

Transact Advantage

User Guide



© 2006 Nodus Technologies, Inc. - All Rights Reserved

Table of Contents

TABLE OF CONTENTS	2
INTRODUCTION	4
PRODUCT FEATURES	5
TERMS USED IN THIS GUIDE	6
HOW IT ALL WORKS	7
OVERVIEW	8
ABOUT YOUR PAYMENT GATEWAY	10
VeriSign	10
Test TRANSACT ADVANTAGE with VeriSign:	10
Paymentech Processor	11
Test TRANSACT ADVANTAGE with Paymentech:	11
Accessing VeriSign Daily Activity Report	12
Run VeriSign Daily Activity Reports:	12
Accessing Paymentech Daily Activity Report	13
Run Paymentech Reports:	13
SUPPORTED TRANSACTIONS TYPES	14
Sale	14
Book	14
Ship	14
Credit	15
Force	15
Void	15
INSTALLATION / REGISTRATION INSTRUCTIONS	16

Installation	16
Prerequisites.....	16
Running the Installation	16
SETTING UP TRANSACT ADVANTAGE.....	18
TRANSACT ADVANTAGE Setup	18
TESTING TRANSACT ADVANTAGE.....	20
USING TRANSACT ADVANTAGE.....	21
Using Visual Basic.....	21
Using Visual C++	24
Using Perl.....	25
Using ASP	26
Using ColdFusion.....	27
Using PHP	28
CUSTOMIZING HTML TEMPLATES.....	29
APPENDIX A: TROUBLESHOOTING	31
APPENDIX B: TECHNICAL SUPPORT	32
APPENDIX C: TRANSACTION FIELDS.....	33
APPENDIX D: TRANSACTION FUNCTIONS	38
APPENDIX E: XML DEFINITIONS	39
Transaction Request XML.....	39
Transaction Response XML.....	39
Transaction Post XML.....	40

Introduction

TRANSACT ADVANTAGE is a highly customizable easy to use electronic payment solution that's designed to fulfill all your payment processing needs. Through easy-to-use interfaces and fully customizable HTML templates TRANSACT ADVANTAGE gives you the power to quickly integrate real-time electronic payment processing into any e-commerce application.

It supports all major payment processors with a wide range of Credit Card, Electronic Check and ACH payment types. Its unique collaborative framework allows users to switch payment processors without having to re-write the transaction processing logic. And with its ability to seamlessly integrate with existing business environments, it's the only end-to-end solution that covers all aspects of electronic transaction processing.

TRANSACT ADVANTAGE is available as a downloadable Software Development Kit (SDK). Built using Microsoft's COM and .NET technology it can be integrated into any web-site or application that supports COM or SOAP interfaces in no time.

Product Features

TRANSACT ADVANTAGE is an end-to-end electronic payment processing solution designed to provide both transaction management and processing support in one complete package. Features include:

- Multiple payment origination points. Electronic payments could originate for any store front, POS, B2B portals, call centers, ERP or CRM solutions.
- Support for major payment gateways
 - Authroize.Net
 - CyberSource
 - Moneris
 - Paymentech
 - VeriSign
 - Wells Fargo
- Support for multiple payment types including credit cards, electronic checks and ACH.
- Seamless integration into existing e-commerce web sites through easy-to-use interface and fully customizable HTML templates.
- HTML auto-generation support, for transaction entry and response web-pages.
- Front-to-back data transfer support through business adapters.

Terms Used in this Guide

Various terms are used throughout the credit card industry. Often several different terms may have the same meaning. This section will tell you which terms we have chosen to use and what they mean. You can also find terms not used in this guide along with the terms we are using in their place.

Terms referring to specific transaction types are not covered in this section. For example, a 'Pre-Authorize' transaction at one processor may be called a 'Book' transaction by a different processor. Transaction Types terms are covered in their own section *Transaction Types*.

Payment Gateway	The software and / or service that transmit the credit card transactions to the processor. Examples of a Payment Gateway are VeriSign and Paylinx. A Payment Gateway is also referred to as a back-engine, gateway or processing engine.
Processor	The Processor is also known as a network. The Processor is a middle-man between the merchant and the customer's bank. The Processor allows the merchant to connect to the approving bank (cardholder's bank) to check the validity of the card and the availability of the funds and sends a response back through a Payment Gateway.
Payment Gateway Connector	Payment Gateway Connectors provide all the necessary functionality to process transactions through a specific payment gateway. Every payment gateway has a specific connector that links it to the Transaction Server.
Business Adapter	Business Adapters provide integration support for connecting to the back-end business environment. They are used to post transactions submitted through the Transaction Server to any external system that needs them.
Transact Server	Transact Server (Transaction Server) is the core interface responsible for routing the transactions from different origination points to appropriate payment gateways and business adapters, and returning the captured response back to the client.
Transact Service	Transact Service is a web service that runs on top of the Transaction Server, and provides additional support for deploying TRANSACT ADVANTAGE in a client service environment, making it accessible to both windows and non-windows based application through Simple Object Access Protocol (SOAP).

How it All Works

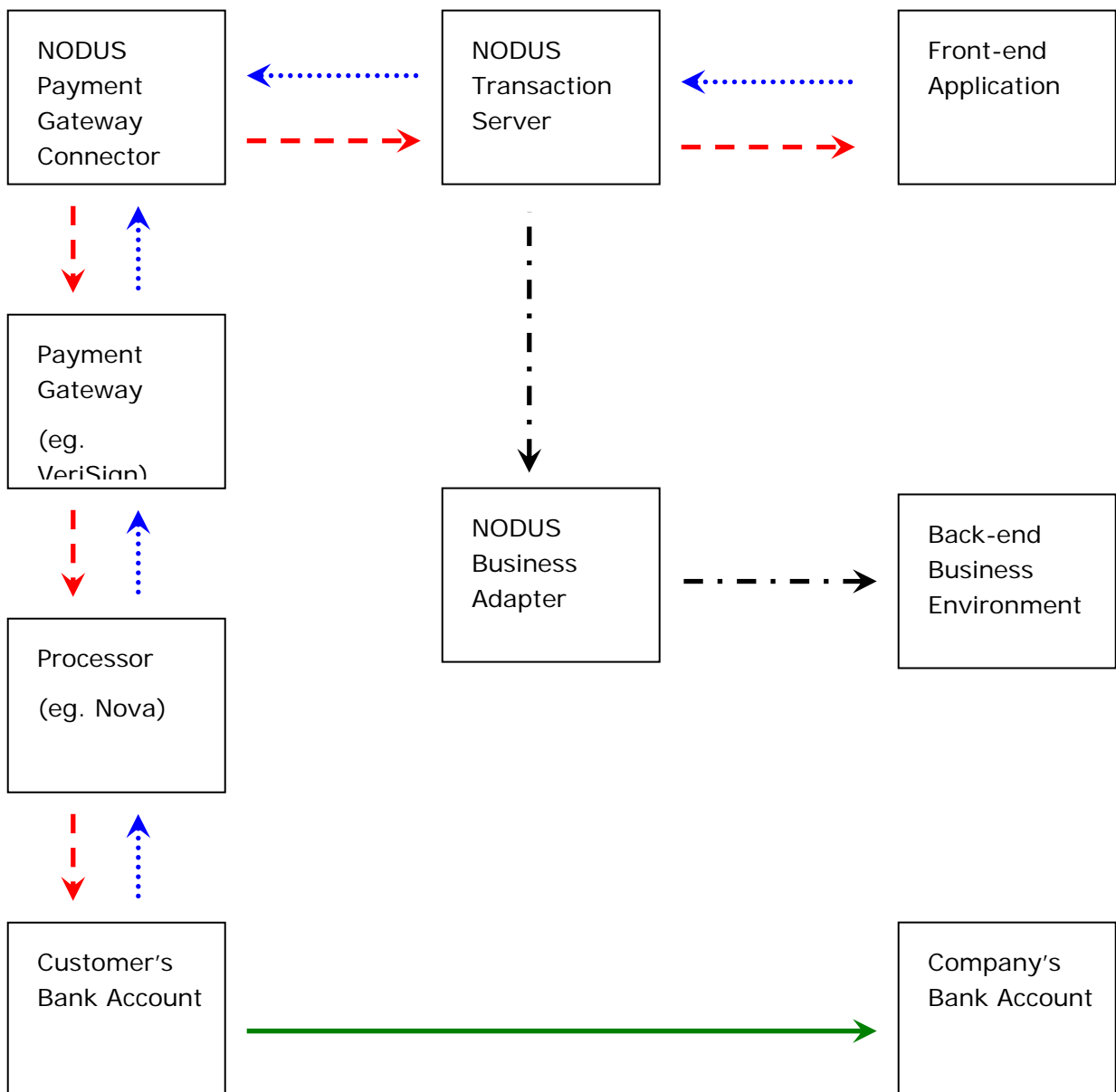
Below is a graphic representing the transaction flow when using TRANSACT ADVANTAGE.

The red dashed line represents the **Request**.

The blue dotted line represents the **Response**.

The green solid line represents **Settlement**.

The black dashed/dotted line represents **Approved Transaction**.

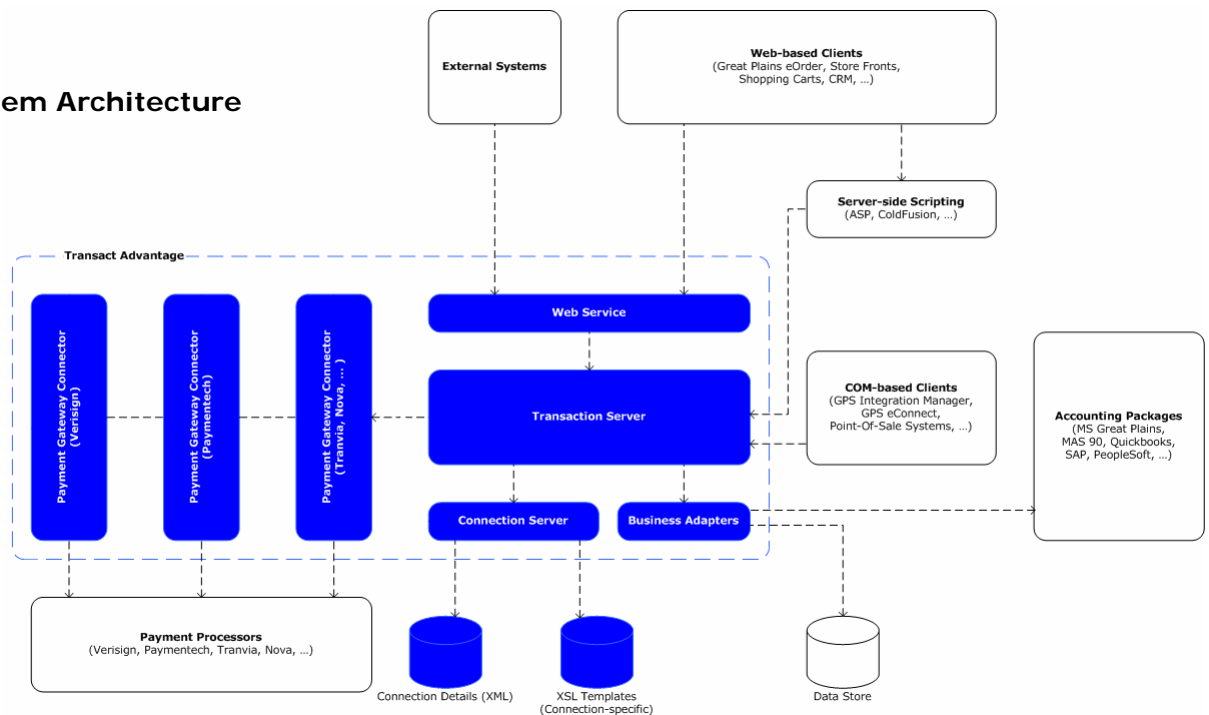


Overview

TRANSACT ADVANTAGE is designed using a collaborative framework that provides an open architecture for supporting multiple payment gateways and business environments which could be added or replaced as needed.

At the core of TRANSACT ADVANTAGE is the Transaction Server. Responsible for getting payments from any web-site or e-commerce application, processing them through the appropriate payment gateway, and posting them to the associated business environment, Transaction Server act as a powerful backbone for our current and future products aimed at providing a complete solution that will address all your electronic payment processing needs.

System Architecture



TRANSACT ADVANTAGE comprises of following components:

Payment Gateway Connector

Payment Gateway Connectors provide all the necessary functionality to process transactions through a specific payment gateway. Every payment gateway has a specific connector that links it to the Transaction Server.

Business Adapter

Business Adapters provide integration support for connecting to the back-end business environment. They are used to post transactions submitted through the Transaction Server to any external system that needs them.

Transact Server

Transact Server (Transaction Server) is the core interface responsible for routing the transactions from different origination points to appropriate payment gateways and business adapters, and returning the captured response back to the client.

Transact Service

Transact Service is a web service that runs on top of the Transaction Server, and provides additional support for deploying TRANSACT ADVANTAGE in a client service environment, making it accessible to both windows and non-windows based application through Simple Object Access Protocol (SOAP).

About Your Payment Gateway

VeriSign

VeriSign works with a number of different processors. If using VeriSign you have your choice of any of the processors VeriSign supports. If you have not already set up an account with VeriSign, you will be given an application when you purchase TRANSACT ADVANTAGE. The VeriSign application asks for information about your company, your bank and the processor you have chosen. When entering setup information in TRANSACT ADVANTAGE the following information will be helpful.

Processing Server Name – Test mode:	test-payflow.verisign.com
Processing Server Name – Live mode:	payflow.verisign.com
Port:	443
VeriSign Partner ID:	<p>If you signed up through Nodus – “nodus”</p> <p>If you signed up directly with VeriSign – “verisign”</p> <p>If you signed up through a different VeriSign partner, check with that partner for the Partner ID.</p>
Website	https://manager.verisign.com

Test TRANSACT ADVANTAGE with VeriSign:

1. Install TRANSACT ADVANTAGE according to the instructions in this guide.
2. Setup TRANSACT ADVANTAGE according to the instructions in this guide. Use “test-payflow.verisign.com” as the IP address. Use User ID, Vendor ID = “Nodus1”, Password = “nodus!”, and Partner ID = “Nodus”.
3. Start sample application from the following location,


```
<TRANSACT ADVANTAGE - Install Directory>
  \Examples\VB\TransactAdvantageTestApp.exe
```
4. Specify connection name and select transaction type. Use this test account number: Visa – 4111-1111-1111-1111, for the credit card. Use any future expiration date. Depending on transaction type certain fields will be grayed out. Transactions for \$999.99 or less should be approved. Transactions for \$1000.00 or more should be denied. The expiration date must be in the format of MMY.

Paymentech Processor

If you plan to use Paymentech processor, you will be given an application when you purchase TRANSACT ADVANTAGE. The Paymentech application asks for information about your company, and your bank. You need to sign a Paymentech account through Nodus in order to use TRANSACT ADVANTAGE with Paymentech processor. When entering setup information in TRANSACT ADVANTAGE the following information will be helpful.

Processing Server Name–Test mode:	epayhipvar.paymentech.net
Processing Server Name – Live mode:	epayhip.verisign.com
Port:	443
Website	https://www.paymentech.net/manager

Test TRANSACT ADVANTAGE with Paymentech:

1. Install TRANSACT ADVANTAGE according to the instructions in this guide.
2. Setup TRANSACT ADVANTAGE according to the instructions in this guide. Use "epayhipvar.paymentech.net" as the IP address. Use Merchant ID = "700000000403", BIN = "000002", and Terminal ID = "001".
3. Start sample application from the following location,
<TRANSACT ADVANTAGE - Install Directory>
\Examples\VB\TransactAdvantageTestApp.exe
4. Specify connection name and select transaction type. Use this test account number: Visa – 4111-1111-1111-1111, for the credit card. Use any future expiration date. Depending on transaction type certain fields will be grayed out*. Transactions for \$999.99 or less should be approved. Transactions for \$1000.00 or more should be denied. The expiration date must be in the format of MMY.

*The invoice number is required for most of the transaction types when using Paymentech.

Accessing VeriSign Daily Activity Report

The steps to run the VeriSign Daily Activity reports follow.

Run VeriSign Daily Activity Reports:

1. Go to the VeriSign website <<https://manager.verisign.com>>.
2. Enter Partner ID. For Logon enter your Merchant ID.
3. Choose Reports.
4. Select Daily Activity Reports
5. From drop-down, choose to view live transactions.
6. Choose Submit.

Transaction Type Names

Following is a list of the transaction type names used by VeriSign and the equivalent terms in TRANSACT ADVANTAGE.

VeriSign	TRANSACT ADVANTAGE
Authorize	Book
Delayed Capture	Ship
Sale	Sale
Credit	Credit
Void	Void
Voice Authorization	Force

Accessing Paymentech Daily Activity Report

The steps to run the Paymentech Daily Activity reports follow.

Run Paymentech Reports:

1. Goto the Paymentech website
<<https://www.paymentech.net/manager>>.
2. Enter Group ID (mid), User ID and Password.
3. Choose Review.
4. Select Batch Search or Transaction Search.
5. Select Date range or Batch # or Document #.
6. Choose Search.

Supported Transactions Types

This section explains each of the transaction types in TRANSACT ADVANTAGE. The steps for processing each of the transaction types are nearly identical. Slight variations are identified and explained in this section. Please note: the various names of the transaction types vary by processor. A *Sale* may be called *Capture* by some processors. A *Book* may be a *Pre-Authorize*. There are other variations. The names chosen for TRANSACT ADVANTAGE are among the more common but are not universal. Though the names may vary, the processing details do not.

Sale

An approved *Sale* is an immediate charge to the customer's credit card or account. A *Sale* can only be reversed with a *Void* or a *Credit*. Some payment gateways do not allow you to void a *Sale* transaction.

Book

A *Book* is a reserve of a specified amount on the customer's credit card or account. A *Book* prevents the customer from using that portion of their credit / funds, but does not actually charge the card nor transfer any funds. A *Book* is useful for companies that ship merchandise one or more days after receiving an order. By issuing a *Book*, a company reserves the necessary amount on the customer's card at the time of the order. This reserve assures an approved *Ship* transaction at the time the merchandise is eventually shipped. A *Ship* transaction is necessary to complete the *Book*.

The number of days a *Book* will stay open is determined by each cardholder's issuing bank. The most common number is 7 to 10 days, but some banks may hold *Books* for as little as three days and as long as four weeks.

Ship

A *Ship* can only be issued for a transaction that was previously a *Book*. Under ordinary circumstances, a *Ship* is assured approval as long as the amount is equal to or less than the original *Book* amount and the *Ship* is sent before the *Book* has expired. A *Ship* results in an immediate charge to the customer's credit card or account. If the *Ship* is for less than the original *Book* amount, the remainder of the original *Book* amount is released back to the customer's credit line or account.

Credit

A *Credit* is issued to transfer money from the company's account to the customer's account or credit card. When a Return is processed a *Credit* can be issued to return money to the customer.

Force

A *Force* is used to enter already approved transactions. A *Force* is typically used for a transaction processed through a phone authorization. When entering a *Force* you will be required to enter the authorization code.

Void

A Void is issued for an unsettled approved transaction. When a Void is successfully issued, neither the Void nor the original transaction will appear on the customer's statement. A Void can only be issued against an unsettled transaction. When a Void is sent, if the original transaction has already been settled, the Void will be denied and a warning will be returned. A settled Sale transaction must be reversed with a Credit.

NOTE:

Paymentech processor does not allow Ship transactions with ship amounts greater than the Book amount.

Installation / Registration Instructions

There are several steps to install TRANSACT ADVANTAGE. Each section below covers one of the steps. This chapter contains the following sections:

- *Installation*
- *Registration*

Installation

This section explains how to install the TRANSACT ADVANTAGE software development kit (SDK). It also identifies any prerequisite software for the desired installation platform.

Prerequisites

Before installing TRANSACT ADVANTAGE, verify that your system has the following minimum requirements:

- Windows 2000 Service Pack 2
- Windows 2000 Server and Advanced Server with Service Pack 2
- Windows NT 4.0 Service Pack 6
- Windows 95 Service Release 2
- Windows XP Professional
- Windows XP Home

Running the Installation

You will need administrative privileges on your workstation in order to run the TRANSACTADVANTAGE*.exe which is an InstallShield program. The InstallShield routines will do the following:

- Update the InstallShield if necessary.
- Install the User Guide and Release Notes
- Install various .dlls and system files
- Install Transaction Server to your local drive
- Install the Nodus Framework to your local drive

Execute the TRANSACTADVANTAGE*.exe:

1. Double-click the TRANSACTADVANTAGE*.exe file.
2. Follow the instructions on-screen.
3. To complete the installation, reboot the computer when prompted.

Setting Up TRANSACT ADVANTAGE

TRANSACT ADVANTAGE Setup

TRANSACT ADVANTAGE is designed to support multiple processors and Payment Gateways. The Setup ID entered in the Setup window will designate the appropriate Payment Gateway, along with necessary connection information.

Transact Advantage Setup

Save Delete Clear

Setup ID: TransactAdvantageTest

Connector

Connector: Verisign

Processor: American Express 800-297-5555

Card Class: Credit

Parameter	Value
Server.Address	test-payflow.verisign.com
Server.Port	443
Server.ProxyAddress	
Server.ProxyPort	

Installed Adapters

Adapter: Great Plains Adapter Setup Activate

Parameter	Value
Provider	SQLOLEDB.1
Password	****
Persist Security Info	True

Connection String: Provider=SQLOLEDB.1;Password=****;Persist Security

Create a Setup ID using Transact Advantage Setup Window:

1. Open the Transact Advantage Setup: Start Menu ⇒ Programs ⇒ Nodus Technologies ⇒ Connection Manager Setup.
2. Enter in a Setup ID name. Do not use any special characters.
3. Select Payment Gateway Connector, Processor and Card Class.

4. Enter the Payment Gateway's connection details.
5. Select Great Plains Adapter and mark the Activate checkbox.
6. Enter in your connection information for connecting to the database.
7. Choose Save. Close the window by clicking "X" on top right corner.

Testing TRANSACT ADVANTAGE

Once you have created the connections using the TRANSACT ADVANTAGE Setup you can start using the SDK. There is no restriction on the number of connections. However it is recommended that you use different connections for testing & development and different ones for going live.

You can use the sample program to test your installation. If you have Internet Information Server installed on the computer you can use the ASP Demo site to test the SDK. Just create a virtual directory on the web server and make it point to following path:

```
<TRANSACT ADVANTAGE install directory>\Examples\ASP
```

and use this URL to run the demo

```
http://localhost/<test-site>/default.asp
```

If the COM SDK is not installed on the web server use the sample VB application (TransactAdvantageTestApp.exe) to test TRANSACT ADVANTAGE. It is installed under the following location:

```
<TRANSACT ADVANTAGE install directory>\  
Examples\VB\Application
```

You can also use the vbscript file to see a sample of the code that you would write to work with TRANSACT ADVANTAGE. There are six script files you can run: FullCode.vbs, SampleCode.vbs, Test1.vbs, Test2.vbs, Test3.vbs, and Test4.vbs. They are installed under the following location:

```
<TRANSACT ADVANTAGE install directory>\  
Examples\VB\Scripts
```

Using TRANSACT ADVANTAGE

TRANSACT ADVANTAGE can be used from any development environment that supports COM. Following are some examples on how to integrate TRANSACT ADVANTAGE into your application.

Using Visual Basic

Steps

- Create a Standard EXE application in Visual Basic.
- Set a reference to Nodus Transact Server by going to Project ⇒ References.
- Create control arrays of text box (name: txtTrxField) and label (name: lblTrxField) for transaction entry fields.
- Create text box (name: txtResponse) for transaction response fields.
- Create a command button (name: cmdProcess) for processing transaction
- Add the following code to your main form

```
Private mobjTrx As Transaction

' Creates a new transaction object and displays transaction
' entry fields

Public Sub InitializeTransaction()
    Dim i As Integer
    Dim objField As Field
    Dim objConn As Connection

    ' Instantiate Connection object
    Set objConn = New Connection

    ' Open Connection
    objConn.Connect "<Setup ID>"

    ' Open Transaction
    Set mobjTrx = objConn.OpenTransaction( _
        objConn.TransactionType_Book, False)

    ' You can iterate through all the transaction entry fields
    ' and add them to the form dynamically

    i = 1
    For Each objField in mobjTrx.RequestFields

        ' Create label for transaction entry field
        Load lblTrxFIELD(i)
        lblTrxFIELD(i).Caption = objField.Description

        ' Create text box for entering field value
        Load txtTrxFIELD(i)
        txtTrxFIELD(i).Tag = objField.Name

        i = i + 1
    Next

End Sub

' Processes the transaction

Public Sub ProcesTransaction()
    Dim i as Integer
    Dim objField As Field

    For i = 1 to txtTrxFIELD.Count

        ' Set field values
        mobjTrx.SetField txtTrxFIELD.Tag, txtTrxFIELD.Text

    Next i

    ' Process transaction
    mobjTrx.Process

End Sub
```

```
' Displays response data returned by the processor

Public Sub ShowResponse()
    Dim objField as Field

    For Each objField in mobjTrx.StatusFields

        ' Add transaction response fields in a text box
        txtResponse.Text = txtResponse.Text & _
            objField.Name & ": " & _
            objField.Description & _
            Chr(13) & Chr(10)

    Next

End Sub

' Loads the form

Private Sub Form_Load()

    InitializeTransaction ' initalizes transaction object

End Sub

' Click event for Porcess command button

Private Sub cmdProcess_Click()

    ProcessTransaction    ' processes transaction

    ShowResponse          ' displays transaction results

End Sub
```

Using Visual C++

```
#import "<TRANSACT ADVANTAGE install directory>\TransactServer.dll"
                                         no_namespace named_guids

// Creates a new transaction object set the transaction
// entry fields and processes transaction.

bool ProcessTransaction()
{
    // Instantiate Connection object
    _ConnectionPtr objConn(CLSID_Connection);

    // Open Connection
    objConn->Connect("<Setup ID>");

    // Open Transaction
    _TransactionPtr objTrx = objConn->OpenTransaction(
        objConn->TransactionType_Book, VARIANT_FALSE);

    // Set Credit Card Type
    objTrx->SetField("CCType", "Visa");

    // Set Credit Card Number
    objTrx->SetField("CCNumber", "4111111111111111");

    // Combine CC Expiry Month & Year
    objTrx->SetField("CCExpDate", "0205");

    // Set Transaction Amount
    objTrx->SetField("TrxAmount", "10.00");

    // Set the Invoice Number
    objTrx->SetField("InvoiceNumber", "0001");

    // Process transaction
    if (objTrx->Process() == objTrx->Status_Approved)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

Using Perl

```
use OLE;

# Creates a new transaction object set the transaction
# entry fields and processes transaction.

sub ProcessTransaction()
{
    # Instantiate Connection object
    $objConn = CreateObject OLE 'TransactServer.Connection';

    # Open Connection
    $objConn->Connect('<Setup ID>');

    # Open Transaction
    $objTrx = $objConn->OpenTransaction(
        $objConn->{TransactionType_Book}, false);

    # Set Credit Card Type
    $objTrx->SetField('CCType', 'Visa');

    # Set Credit Card Number
    $objTrx->SetField('CCNumber', '4111111111111111');

    # Combine CC Expiry Month & Year
    $objTrx->SetField('CCExpDate', '0205');

    # Set Transaction Amount
    $objTrx->SetField('TrxAmount', '10.00');

    # Set the Invoice Number
    $objTrx->SetField('InvoiceNumber', '0001');

    # Process transaction
    if ($objTrx->Process() == $objTrx->{Status_Approved})
    {
        print "transaction approved";
    }
    else
    {
        print "transaction declined";
    }
}
```

Using ASP

Transaction Entry

```
<%
' Instantiate Connection object
set objConn = Server.CreateObject("TransactServer.Connection")

' Open Connection
objConn.Connect "<Setup ID>"

' Open Transaction
set objTrx = objConn.OpenTransaction( _
    objConn.TransactionType_Book, false)

' Insert HTML for transaction entry fields
Response.Write(objTrx.RequestDataHTML)

' Save Transaction object
set Session("objTrx") = objTrx
%>
```

Transaction Response

```
<%
' Retrieve Transaction object
set objTrx = Session("objTrx")

if not (objTrx is nothing) then

' Set Credit Card Type
objTrx.SetField "CCType", Request("CCType")

' Set Credit Card Number
objTrx.SetField "CCNumber", Request("CCNumber")

' Combine CC Expiry Month & Year
objTrx.SetField "CCExpDate", Request("CCExpMonth") & _
    Right(Request("CCExpYear"), 2)

' Set Transaction Amount
objTrx.SetField "TrxAmount", Request("TrxAmount")

' Set the Invoice Number
objTrx.SetField "InvoiceNumber", Request("InvoiceNumber")

' Alternatively you can pass the ASP Request to
' Transaction object and it will retrieve the transaction
' entry fields automatically, provided the input field
' names on the form match the field names expected by the
' Transaction object.
'
' objTrx.SetFields Request

' Process Transaction
objTrx.Process

' Show Processor Response
Response.Write(objTrx.StatusDataHTML)

set Session("objTrx") = nothing

end if
%>
```

Using ColdFusion

Transaction Entry

```

<!--- Instantiate Connection object --->
<cfobject
    type = "COM"
    action = "Create"
    name = "objConn"
    class = "TransactServer.Connection">

<!--- Open Connection --->
<cfset objConn.Connect("<Setup ID>")>

<!--- Open Transaction --->
<cfset objTrx=objConn.OpenTransaction(
    objConn.TransactionType_Book, false)>

<!--- Insert HTML for transaction entry fields --->
<cfoutput>
    #objTrx.RequestDataHTML#
</cfoutput>

<!---
Before you can use session variables make sure session management
is enabled
<cfapplication name = "<Application Name>" sessionmanagement = "Yes"
    sessiontimeout = "#CreateTimeSpan(0, 0, 20, 0)#">
--->

<!--- Save Transaction object --->
<cfset session.objTrx = objTrx>

```

Transaction Response

```

<cfif IsDefined("session.objTrx")>

    <!--- Retrieve Transaction object --->
    <cfset objTrx = session.objTrx>

    <!--- Set Credit Card Type --->
    <cfset objTrx.SetField("CCType", Form.CCType)>

    <!--- Set Credit Card Number --->
    <cfset objTrx.SetField("CCNumber", Form.CCNumber)>

    <!--- Combine CC Expiry Month & Year --->
    <cfset objTrx.SetField("CCExpDate", Form.CCExpMonth &
        Right(Form.CCExpYear, 2))>

    <!--- Set Transaction Amount --->
    <cfset objTrx.SetField("TrxAmount", Form.TrxAmount)>

    <!--- Set the Invoice Number --->
    <cfset objTrx.SetField("InvoiceNumber", Form.InvoiceNumber)>

    <!--- Process Transaction --->
    <cfset objTrx.Process()>

    <!--- Show Processor Response --->
    <cfoutput>
        #objTrx.StatusDataHTML#
    </cfoutput>

    <cfset StructDelete(session, "objTrx")>

</cfif>

```

Using PHP

Transaction Entry

```
<?
# Instantiate Connection object
$objConn = new COM("TransactServer.Connection");

# Open Connection
$objConn->Connect("<Setup ID>");

# Open Transaction
$objTrx = $objConn->OpenTransaction(
    $objConn->TransactionType_Book, false);

# Insert HTML for transaction entry fields
echo $objTrx->RequestDataHTML;

# Save Transaction object
$_SESSION['objTrx'] = $objTrx
?>
```

Transaction Response

```
<?
if (isset($_SESSION['objTrx']))
{
# Retrieve Transaction object
$objTrx = $_SESSION['objTrx'];

# Set Credit Card Type
$objTrx->SetField("CCType", $CCType);

# Set Credit Card Number
$objTrx->SetField("CCNumber", $CCNumber);

# Combine CC Expiry Month & Year
$objTrx->SetField("CCExpDate",
    $CCExpMonth .
    substr($CCExpYear, strlen($CCExpYear) - 2 - 1, 2));

# Set Transaction Amount
$objTrx->SetField("TrxAmount", $TrxAmount);

# Set the Invoice Number
$objTrx->SetField("InvoiceNumber", $InvoiceNumber);

# Process Transaction
$objTrx->Process();

# Show Processor Response
echo $objTrx->StatusDataHTML;

$_SESSION['objTrx'] = null;
}
?>
```

Customizing HTML Templates

To help users customize their ecommerce web sites and to guarantee seamless integration, TRANSACT ADVANTAGE comes equipped with HTML auto-generation support for transaction entry and response web-pages. The templates are in the form of standard XSL style sheets editable by any text editor.

Whenever you create a new Setup ID using TRANSACT ADVANTAGE Setup screen a new folder is created under:

```
<TRANSACT ADVANTAGE install directory>\
  ConnectionManager\Connections\<Setup ID>
```

This folder contains all the XSL style sheets used by the TRANSACT ADVANTAGE to auto-generate HTML for transaction entry and response fields. The following table summarizes the different templates, their function and their usage.

XSL Style Sheet	Description
<p>RequestData.xsl</p> <p>Usage: Transaction.RequestDataHTML</p> <p>XML Definition: Transaction Request XML</p>	<p>Used by RequestDataHTML method to generate HTML for transaction entry fields.</p> <p>Each payment gateway has a set of required and optional parameters that need to be set before a transaction is processed.</p> <p>The fields returned in the XML contain all the parameters specified by the payment gateway regardless of whether they are optional or required.</p>
<p>ResponseData.xsl</p> <p>Usage: Transaction.ResponseDataHTML</p> <p>XML Definition: Transaction Response XML</p>	<p>Used by ResponseDataHTML method to generate HTML for transaction response fields returned by the processor.</p> <p>Only applicable after the transaction has been processed. The XML would be empty if the transaction failed due to communication failure or missing fields etc.</p>

<p>FailureData.xsl</p> <p>Usage: Transaction.FailureDataHTML</p> <p>XML Definition: Transaction Response XML</p>	<p>Used by FailureDataHTML method to generate HTML for any failure modes that might have stopped the transaction from being submitted to the processor.</p> <p>Only applicable after the transaction has been processed, the XML would be empty if the transaction was approved or denied, communication failure or missing fields etc.</p>
<p>NeededData.xsl</p> <p>Usage: Transaction.NeededDataHTML</p> <p>XML Definition: Transaction Response XML</p>	<p>Used by NeededDataHTML method to generate HTML for all required fields that were either not set or left blank before transaction was processed.</p> <p>Only applicable after the transaction has been processed and the transaction status is set to <Status_MoreInfo>.</p>
<p>StatusData.xsl</p> <p>Usage: Transaction.StatusDataHTML</p> <p>XML Definition: Transaction Response XML</p>	<p>Used by StatusDataHTML method to generate HTML for all failure, response and needed fields.</p> <p>Only applicable after the transaction has been processed. Use this method when source of transaction failure is not important.</p>

Appendix A: Troubleshooting

Problem	Remedies
'Transaction Failure'	Be sure Setup is correct. Be sure Internet connection is live. Be sure operating system is Windows 2000, NT or XP. Be sure Partner ID entered during installation is correct. See <i>About Your Payment Gateway</i> .

Appendix B: Technical Support

If you have any questions about this program, you should first consult this User's Guide. If you are still having trouble, there are a couple of support options.

Contact the Nodus / Partner from whom this software was purchased.

Nodus provides support on a cost-per-incident basis. Pre-paid support is also available. To find out more, contact Nodus directly:

Through	Address/Numbers
e-mail	support@nodustech.com
Web Site	http://www.nodustech.com
Telephone	(909) 482-4701
Fax	(909) 482-4705
Mail	250 West First Street Suite 302 Claremont, California 91711

Appendix C: Transaction Fields

You can set the fields by using the SetField method of the Transaction class. An example of calling this method is:

```
objTrx.SetField("CCNumber", "4111111111111111")
```

The following table contains the fields that can be submitted to the payment gateway.

Field ID	Field Name	Field Description
TrxField_D1	CCNumber	Credit Card Number
TrxField_D2	ECCNumber	Encrypted Credit Card Number
TrxField_D3	CCExpDate	Expiration Date MMY
TrxField_D4	CCExpDateYYMM	Expiration Date YYMM
TrxField_D5	FirstName	Card Holder First Name
TrxField_D6	MiddleName	Card Holder Middle Name
TrxField_D7	LastName	Card Holder Last Name
TrxField_D8	Address1	Card Holder Address 1
TrxField_D9	Address2	Card Holder Address 2
TrxField_D10	Address3	Card Holder Address 3
TrxField_D11	City	Card Holder City
TrxField_D12	State	Card Holder State
TrxField_D13	Zip	Card Holder Zip
TrxField_D14	TrxType	Transaction Type
TrxField_D15	TrxAmount	Transaction Amount
TrxField_D16	TrxID	Origination ID
TrxField_D17	TrxResultCode	Result Code
TrxField_D18	CCType	Card Type
TrxField_D19	PaymentType	Card Class
TrxField_D20	MerchantID	Merchant ID
TrxField_D21	MerchantRefID	Merchant Referral ID
TrxField_D22	UserID	User ID
TrxField_D23	Password	User Password
TrxField_D24	AuthCode	Authorization Code
TrxField_D25	Comment1	User Comment 1
TrxField_D26	Comment2	User Comment 2
TrxField_D27	Comment3	User Comment 3

TrxFld_D28	Comment4	User Comment 4
TrxFld_D29	Comment5	User Comment 5
TrxFld_D30	MICR	MICR Line
TrxFld_D31	ResponseMsg	Processor Response Message
TrxFld_D32	AVSAddResponse	AVS Address Response
TrxFld_D33	AVSZipResponse	AVS Zip Response
TrxFld_D34	PONumber	PO Number
TrxFld_D35	Desc	General Description
TrxFld_D36	Desc1	General Description 1
TrxFld_D37	Desc2	General Description 2
TrxFld_D38	Desc3	General Description 3
TrxFld_D39	Desc4	General Description 4
TrxFld_D40	InvoiceNumber	Invoice Number
TrxFld_D41	ShipToZip	Ship to Zip
TrxFld_D42	TaxAmount	Tax Amount
TrxFld_D43	CommCardType	Commercial Card Type
TrxFld_D44	DutyAmount	Duty Amount
TrxFld_D45	FreightAmount	Freight Amount
TrxFld_D46	TaxExempt	Tax Exempt
TrxFld_D47	CountryCode	Country Code
TrxFld_D48	CustomerCode	Customer Code
TrxFld_D49	CVV2	CVV2
TrxFld_D50	DiscountAmount	Discount Amount on Total Sale
TrxFld_D51	OrderDate	Order Data MMDDYY
TrxFld_D52	ShipFromZip	Ship From Zip
TrxFld_D53	CheckNumber	Check Number
TrxFld_D54	AccountName	Account Holder Name
TrxFld_D55	AccountStreet	Account Holder Street
TrxFld_D56	DriverLicense	Driver's License (State + Number)
TrxFld_D57	Email	Email
TrxFld_D58	SSN	Social Security Number
TrxFld_D59	RoutingNumber	ABA Routing Number
TrxFld_D60	AccountNumber	Account Number
TrxFld_D61	AccountType	Account Type

TrxFld_D62	ItemQuantity	Line Item Quantity
TrxFld_D63	ItemUPC	Line Item UPC
TrxFld_D64	ItemDesc	Line Item Description
TrxFld_D65	ItemUOM	Line Item UOM
TrxFld_D66	ItemCost	Line Item Cost
TrxFld_D67	ItemProdCode	Line Item Product Code
TrxFld_D68	ItemDiscount	Line Item Discount
TrxFld_D69	ItemAmount	Line Item Amount
TrxFld_D60	ItemTaxAmount	Line Item Tax Amount
TrxFld_D71	Prenote	Prenote
TrxFld_D72	ECOrderNumber	EC Order Number
TrxFld_D73	TimeZone	Time Zone
TrxFld_D74	CurrencyCode	Currency Code
TrxFld_D75	CurrencyExponent	Currency Exponent
TrxFld_D76	TrxDate	Transaction Date
TrxFld_D77	ShipRef	Shipping Reference
TrxFld_D78	Phone	Phone Number
TrxFld_D79	MsgType	Message Type
TrxFld_D80	TrxRefIndex	Transaction Reference Index
TrxFld_D81	BIN	BIN
TrxFld_D82	ResponseCode	Response Code
TrxFld_D83	CVV2ResponseCode	CVV2 Response Code

The following table is a list of the fields that are used to submit the transaction to the Nodus tables.

Field Name	Field Description
MSO_Doc_Number	Document number of the transaction
MSO_Doc_Type	The type of document. (1) Quote, (2) Order, (3) Invoice, (4) Back Order, (5) Return
MSO_Source_Of_Document	(1) Sales Order,
BACHNUMB	The name of the batch in Great Plains.
BCHSOURC	The source of the batch
MSO_TrxType	The transaction type
CUSTNMBR	The customer number

MSO_FirstName	Customers first name
MSO_MiddleName	Customers middle name
MSO_LastName	Customers last name
MSO_CardExpDate	The expiration date of the credit card.
MSO_CardName	The card name. If integrated with Great Plains, this needs to be a card setup in Great Plains.
MSO_IsCardValid	Set to "1" if the card was successfully processed.
CRCRDAMT	Set this to the Transaction amount
MSO_Auth_Amount	Set this to the Transaction amount
ZIPCODE	The zip code
CCode	The country code
PHONNAME	The phone number
MSO_ShipFromZip	The ship from zip code
MSO_TaxAmount	The tax amount
MSO_FreightAmount	The freight amount
MSO_MiscAmount	The miscellaneous amount
MSO_DESC	The general description of the transactions
MSO_DESC1	Additional description field
MSO_DESC2	Additional description field
MSO_DESC3	Additional description field
MSO_DESC4	Additional description field
MSO_COMMENT1	Comment field
MSO_COMMENT2	Comment field
MSO_COMMENT3	Comment field
MSO_COMMENT4	Comment field
MSO_TrxStatus	The transaction status
MSO_IsBatched	Flag for batching transactions
MSO_IsSettled	Flag for settled transactions
MSO_Number_Times_Card_De	Number of time the card was denied
MSO_Denied_Edited	
USERID	User Id from Accounting framework
USERDATE	The date the transaction was posted

MSO_Last_Xmit_Date	The date the transaction was submitted
MSO_Last_Xmit_Time	The time the transaction was submitted
MSO_Last_Settled_Date	The date the transaction was settled
MSO_Last_Settled_Time	The time the transaction was settled
MSO_IsVoid	If the transaction was voided
MSO_Source_Of_Orig	The source of the original document

The following table are flags available in Transact Advantage.

Field Name	Field Description
FailOnAddressMismatch	If this field is set to "1" then the transaction will be denied if the Address that is submitted does not match what is on the credit card.
FailOnZipMismatch	If this field is set to "1" then the transaction will be denied if the zip code that is submitted does not match what is on the credit card.
SaveCreditCard	This will save the credit card information to the MS273527 table.

Appendix D: Transaction Functions

The Transaction class has three basic functions: Process, Post, booleanPost.

The Process function will submit your transaction to the payment gateway for verification. When transaction is returned the information is available in the ResponseFields. You can check the status of the transaction by comparing it to: Status_Approved, Status_Denied, Status_AVSFailure, Status_None, Status_Failure, and Status_MoreInfo properties.

The Post function will simply submit a transaction to the database specified in the connection string.

The booleanPost will submit a transaction to the database but will also return a true or false depending on if the transaction was able to post. True if the post was successful, false otherwise. If the value returned is false then you can check the error message by calling the ErrorString property of the transaction object.

Appendix E: XML Definitions

Transaction Request XML

```
<Schema name="trx_request_schema"
  xmlns="urn:schemas-microsoft-com:xml-data">
  <ElementType name="Transaction" content="textOnly" model="closed" />
  <ElementType name="Type" content="textOnly" model="closed" />
  <ElementType name="Status" content="textOnly" model="closed" />
  <ElementType name="Fields" content="eltOnly" model="closed">
    <ElementType name="Field" content="eltOnly" model="closed">
      <AttributeType name="id" />
      <ElementType name="Name" content="textOnly" model="closed" />
      <ElementType name="Desc" content="textOnly" model="closed" />
      <ElementType name="Required" content="textOnly" model="closed" />
      <ElementType name="Encrypted" content="textOnly" model="closed" />
      <ElementType name="Type" content="textOnly" model="closed" />
      <attribute type="id" />
      <element type="Name" />
      <element type="Desc" />
      <element type="Required" />
      <element type="Encrypted" />
      <element type="Type" />
    </ElementType>
    <element type="Field" minOccurs="1" maxOccurs="*" />
  </ElementType>
</Schema>
```

Transaction Response XML

```
<Schema name="trx_response_schema"
  xmlns="urn:schemas-microsoft-com:xml-data">
  <ElementType name="Transaction" content="textOnly" model="closed" />
  <ElementType name="Type" content="textOnly" model="closed" />
  <ElementType name="Status" content="textOnly" model="closed" />
  <ElementType name="Fields" content="eltOnly" model="closed">
    <ElementType name="Field" content="eltOnly" model="closed">
      <AttributeType name="id" />
      <ElementType name="Name" content="textOnly" model="closed" />
      <ElementType name="Desc" content="textOnly" model="closed" />
      <attribute type="id" />
      <element type="Name" />
      <element type="Desc" />
    </ElementType>
    <element type="Field" minOccurs="1" maxOccurs="*" />
  </ElementType>
</Schema>
```

Transaction Post XML

```
<Schema name="trx_post_schema"
  xmlns="urn:schemas-microsoft-com:xml-data">
  <ElementType name="Transaction" content="textOnly" model="closed" />
  <ElementType name="Type" content="textOnly" model="closed" />
  <ElementType name="Status" content="textOnly" model="closed" />
  <ElementType name="Fields" content="eltOnly" model="closed">
    <ElementType name="Field" content="eltOnly" model="closed">
      <AttributeType name="id" />
      <ElementType name="Name" content="textOnly" model="closed" />
      <ElementType name="Desc" content="textOnly" model="closed" />
      <ElementType name="Required" content="textOnly" model="closed" />
      <ElementType name="Encrypted" content="textOnly" model="closed" />
      <ElementType name="Type" content="textOnly" model="closed" />
      <ElementType name="Value" content="textOnly" model="closed" />
      <attribute type="id" />
      <element type="Name" />
      <element type="Desc" />
      <element type="Required" />
      <element type="Encrypted" />
      <element type="Type" />
      <element type="Value" />
    </ElementType>
    <element type="Field" minOccurs="1" maxOccurs="*" />
  </ElementType>
</Schema>
```