RESPONSE | Complete text of response message from Authorize.Net |
| X_RESPONSE_CODE | Indicates the result of the transaction:<br>1 = Approved<br>2 = Declined<br>3 = Error<br>4 = Held For Review |
| X_RESPONSE_SUBCODE | Please see the Authorize.Net AIM Developer Guide. |
| X_RESPONSE_REASON_CODE | Please see the Authorize.Net AIM Developer Guide. |
| X_RESPONSE_REASON_TEXT | Please see the Authorize.Net AIM Developer Guide. |
| X_RESPONSE_CARD_CODE | Please see the Authorize.Net AIM Developer Guide. |
| X_APPROVAL_CODE | Please see the Authorize.Net AIM Developer Guide. |
| X_AVS_RESULT_CODE | Please see the Authorize.Net AIM Developer Guide. |
| X_TRANS_ID | Please see the Authorize.Net AIM Developer Guide. |
| X_HASH | Please see the Authorize.Net AIM Developer Guide. |
| X_CAVV_RESPONSE_CODE | Please see the Authorize.Net AIM Developer Guide. |
| X_ACCOUNT_NUMBER | Please see the Authorize.Net AIM Developer Guide. |
| X_CARD_TYPE | Please see the Authorize.Net AIM Developer Guide. |

***Table 2 – CFX_AIM_JAVA Output Parameters***

For a summary of "RC" error codes please see ***Appendix A.***

 

## 3.3. Installation

**Step 1 – Unzip the distribution file.**

The tag is distributed in a zip file (CFX_AIM_JAVA..zip). The zip file contains the following files:

| File: | Description: |
|---|---|
| CFX_AIM_JAVA.jar | This is the tags executable code. |
| CFX_AIM_JAVA.doc | This document. |
| CFX_AIM_JAVA_License.doc | The license file. |
| AimError501.cfm | A sample CF program with a missing X_LOGIN value! Because of the error, the transaction is not forwarded to AN for execution. |
| AimError502.cfm | A sample CF program with a missing X_TRAN_KEY value! Because of the error, the transaction is not forwarded to AN for execution. |
| AimError503.cfm | A sample CF program using a CC number that fails the LUHN test. See ***Appendix B***. Because of the error, the transaction is not forwarded to AN for execution. |
| AimDisplayLicense.cfm | A sample CF program that displays the tags license. This sample program does not forward the transaction to AN for execution. |
| AimDebug.cfm | A sample CF program that displays the input parameters passed the tag and the license information. This sample program does not forward the transaction to AN for execution. |
| AimCheckAllParameters.cfm | A sample CF program that captures and displays all the input parameters supported by the tag and displays them. This sample program does not forward the transaction to AN for execution. |
| AimAuthCapture.cfm | A sample CF program that runs a AUTH_CAPTURE transaction and displays the results returned by Authorize.Net |

***Table3 – Distribution Zip File***

 

**Step 2 - Place the CFX_AIM_JAVA.jar file in the classpath directory** that you define in ColdFusion administration. For example place the tag in one of the directories specified in “Coldfusion Class Path”:

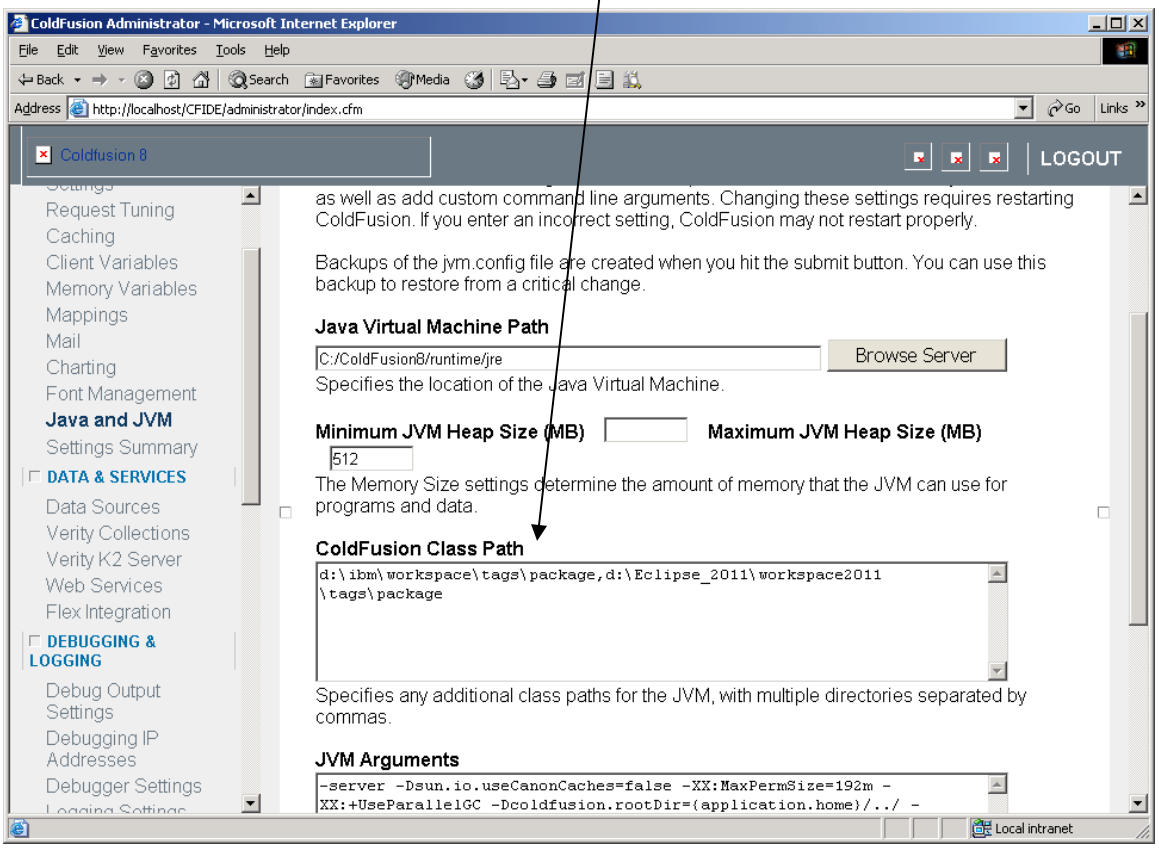


***Figure 2 – Setting Java’s Classpath In ColdFusion***

**Step 3 – Stop Coldfusion**

 

**Step 4 - Register the tag as follows:**

Register the tag. You can use any “Tag Name” however whatever name you define, this must be how you reference the tag in your code.

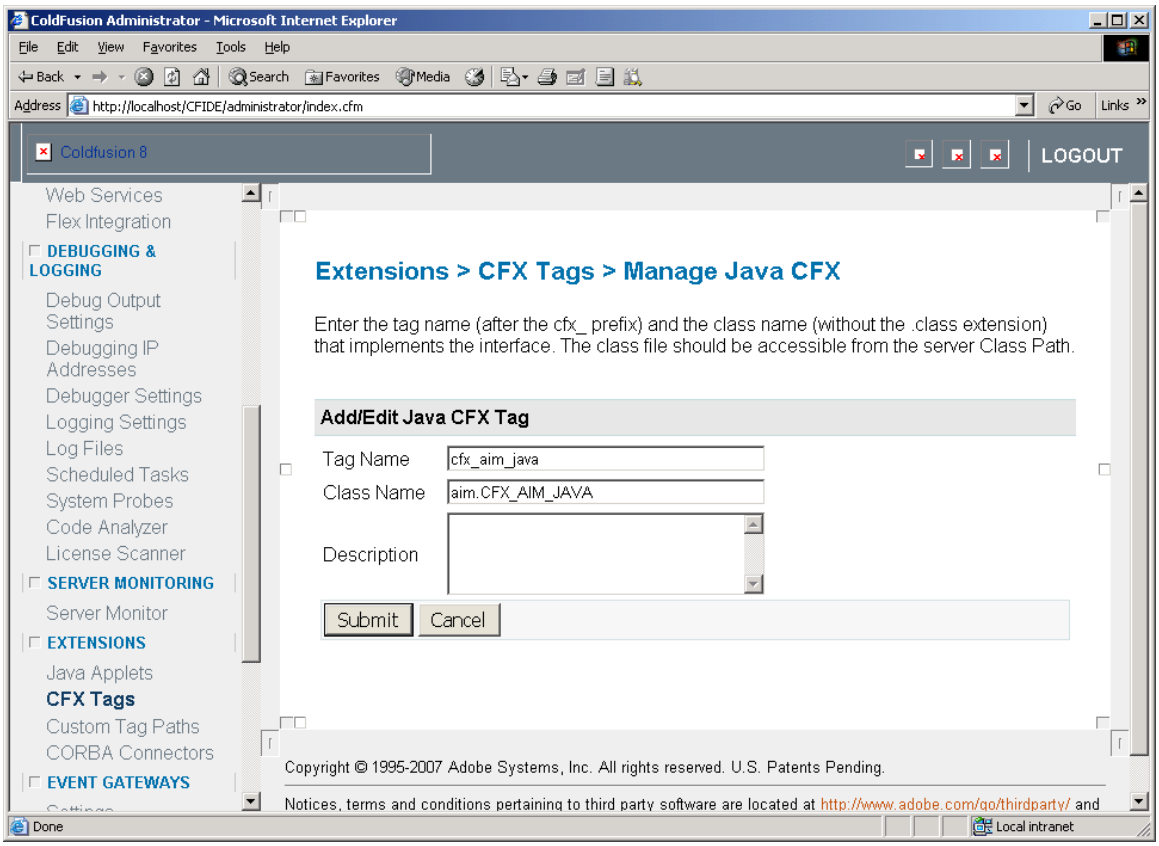


***Figure 3 – Registering the Tag***

**Step 3 – Start  Coldfusion**

## 3.4. Digital Certificates

Strictly speaking, our gateway does not require the merchant to purchase a digital certificate and install it on their web server to make the connection between the merchant and Authorize.Net. Why? Because the SSL transaction initiated by the CFX_AIM_JAVA gateway starts by sending a handshake to Autherize.Net's server. It is the responsibility of Authrize.Net's server's to send its certificate in the response. This certificate contains a public key associated with the server and other information, such as the owner of the certificate, its expiration date, and the fully qualified domain name associated with the server. The tag then uses the public key contained in this certificate to encrypt data sent to Authorize.Net. During the connection process, Authorize.Net's server proves its identity by using its private key to successfully decrypt a challenge that the client encrypted with the server's public key.

The reason the merchant needs to have an SSL certificate on their server is NOT for the connection to Authorize.Net. The merchant needs an SSL certificate for the handshake between the merchant's web server and the customer's web browser. This way, the connection between the customer's web browser and the merchant's web server is secure. Once again, the merchant will need a digital certificate on their server to support an SSL connection between the merchant's web server and the customer's browser.

## 3.5. Minimum System Requirements

The tag has been tested on several Windows, Linux and UNIX platforms and will run on both 32-bit and 64-bit systems. The tag requires either a Java 1.5 or 1.6 execution environment. The Windows version should function properly on all Windows platforms. We have tested the tag on ColdFusion 8 and 9.

# 4. OTHER STUFF

## 4.1. Software License

See a copy of the license in the distribution zip file. **Tags that are purchase from CFXWorks and are licensed on an annual subscription basis contain an expiration data.** The tag will stop working when the date expires.

 

## 4.2. Technical Limitations

There are no known technical limitations on the use of this tag.

## 4.3. Support

Support is provided for this offering via email. If you have questions on need assistance please send an email to support@cfxworks.com or call me at 770-441-0952.

Please read the license file “CFX_AIM_JAVA_License.doc” included in the distribution zip file.

 

# APPENDIX A. RC & RESPONSE CODES

**Note that the RC value is set by the tag and is generated internally by the tag. The X_RESPONSE_CODE value is set by Authorize.Net and is generated by them.**

| RC | X_RESPONSE_CODE | Tag | Authorize.Net |
|---|---|---|---|
| 0 | 1 | The tag worked. | The transaction was **approved** by Authorize.Net. |
| 402 | 2 | The tag worked. | The transaction was **declined** by Authorize.Net. |
| 403 | 3 | The tag worked. | There was an **error** discovered by Authorize.Net processing the transaction. |
| 404 | 4 | The tag worked. | The transaction is being **held** by Authorize.Net for review. |
| 405 | blank | An error has occurred connecting to Authorize.Net. | Most likely the transaction was never received by Authorize.Net. |
| 501 | blank | Missing X_LOGIN value | The transaction was not sent to Authorize.Net |
| 502 | blank | Missing X_TRAN_KEY value | The transaction was not sent to Authorize.Net |
| 503 | Blank | Card #failed LUHN test. | The transaction was not sent to Authorize.Net |
| 601 | Blank | License has invalid digital signature. | The transaction was not sent to Authorize.Net |
| 602 | Blank | Tag license has expired. | The transaction was not sent to Authorize.Net |
| 700 | Blank | Error writing to transaction log. | Normally, if a log error occurs, it would have occurred before attempting to send the transaction to Authorize.Net. In this case, the tag recognizes the log error and aborts sending the transaction to AN. |

**Note that transactions which return a 501–700 RC are not written to the tags log!**

# APPENDIX B. CREDIT CARD VALIDATION USING THE LUHN FORMULA

## Credit Card Type and Edit Criteria

| CARD TYPE | LOW RANGE | HIGH RANGE | LENGTH | CHECK DIGIT |
|---|---|---|---|---|
| Visa | 400000 | 499999 | 13 or 16 | Mod 10 |
| MasterCard | 500000 | 559999 | 16 | Mod 10 |
| American Express | 340000<br>370000 | 349999<br>379999 | 15 | Mod 10 |
| Diners Club | 300000<br>360000 | 305999<br>369999 | 14 | Mod 10 |
| Discover | 601100 | 601199 | 16 | Mod 10 |
| JCB | 352800 | 358999 | 16 | Mod 10 |

***Figure 4 - LUHN Formula (Mod 10) for Validation of Account Number***

The steps listed below are required for validation of the PAN (Primary Account Number).

**STEP 1:** Double the value of the alternate digits of the PAN beginning with the second digit from the right (the 1st right-hand digit is the check digit).

**STEP 2:** Add the individual digits resulting from Step One above to each of the unaffected (unused) digits of the original PAN.

**STEP 3:** If the results from Step One is a two digit number, add each digit together to obtain a one digit result for Step Two.

**STEP 4:** The total obtained in Step Two must be a number ending in Zero (10, 20, 30, etc.) for the PAN to be validated.

 