Introduction

Application Settings are a facility within the Sitecore Ecommerce Enterprise Edition application that is used to store system-wide and site-specific settings for the behavior of the application. While there are over 200 of these settings, they tend to group together in some specific categories (i.e. Integration, Order Behavior, Email) and not all settings are used on all sites. Additionally, the power of the application setting module enables Sitecore Ecommerce Enterprise Edition developers to leverage this power for their own use for custom integrations or functionality to be used within the site.

The entries below will describe the general intent and usage of the various settings that are currently standard within the system. Remember than not all settings are used for any given web site – especially the integration ones which tend to be ERP-specific.

Valid values are assumed to be strings unless otherwise specified.

Default values are only shown when they have some particular use.

Credit Card

Credit Cards are managed through the Payment Gateway selection which is a combination of establishing the appropriate gateway using the ISCModule_PaymentGateway setting and the applicable settings by gateway. A gateway is also referred to as a credit card processor Sitecore Ecommerce Enterprise Edition already has several gateways built into it including the following:

Authorize.NET is a credit card processor. There are two variations of using this service – the first is a direct call to the service with credit card data to authorize or process a sale transaction. The second is to use the CIM service (Customer Information Manager) which allows saving of credit card information by customer. Customer Profiles are maintained with a token and each credit card within a customer (called a Customer Payment Profile) is also tokenized so that no credit card data is ever saved in Sitecore Ecommerce Enterprise Edition.

PayFlowPro is another credit card processor. They do not currently allow saving profile information, however they do allow something called a 'reference transaction'. This means that a valid authorization token may be re-used for a future sale or additional authorization as long as it is within some time limits (typically 30 days) established by the merchant bank and/or credit card processor.

CyberSource is used currently for international transactions.

eSelect is a credit card processing module from Moneris (called eSelectPlus) that we integrate with using their API.

PCCharge is another credit card processing module that is installed on a user's system and then accessed directly via an IP port. This is sold typically for a POS application and is produced by VeriFone.

PayPal has been implemented as a payment method rather than an entire payment gateway and can be added to an existing site using a different gateway. It is a way to 'jump out' of the normal credit card process and grab the payment from PayPal and return.

Setting: AuthorizeNetCIMServiceLoginID	**Valid Values:**

Where Used: AuthorizeNetCIMProcessor
Purpose/Notes: This value is provided by Authorize.NET and is used to effectively log into the CIM processor. CIM is the Customer Information Manager and allows for storing credit card information and is accessed by a token.

Setting: AuthorizeNetCIMServiceTransactionKey	**Valid Values:**
Where Used: AuthorizeNetCIMProcessor	
Purpose/Notes: This value is provided by Authorize.NET and is used along with the LoginID to effectively log into the CIM processor.	

Setting: AuthorizeNetCIMServiceURL	**Valid Values:** URL
Where Used: AuthorizeNetCIMProcessor	
Purpose/Notes: This value is used to point to the correct URL for Authorize.Net and can be changed between the test environment and production environment. This is used for creating/maintaining customer profiles.	
Default: `https://apitest.authorize.net/soap/v1/Service.asmx`	

Setting: AuthorizeNetCIMServiceValidationMode	**Valid Values:** test or live
Where Used: AuthorizeNetCIMProcessor	
Purpose/Notes: When creating a customer payment profile (saved card) in live mode, the system will create a .01 authorization and then automatically void it. Some calls to Authorize.Net ask for the mode (live or test) being used and will behave in slightly different ways depending on the value passed. It is recommended to coordinate the URL and validation modes to test for pilot and live for production.	

Setting: BankCode	**Valid Values:**
Where Used: All payment processors	
Purpose/Notes: When a transaction is made using a payment processor, there is a property on the transaction called BankCode which is generally used to indicate the bank account the funds would be entered into with a cash receipt during integration. This code would represent the ERP's bank code in most instances, although it is possible that a specific payment gateway uses the field for a different purpose.	

Setting: PaymentGateway	**Valid Values:** AuthorizeNet CyberSource Dummy ESelect PayflowPro PCCharge
Where Used: DatasourceSingleton	
Purpose/Notes: This setting is used to determine which payment gateway logic is being applied. It should match the pay	

Setting: Payflow_Cert_Path	**Valid Values:**
Where Used: PaymentGateway_PayflowPro	
Purpose/Notes: Location on the server of the SSL certificate, typically in the lib\certs directory of the site and is used as part of the connection to the PayflowPro processor. Seems to only be used in the PaypalProPayFlowProTests program, no needed for production	

Setting: Payflow_Partner	**Valid Values:**
Where Used: PaymentGateway_PayflowPro	
Purpose/Notes: This is the partner ID provided by your VeriSign reseller. If you signed up yourself, use VeriSign.	

Setting: Payflow_Password	**Valid Values:**
Where Used: PaymentGateway_PayflowPro	
Purpose/Notes: This is the password assigned by PayFlowPro to communicate with them	

Setting: Payflow_Port	**Valid Values:** typically 443
Where Used: PaymentGateway_PayflowPro	
Purpose/Notes: This is the specific port ID on the PayFlow system to communicate with	

Setting: Payflow_User	**Valid Values:**
Where Used: PaymentGateway_PayflowPro	
Purpose/Notes: This is the username assigned by PayFlow for transactions being processed – this along with the partner and password help define who the transaction request is coming from. You may have multiple users within a vendor account.	

Setting: Payflow_Vendor	**Valid Values:**

Where Used: PaymentGateway_PayflowPro
Purpose/Notes: This is the vendor code assigned by PayFlow to identify the selling company

Setting: PaymentGateway_Authorize.net_login	**Valid Values:**
Where Used: PaymentGateway_AuthorizeNet	
Purpose/Notes: This is the login ID assigned to you by Authorize.Net – this is used along with the transaction key to identify the company processing the transaction	

Setting: PaymentGateway_Authorize.net_posturl	**Valid Values:**
Where Used: PaymentGateway_AuthorizeNet	
Purpose/Notes: This is the URL to the processor – note that there is a different address for test vs. live. Make sure to reset this value when going into production.	
Default Value: `https://test.authorize.net/gateway/transact.dll`	

Setting: PaymentGateway_Authorize.net_send_email_recipt (sic)	**Valid Values:** TRUE/FALSE
Where Used: PaymentGateway_AuthorizeNet	
Purpose/Notes: This is an indicator to Authorize.Net to send a credit card transaction receipt to the purchaser if the email address is provided during the transaction	

Setting: PaymentGateway_Authorize.net_test_mode	**Valid Values:** TRUE/FALSE
Where Used: PaymentGateway_AuthorizeNet	
Purpose/Notes: Indicates to Authorize.Net if this is a test transaction	

Setting: : PaymentGateway_Authorize.net_transaction_key	**Valid Values:**
Where Used: PaymentGateway_AuthorizeNet	
Purpose/Notes: This is the assigned transaction key that is used along with the net_login to identify the transaction requestor.	

Setting: : PaymentGateway_CyberSource_MerchantID	**Valid Values:**
Where Used: PaymentGateway_CyberSource	
Purpose/Notes: This is the assigned ID to the selling company	

Setting: PaymentGateway_CyberSource_TransactionKey	**Valid Values:**
Where Used: PaymentGateway_CyberSource	
Purpose/Notes: This is an assigned transaction key that, along with the merchant ID, authenticates the transaction. It is quite a long code (about 345 characters)	

Setting: PaymentGateway_CyberSource_URL	**Valid Values:**
Where Used: PaymentGateway_CyberSource	
Purpose/Notes: This is the URL to the web service to conduct transactions	
Default: `https://ics2wstest.ic3.com/commerce/1.x/transactionProcessor`	

Setting: : PaymentGateway_eSelect_APIToken	**Valid Values:**
Where Used: PaymentGateway_ESelect	
Purpose/Notes: This is an assigned authentication token to make valid transaction requests.	

Setting: : PaymentGateway_eSelect_Host	**Valid Values:**
Where Used: PaymentGateway_ESelect	
Purpose/Notes: This is the host URL to post transactions to	
Default: `esqa.moneris.com`	

Setting: : PaymentGateway_eSelect_User	**Valid Values:**
Where Used: PaymentGateway_ESelect	
Purpose/Notes: This is the assigned user ID that, along with the API token, is used to authenticate transactions	

Setting: : PaymentGateway_PCCharge_IPAddress	**Valid Values:**
Where Used: PaymentGateway_PCCharge	
Purpose/Notes: Since PCCharge is a PC-based solution, this is the accessible IP address to talk to the processor service	
Setting: : PaymentGateway_PCCharge_MerchantNumber	**Valid Values:**

Where Used: PaymentGateway_PCCharge
Purpose/Notes: This is the assigned merchant number to identify the seller

Setting: : PaymentGateway_PCCharge_Port	**Valid Values:**
Where Used: PaymentGateway_PCCharge	
Purpose/Notes: Port number on the PC to accept the incoming transaction using tcp	
Default: 31419	

Setting: : PaymentGateway_ PCCharge_ProcessorID	**Valid Values:**
Where Used: PaymentGateway_PCCharge	
Purpose/Notes: This identifies the credit card processor that PCCharge is actually using (i.e. PCCharge, ICVerify, MAPP)	

Setting: : PaymentGateway_ PCCharge_UseSessionUsers	**Valid Values:** TRUE/FALSE
Where Used: PaymentGateway_PCCharge	
Purpose/Notes: If in test mode, this should be FALSE and will use 'User1' as the default value. Once it is set to TRUE, the session ID is set as the user for the transaction to ensure multiple transactions submitted concurrently do not conflict.	

Setting: PayPal_APIPassword	**Valid Values:**
Where Used: PayPalExpress	
Purpose/Notes: This is the password assigned by PayPal to be used in authentication	

Setting: PayPal_APISignature	**Valid Values:**
Where Used: PayPalExpress	
Purpose/Notes: This is another piece of the authentication process and is assigned by PayPal and used along with the password and user name	

Setting: PayPal_APIUsername	**Valid Values:**
Where Used: PayPalExpress	
Purpose/Notes: Assigned by PayPal to identify the selling company	

Setting: PayPal_Live	**Valid Values:** TRUE/FALSE
Where Used: PayPalExpress	

Purpose/Notes: If TRUE, uses www.paypal.com for transactions, otherwise uses www.sandbox.paypal.com

Setting: Show_CC_Errors	**Valid Values:** TRUE/FALSE
Where Used: Web.config,PaymentGateways.CreditCardTransactionException	
Purpose/Notes: If true, sends credit card processor errors to the application log	

Setting: HostCompanyName	**Valid Values:**
Where Used: PayPalExpress, Web.config	
Purpose/Notes: Used to construct the credit card order description (Company Name + " Web Order")	

Email Processing

There are several optional emails that the system can generate. Some are process oriented, such as an Order Confirmation email or a Shipment Confirmation email while others are for error alerts such as when a critical transaction fails and email is used as part of an alert system.

Setting: AdminEmailAddress	**Valid Values:**
Where Used: Multiple	
Purpose/Notes: Generally, this is the web site's administrator and various emails will be generated to alert the admin that there are some difficulties with the site as noted below	
Specific Use: Insite.Model.WebSite - Sends an email to the admin if web page content is missing	
Specific Use: ErrorHandler – errors are send to this address as well as to the SystemAdminEmailAddress recipient	

Setting: CustomerServiceEmailAddress	**Valid Values:**
Where Used: BuildEmailValues_Generic	
Purpose/Notes: If the system is in live mode (Emails_SendToTestAddress = false) and there is a valid email in this setting, the shipment confirmation email is sent to the customer service rep as well as to the customer.	

Setting: DefaultEmailAddress	**Valid Values:**
Where Used: Emails.EmailListProcessor, Emails.ShipmentConfirmationEmail,	
Purpose/Notes: This is the "from" email address used for emails being sent when not specifically defined in the EmailList object	

Setting: Emails_SendToTestAddress	**Valid Values:** TRUE/FALSE
Where Used: Emails.BuildEmailValues_Generic	
Purpose/Notes: If set to true, then the EmailTestAddress is used rather than the customer's email for order confirmation and shipment confirmation emails	

Setting: EmailTestAddress	**Valid Values:**
Where Used: BuildEmailValues_Generic	
Purpose/Notes: When the email test mode is true (Emails_SendToTestAddress) then this address is the recipient of shipment confirmation emails and order confirmation emails.	

Setting: OrderNotificationEmail	**Valid Values:**
Where Used: Emails.BuildEmailValues_Generic, RefreshBase	
Purpose/Notes: If the emails are not in test mode (Emails_SendToTestAddress = false), then this is the optional recipient of all order confirmation emails in addition to the customer. Additionally, if there is an error sending email, this person will get a notification along with the SystemAdminEmail recipient.	

Setting: SendShipmentConfirmationWithoutPackageLine	**Valid Values:** TRUE/FALSE
Where Used: Emails.ShipmentConfirmationEmail	
Purpose/Notes: In general, shipment confirmation emails are only sent if there are package(s) defined within shipment(s) in the order. If this option is turned on, it will only send the email if there are actual package lines (package detail, typically from InSiteShip) with quantities shipped.	

Setting: SystemAdminEmailAddress	**Valid Values:**
Where Used: RefreshBase, ErrorHandler	
Purpose/Notes: If there is a problem/error sending emails or with other errors, the person with this email address will receive a notification	

Order Calculations

Setting: AutoApplyPromotions	**Valid Values:** TRUE/FALSE
Where Used: CustomerOrder, OrderLine	
Purpose/Notes: If this value is false, promotions are not applied to the order	

Setting: AutoCalculatePrice	**Valid Values:** TRUE/FALSE
Where Used: CustomerOrder, OrderLine	
Purpose/Notes: If this flag is set to TRUE, then the order line will be recalculated when data in the line changes such as the quantity. This is important for break-pricing when the quantity could change the price. The initial add line will calculate the price regardless and this is a setting that can help improve speed if the price calculation is slow. If this is FALSE, then a manual call to CalculatePrice will need to be done as appropriate.	

Setting: AutoCalculateShipping	**Valid Values:** TRUE/FALSE
Where Used: CustomerOrder	
Purpose/Notes: If this flag is set to TRUE, then if the requested shipdate is changed (not a typical field to be set), the system will recalculate the current shipping costs. Normally this would be used, for example, to automatically set the next day delivery charge if the requested shipdate is 1 day away. This setting is likely to be deprecated.	

Setting: TaxCalculator_StorePickupShipCode	**Valid Values:** ShipViaCode
Where Used: CustomerOrder	
Purpose/Notes: If the order is going to be picked up at the store (only single store supported for now) then the taxes applied need to be calculated based on the store's location, not the ship-to address. The ship via code defined here will trigger this behavior.	

Setting: TaxCalculator_StorePickupStateAbbreviation	**Valid Values:** State code
Where Used: CustomerOrder	
Purpose/Notes: If the order is being picked up at a store, as defined by the ship via code, this is used to determine which state for which to calculate state tax	

Setting: TaxCalculator_StorePickupZipCode	**Valid Values:** Zip code
Where Used: CustomerOrder	
Purpose/Notes: If the order is being picked up at a store, as defined by the ship via code, this is used to determine which tax locality to use to calculate local tax	

Setting: TaxCalculator_TaxCode1ForPercent	**Valid Values:**

Where Used: TaxCalculator_Generic
Purpose/Notes: Sets the taxcode1 field in the order if this value is present and if there is a tax calculation percent applied to the entire web site (overrides local/state tax)

Setting: UseBasicPricing	**Valid Values:** TRUE/FALSE
Where Used: CustomerOrder	
Purpose/Notes: If this setting is set to TRUE, then the Basic List Price or the Basic Sale Price will be used to determine the order total. If this setting is set to FALSE, the Price Matrix will be used to determine pricing.	

Setting: UseWebSiteCarriers	**Valid Values:** TRUE/FALSE
Where Used: CustomerOrder	
Purpose/Notes: If this setting is set to TRUE, then the carriers associated with the web site are used for calculating shipping costs and the customer carrier information is not consulted. If, however, the setting is FALSE, then it will consult the settings of the individual customer which will control the shipping options available. Note that every customer must then have shipping options established if this option is used.	

Shipping Options

The following settings control how the shipping engine behaves.

Setting: DefaultPackageHeight	**Valid Values:** integer
Where Used: OrderPackager_Generic	
Purpose/Notes: If no specific package is defined for a carrier, this is the default for calculations	

Setting: DefaultPackageLength	**Valid Values:** integer
Where Used: OrderPackager_Generic	
Purpose/Notes: : If no specific package is defined for a carrier, this is the default for calculations	

Setting: DefaultPackageWidth	**Valid Values:** integer
Where Used: OrderPackager_Generic	
Purpose/Notes: If no specific package is defined for a carrier, this is the default for calculations	

Setting: FedExSaveXML	**Valid Values:** TRUE/FALSE
Where Used: FedExRatingService	
Purpose/Notes: Save the XML of the FedEx Rating Service. Set to false unless debugging.	

Setting: Shipping_GiftCardCarrier	**Valid Values:** Valid shipvia
Where Used: ShippingEngine	
Purpose/Notes: If the order contains only a gift card, then pre-specify the carrier using this option, otherwise the normal rules for shipping will apply	

Setting: TrackingUrl_<carrier>	**Valid Values:**
Where Used: ShipmentPackage	
Purpose/Notes: This is the URL associated with the specific carrier for doing shipment tracking and is used in both the order history page for an embedded link and in the shipment confirmation email. Current links include carrier codes UPS,FDX,SPD,DHL,OLDD (Old Dominion) for when use custom URLs is false. Note that the actual tracking number replaces the phrase “[trackingnumber]” in the specified custom URL.	

Setting: TrackingUrl_UseCustomUrls	**Valid Values:** TRUE/FALSE
Where Used: ShipmentPackage	
Purpose/Notes: If this is set to true, then use the URL specified in trackingurl_<carrier>, otherwise use the hardcoded ones denoted	

Setting: UPSUseResidentialShipping	**Valid Values:** TRUE/FALSE
Where Used: UPSRatingService	
Purpose/Notes: Determines how to set the residential flag when rating UPS shipments – this is a global setting for all calls to UPS and is currently not customer or address specific and the default is TRUE	

Web Site Parameters

The following settings are used throughout the specific site for various behaviors and display options, many requiring design to implement.

Setting: AllowNegativeQtyOnHand - DEPRECATE	**Valid Values:** TRUE/FALSE
Where Used: CustomerOrderProcessor	
Purpose/Notes: When this value is turned on, then the quantity on hand is decremented on every order, otherwise it is only decremented if there is sufficient quantity. This option will be removed in a future version or implemented as an actual ordering constraint.	

Setting: ApplicationLog_LogDebugMessages	**Valid Values:** TRUE/FALSE
Where Used: ApplicationLog, ApplicationLogController	
Purpose/Notes: When a debug message is issued, if this parameter is on, it will log the message to the application log. There are a number of specific places where this may be called within the application. It should only be turned on during development or to track/troubleshoot issues. It is used heavily in the integration services but is also used elsewhere.	

Setting: DefaultCountry	**Valid Values:**
Where Used: Country	
Purpose/Notes: Unable to locate where this is used	

Setting: DefaultState	**Valid Values:**
Where Used: State	
Purpose/Notes: Unable to locate where this is used	

Setting: GiftCardExpYears	**Valid Values:** integer
Where Used: CustomerOrderProcessor	
Purpose/Notes: When a gift card is purchased and created by the system, this parameter is used to automatically set the expiration date of the card. If you wish to effectively never expire the card, set this value to something like 99	

Setting: ImagePath	**Valid Values:**
Where Used: CommonPageBase	
Purpose/Notes: Identifies to the web site the relative path of where images are stored. This is typically stored with a tilde to indicate a relative path such as ~/Images/	

Setting: InSiteCommerceMode	**Valid Values:** Pilot, Production, Development

Where Used: CommonPageBase
Purpose/Notes: This is used to determine if the pages need to be served up in SSL or not (https:). Typically the pilot and production sites are served up this way but the development site is not.

Setting: NotFoundLargeImagePath	**Valid Values:**
Where Used: Product	
Purpose/Notes: The specific image to be used when a large image is not found	

Setting: NotFoundMediumImagePath	**Valid Values:**
Where Used: Product	
Purpose/Notes: The specific image to be used when a medium image is not found	

Setting: NotFoundSmallImagePath	**Valid Values:**
Where Used: Product	
Purpose/Notes: The specific image to be used when a small image is not found	

Setting: PageTitleDelimiter	**Valid Values:** single character
Where Used: CommonPageBase	
Purpose/Notes: This is used when generating the page title which includes, optionally, the site name and delimited with the product short description where needed	

Setting: PageTitleIncludesSiteName	**Valid Values:** TRUE/FALSE
Where Used: CommonPageBase	
Purpose/Notes: Indicates to the page constructor if the site name should be included in the title or not	

Setting: PageTitleSiteNameAfterTitle	**Valid Values:** TRUE/FALSE
Where Used: CommonPageBase	
Purpose/Notes: Indicates to the page constructor if the site name (if included) is before or after the other page title information	

Setting: PreviouslyViewedItemsToStore	**Valid Values:** integer
Where Used: CommonPageBase	

Purpose/Notes: Stores a list of other products that have been looked at and can be incorporated into the web site to allow the user to see what they have already shopped for. This comma delimited list is stored in the session variable

Setting: ReleaseForNewOrderLines	**Valid Values:** integer
Where Used: CustomerOrder	
Purpose/Notes: The order lines contain both a line and release # which is used by various ERP systems. If your ERP system needs to have this, set the value as needed, the default is 0 and is stored in OrderLine.Release	

Setting: ReplaceOnAddToCart	**Valid Values:** TRUE/FALSE
Where Used: CustomerOrder, WishList	
Purpose/Notes: If TRUE then the quantity selected on a page simply replaces the current quantity in the page otherwise it is added to the quantity. For example, if the default is 1 and this value is false (default) and the user hits the add to cart button 3 times, they will have a quantity of 3 in the cart. This also affects WishList behavior.	

Setting: StorePreviouslyViewedItems	**Valid Values:** TRUE/FALSE
Where Used: CommonProductPageStrategy	
Purpose/Notes: Used in conjunction with the number of viewed items stored, this option triggers the session storing the items viewed. This is a design-specific parameter that must be supported in the particular web site.	

Integration Settings

There are many points of integration between the site and the backend ERP system that both pull and push data. Each integration starts off with its own project area and is incorporated into the overall integration strategy. We will be moving towards a more centralized configuration approach in the future so many of these settings will be consolidated or deprecated.

Please note that while there are many parameters to allow/disallow a given integration to be run, unless the integration has been created for the specific target ERP as a valid task, it will not have any effect.

Refer to the Integration Technical Details document to explore how the task builder is invoked from the integration service. A standardized dataset is built from the given ERP datasource for things like Product Refresh, Inventory Refresh, and Customer Refresh. These values are then placed into the Sitecore Ecommerce Enterprise Edition data store in a standardized fashion.

Setting: AssignPackageNumber	**Valid Values:** TRUE/FALSE
Where Used: ShipmentRefresh	
Purpose/Notes: If set to true, the assumption is that a package number is assigned during the shipment refresh and then we transcribe this data into the ShipmentPackage data. If false and if we are creating package detail lines, then all lines would go into the first package.	

Setting: SavedOrderRetentionDays	**Valid Values:**
Where Used: CustomerOrder.aspx	
Purpose/Notes: Used in the PurgeSavedOrders to automatically delete old orders over a certain number of days old that were never completed	

Setting: CustomerDefault_Warehouse	**Valid Values:**
Where Used: Syteline7Integration	
Purpose/Notes: Used to establish what warehouse the order line will be submitted with and to ensure that there is an itemwhse record for the part and warehouse established	

Setting: DaysToStoreEncryptedCC	**Valid Values:**
Where Used: BatchProcessor – old MC	
Purpose/Notes: Part of the PurgeEncryptedNumbers batch job to clear out encrypted credit card data over a certain number of days old	

Setting: Default_BankCode	**Valid Values:**
Where Used: Signin.aspx	
Purpose/Notes: Used to establish default on new customers	

Setting: Default_CreditLimit	**Valid Values:**

Where Used: Signin.aspx
Purpose/Notes: Used to establish default on new customers

Setting: Default_CurrencyCode	**Valid Values:**
Where Used: Signin.aspx	
Purpose/Notes: Used to establish default on new customers	

Setting: Default_CustomerType	**Valid Values:**
Where Used: web site code	
Purpose/Notes: Used to establish default on new customers	

Setting: Default_EndUserType	**Valid Values:**
Where Used: Signin.aspx	
Purpose/Notes: Used to establish default on new customers	

Setting: Default_InvoiceFrequency	**Valid Values:**
Where Used: Signin.aspx	
Purpose/Notes: Used to establish default on new customers	

Setting: Default_PriceCode	**Valid Values:**
Where Used: Signin.aspx	
Purpose/Notes: Used to establish default on new customers	

Setting: Default_Salesman	**Valid Values:**
Where Used: Signin.aspx	
Purpose/Notes: Used to establish default on new customers	

Setting: Default_ShipCode	**Valid Values:**
Where Used: Signin.aspx	
Purpose/Notes: Used to establish default on new customers	

Setting: Default_TermsCode	**Valid Values:**
Where Used: Signin.aspx	

Purpose/Notes: Used to establish default on new customers

Setting: Default_Warehouse	**Valid Values:**
Where Used: Signin.aspx	
Purpose/Notes: Used to establish default on new customers	

Setting: EnableOrderStatusChange	**Valid Values:**
Where Used: OLD M/C	
Purpose/Notes: Allowed or disallowed the editing of an order status	

Setting: ERP_AddShipTos	**Valid Values:**
Where Used: SXeAPIIntegration	
Purpose/Notes: Add ShipTos to ERP during order submission	

Setting: ERP_AddShipToPrefix	**Valid Values:**
Where Used: SXeAPIIntegration	
Purpose/Notes: Prefix added to ship-to code when creating a new entry	

Setting: ERP_BankCode	**Valid Values:**
Where Used: Syteline7Integration	
Purpose/Notes: Default value used for SyteLine7 arpmt.bank_code when submitting payments	

Setting: ERP_CompanyNumber	**Valid Values:**
Where Used: TaskBuilder_SXe, SXeIntegration	
Purpose/Notes: Company number used for SX integration	

Setting: ERP_ConnectionString	**Valid Values:**
Where Used: TaskBuilderBase	
Purpose/Notes: Connection string used to create direct ERP connections	

Setting: ERP_CreditCardCashAccountID	**Valid Values:**
Where Used: Epicor9Integration, VantageIntegration	
Purpose/Notes: Used for submitting the payment information for credit cards. This is used to set the BankAcctID on the cash receipts batch.	

Setting: ERP_CustomerDefaultWarehouse	**Valid Values:**
Where Used: SXeAPIIntegration	
Purpose/Notes: Customer warehouse on new customers being created	

Setting: ERP_DealerRefreshDeletesDealers	**Valid Values:** TRUE/FALSE
Where Used: DealerRefresh	
Purpose/Notes: When used, this parameter indicates if the current set of details should be cleared prior to re-importing	

Setting: ERP_DefaultCurrencyCode	**Valid Values:**
Where Used: PriceCalculator_Generic	
Purpose/Notes: Used for the price calculator to pass in a price to the price matrix unless there is a session-specific currency code which is set in the site optionally	

Setting: ERP_DeleteSalesmanOnRefresh	**Valid Values:** TRUE/FALSE
Where Used: SalesmanRefresh	
Purpose/Notes: This parameter indicates if the salesman file should be cleared prior to importing new data	

Setting: ERP_DepositAccount	**Valid Values:**
Where Used: Syteline7Integration	
Purpose/Notes: : Used for arpmtd.deposit_acct when submitting payments	

Setting: ERP_EpicorConnectionUserName	**Valid Values:**
Where Used: Epicor9Integration, VantageIntegration	
Purpose/Notes: Used for logging into the web services	

Setting: ERP_EpicorConnectionPassword	**Valid Values:**
Where Used: Epicor9Integration, VantageIntegration	
Purpose/Notes: Used for logging into the web services	

Setting: ERP_EpicorX509CertName	**Valid Values:**
Where Used: Epicor9Integration, VantageIntegration	
Purpose/Notes: The friendly name of the certificate for WSE policy settings	

Setting: ERP_EpicorX509Encryption	**Valid Values:** TRUE/FALSE
Where Used: Epicor9Integration,VantageIntegration	
Purpose/Notes: Used to determine if an X509 certificate is being used for access to the Epicor/Vantage web services. This is normally set to false since we install the integration service on the same box as the web services.	

Setting: ERP_EpicorServicesURLHead	**Valid Values:**
Where Used: VantageIntegration, Epicor9Integration	
Purpose/Notes: Used to denote where the Vantage/Epicor9 web services are located relative to where the integration service is running	

Setting: ERP_ExecuteQuery	**Valid Values:** TRUE/FALSE
Where Used: ExecuteQuery	
Purpose/Notes: If this parameter is on, execute query tasks are executed otherwise they are not. These tasks tend to be ERP and customer-specific and offer another way of getting integration data.	

Setting: ERP_FreightCode	**Valid Values:**
Where Used: Epicor9Integration, VantageIntegration	
Purpose/Notes: Indicates the miscellaneous charge code to use when posting over quoted freight	

Setting: ERP_FreightEstimated	**Valid Values:** TRUE/FALSE
Where Used: Epicor9Integration, VantageIntegration	
Purpose/Notes: Used to determine if the freight will be posted into the ERP system. For Vantage/Epicor, if it is not estimated (it is considered 'quoted') then the bill shipper flag is set so that shipping integration does not add to the freight already added to the order.	

Setting: ERP_ICSOrderNumberField	**Valid Values:**

Where Used: SXeAPIIntegration
Purpose/Notes:.This si the field in the ERP that holds the ISC order #

Setting: ERP_IntegrationServiceAllowedMachines	**Valid Values:**
Where Used: IntegrationService	
Purpose/Notes: This is a comma-separated string of machine names that are allowed to call the integration service. If the machine making the service call is not allowed, an email is sent to the email address indicated in the ERP_IntegrationServiceSendMachineEmails. Currently, if no machines are explicitly allowed, it will allow the machine to communicate but will log a message and send an email to the system administrator. Note that this setting is case sensitive.	

Setting: ERP_InventoryRefreshWarehouse	**Valid Values:**
Where Used: TaskBuilder_SXe,TaskBuilder_SXeAPI	
Purpose/Notes: For SX denotes the specific warehouse to use when retrieving inventory balances	

Setting: ERP_InvoiceLookbackDays	**Valid Values:**
Where Used: SXeAPIIntgration	
Purpose/Notes: For invoice refresh, separate lookback period than ERP_LookbackDays	

Setting: ERP_LookbackDays	**Valid Values:** integer
Where Used: TaskBuilder_MAS200, TaskBuilder_Solomon, TaskBuilder_Syteline6, TaskBuilder_Syteline7, Epicor9Integration, VantageIntegration	
Purpose/Notes: This parameter is used for various historical refreshes to only go back and refresh data for a certain number of days to eliminate re-processing old data. Refresh Shipments, Orders, and Invoices being primary examples.	

Setting: ERP_OrderEntryAddBatchSchedulerRecords	**Valid Values:** TRUE/FALSE
Where Used: SXeIntegration	
Purpose/Notes: If false, the oeebu process mustr be run manually in SX. If true, then the records to kick off the oeebu via the report scheduled will be added	

Setting: ERP_OrderEntryCustomerNumber	**Valid Values:**
Where Used: SXeIntegration	
Purpose/Notes: When a single customer is being used in the ERP, use this value. Currently, the direct write version of SX integration assumes a static customer.	

Setting: ERP_OrderEntryWarehouse	**Valid Values:**
Where Used: SXeIntegration	
Purpose/Notes: If no warehouse is specified, use this value as the default	

Setting: ERP_OrigSite	**Valid Values:**
Where Used: Syteline7Integration	
Purpose/Notes: Value used for SyteLine7 arpmtd.site when submitting payments	

Setting: ERP_OrigSiteQuery	**Valid Values:**
Where Used: Syteline7Integration	
Purpose/Notes: Query to determine value used for SyteLine7 arpmtd.site when submitting payments	

Setting: ERP_Password	**Valid Values:**
Where Used: RefreshBase	
Purpose/Notes: Used along with ERP_Username for login information to ERPs when ERP_SubmitCredentials is true	

Setting: ERP_PricingStoredProcedure	**Valid Values:**
Where Used: PriceCalculator_Generic	
Purpose/Notes: Indicates the stored procedure to run in the ISC db to calculate pricing using the price matrix	

Setting: ERP_Product.ActiveField	**Valid Values:**
Where Used: TaskBuilder_Solomon, TaskBuilder_Syteline7, TaskBuilder_Syteline6	
Purpose/Notes: Field used to query the ERP database to obtain if a product is active	

Setting: ERP_Product.ModelNumberField	**Valid Values:**
Where Used: TaskBuilder_Solomon, TaskBuilder_Syteline7, TaskBuilder_Syteline6	
Purpose/Notes: Field used to query the ERP database to obtain the product model number	

Setting: ERP_Product.PriceField	**Valid Values:**
Where Used: TaskBuilder_Solomon, TaskBuilder_Syteline7, TaskBuilder_Syteline6	
Purpose/Notes: Field used to query the ERP database to obtain the price	

Setting: ERP_Product.UPCCodeField	**Valid Values:**
Where Used: TaskBuilder_Solomon, TaskBuilder_Syteline7, TaskBuilder_Syteline6	
Purpose/Notes: Field used to query the ERP database to obtain the UPC code	

Setting: ERP_ProductCrossRefRecTypes	**Valid Values:**
Where Used: SXeAPIIntegration	
Purpose/Notes: Filter used when querying cross-ref table (icsec)	

Setting: ERP_ProductRefreshCustomProcess	**Valid Values:**
Where Used: SXeAPIIntegration	
Purpose/Notes: Holds any custom process to be run after the Product Refresh – currently used to mark products not updated in a batch mode as obsolete	

Setting: ERP_PromoMiscCharge	**Valid Values:** TRUE/FALSE
Where Used: Epicor9Integration,.VantageIntegration	
Purpose/Notes: When this is set to true, then the promotion is set as a negative miscellaneous charge otherwise the promotion is netted against the individual lines on the order, effectively being discounted.	

Setting: ERP_PromoMiscChargeCode	**Valid Values:**
Where Used: Epicor9Integration,.VantageIntegration	
Purpose/Notes: When posting the promotional discounts as miscellaneous charges, then this is the miscellaneous charge code to use to do so.	

Setting: ERP_PurchaseOrderNumberSuffix	**Valid Values:**
Where Used: SXeAPIIntegration	

Purpose/Notes: Suffix added to the PO # field when submitting an order

Setting: ERP_RefreshCustomerBillToCustomColumns	**Valid Values:**
Where Used: TaskBuilder_SXe	
Purpose/Notes:.List of custom columns to pull in with the bill-to customer	

Setting: ERP_RefreshCustomerPricing	**Valid Values:** TRUE/FALSE
Where Used: TaskBuilder_Solomon, TaskBuilder_Syteline7, TaskBuilder_Syteline6	
Purpose/Notes: Used to determine if customer pricing is to be refreshed or not – this is a separate refresh process for the noted integrations	

Setting: ERP_RefreshCustomerProducts	**Valid Values:** TRUE/FALSE
Where Used: TaskBuilder_Solomon, TaskBuilder_Syteline7, TaskBuilder_Syteline6, VantageIntegration, Epicor9Integration	
Purpose/Notes: This is an optional refresh that is used to pull over customer product-specific product codes, typically only used for B2B sites – populates the Customer Product table	

Setting: ERP_RefreshCustomers	**Valid Values:** TRUE/FALSE
Where Used: CustomerRefresh	
Purpose/Notes: One of the standard refresh functions to pull over/synch customer data from the ERP to the ISC database	

Setting: ERP_RefreshDealers	**Valid Values:** TRUE/FALSE
Where Used: DealerRefresh	
Purpose/Notes: If this function is turned on, then the dealer refresh process will run to synch ERP-based dealers (which could be a subset of customers potentially) into the ISC Dealer table	

Setting: ERP_RefreshInventory	**Valid Values:** TRUE/FALSE
Where Used: InventoryRefresh	
Purpose/Notes: If this function is turned on, then the inventory refresh will run to update the current inventory balances. It also performs a rollup and consolidation function to compress older transactions	
Setting: ERP_RefreshInvoices	**Valid Values:** TRUE/FALSE
Where Used: InvoiceRefresh	
Purpose/Notes: If this function is turned on, the invoices from the ERP are read and synched to the Invoice and Invoice Line tables to support invoice history functions, typically only in B2B sites	

Setting: ERP_RefreshLookbackDays	**Valid Values:** integer
Where Used: BatchProcessor – Old MC	
Purpose/Notes: Used in the RefreshCommerceCustomerOrders or RefreshAll batches – should be changed to use ERP_LookbackDays.	

Setting: ERP_RefreshOrders	**Valid Values:** TRUE/FALSE
Where Used: CustomerOrderRefresh	
Purpose/Notes: If this function is turned on, the orders from the ERP are read and synched to the Order and OrderLine tables. Usually this will be used to add in orders that were not generated from the web and to update status information but line data is not typically updated or resynched from the ERP	

Setting: ERP_RefreshProductColumnsNotToOverwrite	**Valid Values:**
Where Used: ProductRefresh_Generic	
Purpose/Notes: Comma-separated list of fields that should not be overwritten in the ISC database even if they are brought over from the ERP. This is valuable if there is data that is available in the ERP but is maintained within ISC.	

Setting: ERP_RefreshProducts	**Valid Values:** TRUE/FALSE
Where Used: ProductRefresh_Generic	
Purpose/Notes: This is the function used to indicate that products should be synched from the ERP	

Setting: ERP_RefreshSalesman	**Valid Values:** TRUE/FALSE
Where Used: TaskBuilder_Syteline7, SalesmanRefresh	
Purpose/Notes: Indicates if sales reps should be pulled from the ERP – very few of these integrations have been written	

Setting: ERP_RefreshShipmentPackages	**Valid Values:** TRUE/FALSE
Where Used: TaskBuilder_Solomon, TaskBuilder_Syteline6, TaskBuilder_Syteline7	
Purpose/Notes: This refresh is designed to pull in and populate ShipmentPackage and ShipmentPackageLine data	

Setting: ERP_RefreshShipments	**Valid Values:** TRUE/FALSE
Where Used: TaskBuilder_Solomon, TaskBuilder_Syteline6, TaskBuilder_Syteline7, ShipmentRefresh	
Purpose/Notes: Generally this refresh is designed to update the Shipment and ShipmentPackage data	

Setting: ERP_SaveDataSet	**Valid Values:** TRUE/FALSE
Where Used: RefreshBase	
Purpose/Notes: This function will allow the system to save the XML datasets being sent by the integration service for debugging purposes and should always be false unless you are debugging the integration. The data will be sotred in the web site patch “DataSet” if there is a current site context, otherwise it will be stored in c:\temp as an xml file.	

Setting: ERP_SetOrderToCompleteWhenShipped	**Valid Values:** TRUE/FALSE
Where Used: ShipmentRefresh	
Purpose/Notes: The idea is that if there is a shipment against the order that the status in ISC will be set to Complete. This function only looks to see if any shipment has been processed against an order, not if the order has ultimately been shipped complete.	

Setting: ERP_ShipmentDateColumn	**Valid Values:**
Where Used: TaskBuilder_Solomon, TaskBuilder_Syteline6, TaskBuilder_Syteline7	
Purpose/Notes: Identifies the column to use in the retrieval query to limit the shipments to retrieve based on the shipment date for a bulk refresh	

Setting: ERP_ShipmentOrderNumberColumn	**Valid Values:**
Where Used: TaskBuilder_Solomon, TaskBuilder_Syteline6, TaskBuilder_Syteline7	
Purpose/Notes: Identifies the column to use in retrieving the shipments for a specific order #	

Setting: : ERP_ShipmentPackage.Carrier	**Valid Values:**
Where Used: TaskBuilder_Mas200, TaskBuilder_SXe	
Purpose/Notes: Embeds a fixed carrier since there is no carrier information in the MAS200 shipments; Not sure it's actually used in SXe	

Setting: : ERP_ShipmentRefreshOrderOnShipmentRefresh	**Valid Values:** TRUE/FALSE
Where Used: ShipmentRefresh	
Purpose/Notes: If true, then customer orders associated with the shipment are also refreshed – this presumes that the order refresh is not already set, otherwise this would be redundant	

Setting: ERP_RefreshProductCrossSells	**Valid Values:** TRUE/FALSE
Where Used: ProductRefresh	
Purpose/Notes: When this option is activated, it is expected that ALL cross-sells are controlled by the refresh process which will clear and repopulate them	

Setting: : ERP_ShipmentRefreshSendsConfirmations	**Valid Values:** TRUE/FALSE
Where Used: ShipmentRefresh	
Purpose/Notes: This option kicks off sending the shipment confirmation email from the integration service when a shipment is refreshed	

Setting: : ERP_ShowDetailedODBCException	**Valid Values:** TRUE/FALSE
Where Used: SolomonIntegration, SXeIntegration	
Purpose/Notes: When an error occurs on an ODBC-based ERP connection then record the specific error detail in the application log	

Setting: : ERP_ShowDetailedSQLExceptions	**Valid Values:** TRUE/FALSE
Where Used: SolomonIntegration	
Purpose/Notes: When an error occurs and set to true, show the details errors in the application log	

Setting: : ERP_SubmitCredentials	**Valid Values:** TRUE/FALSE
Where Used: RefreshBase	
Purpose/Notes: If set to true then retrieve and set the user and password information	
Setting: ERP_StatusCustomerNumber	**Valid Values:**

Where Used: Epicor9Integration, VantageIntegration
Purpose/Notes: Used for when an order is placed and is always set to a specific customer in the ERP and only creates a new ship to record. This is used for B2C sites so that new customers are not created in the ERP for each new user.

Setting: : ERP_SubmitOrders	**Valid Values:** TRUE/FALSE
Where Used: CustomerOrderRefresh	
Purpose/Notes: Indicates if the integration service should be used to submit the orders into the ERP system. If orders are submitted in batch only then the CustomerOrderRefresh process controls. There is also a setting OnLineOrderSubmission which indicates that the order is submitted immediately and does not require the batch process.	

Setting: : ERP_System	**Valid Values:** Epicor9 Mas200 Navision Solomon SXe SyteLine6 SyteLine7 Vantage Visual
Where Used: Management Console	
Purpose/Notes: This is used for OEM branding of the management console and for other potential switches based specifically on the back end ERP system. It is also used as the basis for which integration service to use so it must match the appropriately named integration service (i.e. Epicor9, not just Epicor).	

Setting: ERP_SXAPI_AppServer	**Valid Values:**
Where Used: SXeAPIIntegration	
Purpose/Notes: Denotes the name of the application server for the API	

Setting: ERP_SX_InvoiceTypesToRefresh	**Valid Values:**
Where Used: SXeAPIIntegration	
Purpose/Notes: Indicates the specific record types to pull in on invoice refresh	

Setting: ERP_SXAPI_UserIdr	**Valid Values:**
Where Used: SXeAPIIntegration	

Purpose/Notes: Denotes the user id to use to connect to the API

Setting: ERP_SXAPI_Password	**Valid Values:**
Where Used: SXeAPIIntegration	
Purpose/Notes: Denotes the password to use to connect to the API	

Setting: ERP_TrackInventory	**Valid Values:** TRUE/FALSE
Where Used: Epicor9Integration, VantageIntegration	
Purpose/Notes: Used to set the TrackInventory flag in the product table	

Setting: : ERP_UpdateInvoices	**Valid Values:** TRUE/FALSE
Where Used: InvoiceRefresh	
Purpose/Notes: Indicates if the invoice refresh should be run. During the invoice refresh, if this is set to false and the invoice already exists, it will not be overwritten but if this option is set to true, then the invoice data will be resynched to the dataset retrieved. Normally this ought to be set to false since the invoice data in the ERP should not change once established.	

Setting: ERP_UseAlternativeTaxCodeForOrderSubmit	**Valid Values:** TRUE/FALSE
Where Used: Syteline7Integration	
Purpose/Notes If true will not set taxcode1, but instead set frt_tax_code1 and msc_tax_code1. Used for VAT tax.	

Setting: ERP_UseProductRefreshStoredProcedure	**Valid Values:** TRUE/FALSE
Where Used: ProductRefresh_Generic	
Purpose/Notes When turned on, the system will call a stored procedure called RefreshProducts passing in the XML dataset rather than processing each record through the model	

Setting: ERP_UseCustomerRefreshStoredProcedure	**Valid Values:** TRUE/FALSE
Where Used: CustomerRefresh_Generic	
Purpose/Notes: When turned on, the system will call a stored procedure name RefreshCustomers passing in the XML dataset rather than processing each record through the model	
Setting: ERP_UseRealTimeOrderHistory	**Valid Values:** TRUE/FALSE
Where Used: NOT IMPLEMENTED STANDARD YET	
Purpose/Notes: When this setting is true, then a real-time call will be made to retrieve the order history list and subsequent order history detail	

Setting: ERP_UseRealTimeInvoiceHistory	**Valid Values:** TRUE/FALSE
Where Used: NOT IMPLEMENTED STANDARD YET	
Purpose/Notes: When this setting is true, then a real-time call will be made to retrieve invoice history information	

Setting: ERP_UserName	**Valid Values:**
Where Used: RefreshBase	
Purpose/Notes: Used with ERP_SubmitCredentials	

Setting: ERP_UseStaticCustomer	**Valid Values:** TRUE/FALSE
Where Used: Epicor9Integration, VantageIntegration	
Purpose/Notes: Used to determine if the static customer should be used	

Setting: ERP_UseWindowsService	**Valid Values:** TRUE/FALSE
Where Used: WebSite, RefreshBase	
Purpose/Notes: Determines if the older (web to ERP) web service is used or the new integration service (ERP to web) service is used. This value should be TRUE unless there is a specific reason the user is using the older implementation methodology.	

Setting: ERP_Version	**Valid Values:**
Where Used: unused	
Purpose/Notes: The intent for the setting is to set the specific version of an ERP for slight variations in the manner in which integration or other functionality is exposed	

Setting: ERP_Warehouse	**Valid Values:**
Where Used: Epicor9Integration, VantageIntegration	
Purpose/Notes: This is used to determine the warehouse from which to obtain inventory balances and for pricing	

Setting: OnLineOrderSubmission	**Valid Values:** TRUE/FALSE
Where Used: CustomerOrderProcessor	
Purpose/Notes: Flag to indicate if orders are submitted to the web service as the order is placed. This is the 'old' version of having the web service on the ERP server instead of on the Web server which is the current methodology so this option is basically obsolete.	

Setting: RealTimeCallsEnabled	**Valid Values:** TRUE/FALSE
Where Used: ScheduledTask	
Purpose/Notes: Certain tasks can be set as real-time such as an inventory inquiry. These are set into a separate queue from the integration service on a separate thread/set of threads so that they will always be processed with higher priority than other scheduled tasks.	

Setting: RealTimeWaitSeconds	**Valid Values:** integer
Where Used: ScheduledTask	
Purpose/Notes: This is the wait time value that a real time call should wait before it times out.	

Setting: RefreshPageSize	**Valid Values:** integer
Where Used: RefreshBase	
Purpose/Notes: This is the nHibernate page size to use before flushing cache to help keep the datasets small when submitting into the ISC database	

Setting: RefreshPreLoadCustomers	**Valid Values:** TRUE/FALSE
Where Used: CustomerRefresh, RefreshBase	
Purpose/Notes: This option is used to pre-load all the customers in the ISC database to compare against customers coming from the ERP. If the list is relatively small (< 500) this will make the refresh go faster.	

Setting: RefreshPreLoadProducts	**Valid Values:** TRUE/FALSE
Where Used: ProductRefresh_Generic, RefreshBase	
Purpose/Notes: Preloads the products in the ISC database to compare against products coming from the ERP. If the list is relatively small (< 500) this will make the refresh go faster.	

Setting: RefreshPurgeOrders - DEPRECATE	**Valid Values:** TRUE/FALSE
Where Used: CustomerOrderRefresh	
Purpose/Notes: This is no longer used	

Setting: SaveERPDataSet	**Valid Values:** TRUE/FALSE
Where Used: RefreshBase	
Purpose/Notes: This option is used for debugging and requires the integration service to call the GetERPDataSet function which will initially get the dataset and save it as a binary dataset and on subsequent calls re-use the same dataset. It is not part of a production environment. This version of the dataset will save in the location defined by StoredDataSetPath. This set is saved as a .dat file to keep it binary.	

Setting: SaveISCCustomerProducts	**Valid Values:** TRUE/FALSE
Where Used: CustomerRefresh	
Purpose/Notes: This feature allows for the preservation of customer product data during customer refresh if the flag is set to FALSE then the data is refreshed from the ERP. If set to TRUE, then the customer product table will only be created if there is not already a record for the specific customer/product combination.	

Setting: SaveXML	**Valid Values:** TRUE/FALSE
Where Used: ShipmentRefresh	
Purpose/Notes: This is a point option to save the ShipmentRefresh XML directly into the application log	

Setting: StoredDataSetPath	**Valid Values:**
Where Used: RefreshBase	
Purpose/Notes: Indicates the location of the stored dataset associated with the SaveERPData setting	

Setting: StoreEncryptedCC	**Valid Values:** TRUE/FALSE
Where Used: CreditCardTransaction, BatchProcessor	
Purpose/Notes: If this option is turned on, then the credit card number is stored in the database encrypted using standard .NET encryption. This should be coupled with the DaysToStoredEncryptedCC setting to ensure that once the data is moved to the ERP system (which is presumably the only reason we would ever store the data) it is removed. In the batch processor, this option mus be on for the PurgeEncryptedNumbers to be run	

Setting: SubmitAllPaymentInfo	**Valid Values:** TRUE/FALSE

Where Used: Epicor9Integration, VantageIntegration
Purpose/Notes: If not set to true and credit card funds are captured on the site, then an error is thrown

Setting: SubmitReturnsOrderNumber - DEPRECATE	**Valid Values:** TRUE/FALSE
Where Used: CustomerOrderRefresh	
Purpose/Notes: If this option is turned on, then during the order submission process, a result is expected to be returned from the ERP integration. Then the idea is to have the original web order # be placed into the CustomerPO field and the ERP-returned order # placed into the Order Number. This should have been deprecated since we now have an ERPOrderNumber field that should be used to store the returning ERP order #.	

Setting: SubmitSaleTransaction	**Valid Values:** TRUE/FALSE
Where Used: CustomerOrderProcessor, Epicor9Integration, VantageIntegration	
Purpose/Notes: This flag indicates if we are capturing the funds at the time of the order rather than just an authorization. This will trigger a payment to be submitted to the ERP, not just the order.	

Setting: SubmitToReviewStatus	**Valid Values:** TRUE/FALSE
Where Used: IntegrationService, CustomerOrderRefresh	
Purpose/Notes: In the integration service, this idea is to set the status of the order to either 'Processing' or 'Review' once we get confirmation from the integration service that it was been successfully submitted to the ERP. If this option is set to false, the value would be 'Processing'. In the customer order refresh process, if the order is submitted via a web service, this flag is used to indicate if a successful submission should change the order status to processing or review.	