Upgrade Guide

CommerceDriver™ for *Windows*®
Version 2.27 to 2.30

CommerceDriver™ Upgrade Guide

EVO Snap* has release the v2.30 version of CommerceDriver™. This guide is designed to assist partners in the migration process from integrations using v2.27 to v2.30 of CommerceDriver™.

Partners who were previously using the 2.27 libraries and who have been directed by their Snap* Solutions Engineer to update their projects to version 2.30 will need to upgrade to the 2.30 version of CommerceDriver™.

Enhancements & New Features

Terminal Service Management (TSM)

All CommerceDriver™ integrators upgrading to version 2.30 are required to support the new TSM features in order to support updates to the Ingenico line of terminals, including the iCMP, iPP320, and iPP350 terminals.

Terminals will receive updates periodically, and if an update is not applied to a terminal by the associated deadline date, the terminal will be unable to transact until the update is installed.

For more information on the new TSM feature, please see the TSM User Guide.

Upgrade from .Net 4.5 to .Net 4.6

CommerceDriver upgraded from .Net 4.5 to .Net 4.6. This upgrade should not affect integration and should not require any changes on behalf of integrators.

EVOCommerceDriverAPI

Constructor

The construction of the `CommerceDriverController` in v2.27 had no parameters, and instead read values from the `App.Config` file. In v2.30, the constructor has been changed to take the service key and the application profile ID so that it no longer relies on the `App.Config` file. The original constructor is still available, but has been marked as obsolete. The new constructor will continue to read values from the `App.Config` file if no service key or application profile ID is passed in.

Version	Code Change
2.27	`CommerceDriverController();`
2.30	`CommerceDriverController(string serviceKey, string appProfileId);`

Events

Obsolete Events

Please note that the following methods have been marked as obsolete and ***should no longer be used***:

Assign Question	`public event AssignQuestionsEventHandler AssignQuestions;`
Change Password	`public event ChangePasswordEventHandler ChangePassword;`
Forgot Password	`public event ForgotPasswordEventHandler ForgotPassword;`
Identity Login	`public event IdentityLoginEventHandler IdentityLogin;`
Password Reset	`public event PasswordResetEventHandler PasswordReset;`

New Events

These events have been added and should be subscribed to:

Response Returned

This event is called at the end of a transaction with all the response information from the transaction.

```
public event ResponseReturnedEventHandler ResponseReturned;
```

Retrieve CVV

This event is called when running an AMEX MSR transaction, which requires the CVV number on the card.

```
public event RetrieveCvvEventHandler RetrieveCvv;
```

Selected Terminal Changed

This event is called when the selected terminal has changed.

```
public event SelectedTerminalChangedEventHandler SelectedTerminalChanged;
```

Properties

Obsolete Properties

Please note that these properties have been marked as obsolete and ***should no longer be used***:

* `public ManufacturerDefinition Manufacturer { get; set; }`
* `public List<ManufacturerInfo> Manufacturers { get; }`
* `public List<ManufacturerDefinition> ManufacturerDefinitions { get; private set; }`
* `public List<Assembly> RegisteredAssemblies { get; private set; }`
* `public TerminalDefinition Terminal { get; set; }`
* `public TerminalSettings TerminalSettings { get; private set; }`
* `public string TerminalSummary { get; }`
* `public ITransactionHandler TransactionHandler { get; set; }`

IsBankcardsSupported and IsGiftCardsSupported

These properties have been replaced by `SelectedMerchantProfile.ServiceType`:

* `public bool IsBankcardsSupported { get; }`
* `public bool IsGiftcardsSupported { get; }`

Service Descriptors

This property has been replaced by `MerchantProfiles`:

* `public List<ServiceDescriptor> ServiceDescriptors { get; }`

New Properties

These properties have been added and should be used:

Is QPS Supported

This property is true if Quick Payment Service is supported:

```
public bool IsQPSSupported { get; }
```

Default Merchant Profile ID

When logging in, this Merchant Profile ID will be set as the Selected Merchant Profile if it is available:

```
public string DefaultMerchantProfileId { get; set; }
```

Merchant Profiles

This property provides a dictionary of available Merchant Profile ID and Merchant Profile pairs that are associated with the logged in user:

```
public Dictionary<string, MerchantProfile> MerchantProfiles { get; }
```

Selected Merchant Profile

This is the current Merchant Profile that is being used when processing transactions:

```
public MerchantProfile SelectedMerchantProfile { get; }
```

Selected Terminal

This is the current terminal that is being used when processing transactions:

```
public ITerminalController SelectedTerminal { get; private set; }
```

Methods

New Methods

Download and Apply Update

This method downloads and applies an update to the terminal and can be called at any time, meaning merchants do not need to wait for an update to be available in order to make this call:

Please see the TSM User Guide for more information on downloading and applying terminal updates.

```
public async Task<DownloadAndApplyResult> DownloadAndApplyUpdate(Action<ProgressMessage>
onProgressMessage)
```

Create Receipt

This method creates a receipt object with all the information needed to create a customized receipt:

```
public Receipt CreateReceipt(TransactionResult result, TransactionResponse response,
TransactionRequest transactionData)
```

Email Receipt

This method emails a receipt of the specified transaction to the specified email address:

```
public bool EmailReceipt(string transactionId, string emailAddress)
```

Set QPS Supported

This method is used to set the `SelectedMerchantProfile` to support Quick Payment Service. Note that `SelectedMerchantProfile` must be for Mexico for QPS to be supported:

```
public void SetQPSSupported(bool enabled)
```

Set Selected Merchant Profile

This method sets the `SelectedMerchantProfile` based on the provided Merchant Profile Id:

```
public bool SetSelectedMerchantProfile(string id)
```

User Account Methods

These methods replaced older methods, or were created in order to simplify and add functionality around user accounts. Most of these methods have both a synchronous and an asynchronous call, so the method can be called according to the user's threading needs.

Login

Version	Code Change
2.27	`public bool PerformIdentityLogin()`
2.30	`public GatewaySession Login(string userName, string password)`
	`public async Task<GatewaySession> LoginAsync(string userName, string password)`

Logout

Version	Code Change
2.27	`public void PerformLogout()`
2.30	`public void Logout()`

Get User Security Questions

Version	Code Change
2.27	`public bool PerformSecurityQuestions(string userName, string password)`

2.30	`public List<SecurityQuestion> GetUserSecurityQuestions()`
	`public async Task<List<SecurityQuestion>> GetUserSecurityQuestionsAsync()`

Forgot User Password

Version	Code Change
2.27	`public bool PerformForgotPassword()`
2.30	`public SecurityQuestionToken ForgotUserPassword(string userName)`
	`public async Task<SecurityQuestionToken> ForgotUserPasswordAsync(string userName)`

Reset User Password

This method resets a user's password after a question has been answered from `ForgotUserPassword()`. An email will be sent to the user with a new temporary password.

Version	Code Change
2.27	`public bool ResetUserPassword(string userName, string token, string text)`
2.30	`public async Task<bool> ResetUserPasswordAsync(string userName, string token, string text)`

Change User Password

Version	Code Change
2.27	`public  bool ChangeUserPassword(string email, string userName, string newPassword, string oldPassword)`
2.30	`public async Task<bool> ChangeUserPasswordAsync(string email, string userName, string newPassword, string oldPassword)`

Get User Password Expiration

Version	Code Change
2.27	`public DateTime? GetUserPasswordExpiration(string userName, string password)`
2.30	`public async Task<DateTime?> GetUserPasswordExpirationAsync(string userName, string password)`

Get Available Questions

Version	Code Change
2.27	`public List<SecurityQuestion> GetAvailableSecurityQuestions()`
2.30	`public async Task<List<SecurityQuestion>> GetAvailableSecurityQuestionsAsync()`

Update User Security Questions

Version	Code Change
2.27	`public bool UpdateUserSecurityQuestions(string userName, string password, List<SecurityAnswer> securityAnswers)`
2.30	`public async Task<bool> UpdateUserSecurityQuestionsAsync(string userName, string password, List<SecurityAnswer> securityAnswers)`

Terminal Management

These methods were created to simplify terminal management and allow multiple terminals to be added, selected, used, and removed.

Add a Terminal

In order to add a new terminal, one of the methods in the `IngenicoTerminalControllerFactoryClass` will need to be called to create the `ITerminalController` object.

Version	Code Change
2.27	`public void SetTerminal(string manufacturerName, string terminalName, string terminalConfigName)`
2.30	`public bool AddTerminal(ITerminalController terminal)`

Select Terminal

In v2.27, there was no capability to have multiple terminals. Since v2.30 allows for multiple terminals, a new terminal will need to be added using `AddTerminal(ITerminalController terminal)` and then selected in order to be used.

Version	Code Change
2.30	`public bool SelectTerminal(string id)`

Initialize Terminal

The InitializeTerminal method of the Commerce Driver object now provides information if an update is available for the terminal currently in use. Users must be signed on to their instance of CommerceDriver™ in order for the initialize terminal process to begin and for the terminal to begin checking for updates. This sign on procedure can be found for each operating system in their respective Quick Start Guides. After performing the steps to authenticate and add a terminal, check the response from the `InitializeSelectedTerminal()` method to determine if updates are available.

IMPORTANT! If a terminal has not downloaded the available terminal updates by the associated deadline date, the terminal will be deactivated, preventing any future transactions.

Version	Code Change
2.27	`public async Task<bool> InitializeTerminal()`
2.30	`public async Task<InitializeTerminalResult> InitializeSelectedTerminal()`

Get Terminal by ID

This method will return the terminal object associated with the terminal ID.

Version	Code Change
2.30	`public ITerminalController GetTerminalById(string id)`

Get Terminals

This method will return a list of all the terminals that have been added.

Version	Code Change
2.30	`public List<ITerminalController> GetTerminals()`

Disconnect Terminal

This method un-initializes the selected terminal.

Version	Code Change
2.30	`public void DisconnectTerminal()`

Remove Terminal

This method removes the terminal from the list of added terminals. If the terminal being removed is the current `SelectedtTerminal`, then the next terminal in the list will be set as the `SelectedTerminal`.

Version	Code Change
2.30	`public bool RemoveTerminal(string id)`

Check Terminal Connection

Version	Code Change
2.30	`public void CheckTerminalConnection()`

Check Terminal Battery Level

This method has replaced the `ProcessAsync()` method that was used to check the battery level in v2.27.

Version	Code Change
2.27	`public Task ProcessAsync(BatteryStatusOperationRequest request)`
2.30	`public void CheckTerminalBatteryLevel()`

Start Terminal Transaction

This method can be used to start a terminal transaction with the selected terminal. The `ProcessAync()` methods that use terminals to run a transaction can also be used.

Version	Code Change
2.30	`public void StartTerminalTransaction(TransactionRequest request)`

Cancel Terminal Transaction

Version	Code Change
2.27	`public void CancelTransaction()`
2.30	`public void CancelTerminalTransaction()`

Print Receipt

This method will print a receipt if the terminal supports print receipting.

Version	Code Change
2.30	`public PrintStatus PrintReceipt(Receipt receipt, bool isMerchantCopy)`

ProcessAsync Methods

Battery Status

Version	Code Change
2.27	`public Task ProcessAsync(BatteryStatusOperationRequest request)`
2.30	`public void CheckTerminalBatteryLevel()`

Removed Methods

All the `StoredValue` methods have been removed in v2.30 because `StoredValue` cards are not currently supported by CommerceDriver™.

`public Task ProcessAsync(StoredValueRedeemOperationRequest request)`
`public Task ProcessAsync(StoredValueAuthorizeCaptureOperationRequest request)`
`public Task ProcessAsync(StoredValueAuthorizeOperationRequest request)`
`public Task ProcessAsync(StoredValueManageAccountActivateOperationRequest request)`
`public Task ProcessAsync(StoredValueManageAccountBalanceTransferOperationRequest request)`
`public Task ProcessAsync(StoredValueManageAccountDeactivateOperationRequest request)`
`public Task ProcessAsync(StoredValueManageAccountReloadOperationRequest request)`
`public Task ProcessAsync(StoredValueManageAccountByIDReloadOperationRequest request)`

```
public Task ProcessAsync(StoredValueUndoOperationRequest request)
```

The `ReadMSROperationRequest` method also has been removed and is **no longer supported:**

```
public Task ProcessAsync(ReadMsrOperationRequest request)
```

Verify

Verify is a transaction operation added to the CommerceDriver™ which can be used to validate a card. The Verify operation will cause the terminal to prompt for a MSR card swipe.

CommerceDriverController Method Call

```
Task ProcessAsync(VerifyOperationRequest request)
```

VerifyOperationRequest

```
public class VerifyOperationRequest: TransactionRequest
    {
        #region Constructors

        public VerifyOperationRequest()

        public VerifyOperationRequest(TransactionRequest request) : base(request)

        #endregion

        #region Properties

        public ProcessOperation Operation { get; }

        #endregion

        #region Methods

        public void AddTransactionRequest(TransactionRequest request)

        public string ToString()

        #endregion
    }
```

Tokenization

Tokenization is the process of using a token to run what would typically be a card only transaction. The EVO Snap* platform generates a unique token associated with a customer's card that can be used instead of the customer's actual card to process a transaction.

How to Run a Tokenized Transaction

In order to run a tokenized transaction, the `PaymentAccountDataToken` property must be populated with a valid payment token in the `TransactionRequest.CardData` field. If the `PaymentAccountDataToken` is populated, then CommerceDriver™ will automatically run the token and no card will be needed to process the transaction.

The transaction types that can use payment tokens are listed below:

* Authorize
* Authorize and Capture
* Return Unlinked

Where to Find the Payment Token

In the response of any transaction that uses a card, there will be a field called `PaymentAccountDataToken`. The value in this field is specific to the card that was used in that transaction and can be used to populate `TransactionRequest.CardData.PaymentAccountData` in order to run another transaction for that card.

Transactions that can return a `PaymentAccountDataToken` are listed below:

* Authorize
* Authorize and Capture
* Return Unlinked
* Verify