Upgrade Guide

CommerceDriver™ for *Android*™
Version 2.27 to 2.30

CommerceDriver™ Upgrade Guide

EVO Snap* has released 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.

Tokenized Transactions

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. For more information on the new Tokenization feature, please see the Tokenization section below.

Deprecated Methods

CommerceDriver.java (Constructor)

Context has been removed from the CommerceDriver™ constuctor, as the service now handles and passes around application context internally via the chain of responsibility pattern. This is intended to allow UI Android context switching without invalidating the CommerceDriver™ state or dropping or mishandling callbacks.

Version	Code Change
2.27	`public CommerceDriver(Context context, String applicationProfileId, String serviceKey) { this.platform = new RetrofitPlatform(context, applicationProfileId, serviceKey); }`
2.30	`public CommerceDriver(String applicationProfileId, String serviceKey) { this.sessionClient = new RealSessionClient(serviceKey); this.terminalClient = new RealTerminalClient(sessionClient); this.queryClient = new RealQueryClient(sessionClient); this.transactionClient = new RealTransactionClient(sessionClient, terminalClient, applicationProfileId); }`

CommerceDriver.java (Terminal)

Add Terminal

Version 2.30 changes the parameters for `addTerminal`, introducing the new `terminalController` object, as well as the controller based paradigms. `terminalController` will return "True" if the terminal was successfully added and returns "False" if otherwise.

Version	Code Change
2.27	`public final boolean addTerminal(String id, Terminal terminal) { Timber.tag(TAG).i("Adding Terminal"); return terminalManager.addTerminal(id, terminal); }`
2.30	`public boolean addTerminal(TerminalController terminalController) { Timber.tag(BuildConfig.TAG).d("[%s] addTerminalController()", TAG); return terminalClient.addTerminalController(terminalController); }`

Remove Terminal

In version 2.30, `removeTerminal` was updated to `removeTerminalById`, which is the ID of the terminal being used in the transaction and will return "True" if the terminal was successfully removed.

Version	Code Change
2.27	`public final boolean removeTerminal(String id) { Timber.tag(TAG).i("Removing Terminal"); Terminal terminal = terminalManager.getTerminalById(id); if (terminal != null) { terminal.shutdown(); } return terminalManager.removeTerminal(id); }`
2.30	`public boolean removeTerminalById(String id) { Timber.tag(BuildConfig.TAG).d("[%s] removeTerminalControllerById()", TAG); return terminalClient.removeTerminalControllerById(id); }`

Select Terminal

In version 2.30, `setActiveTerminal` was updated to `selectTerminal`. This call selects a terminal that has been previously added and returns "True" if the terminal was successfully selected.

Version	Code Change
2.27	`public final boolean setActiveTerminal(String id) { Timber.tag(TAG).i("Selecting Terminal"); return terminalManager.setTerminal(id); }`
2.30	`public boolean selectTerminal(String id) { Timber.tag(BuildConfig.TAG).d("[%s] selectTerminal()", TAG); Timber.tag(BuildConfig.TAG).v("[%s] selectTerminal(%s)", TAG, id); return terminalClient.selectTerminal(id); }`

Get Selected Terminal ID

In version 2.30, `getActiveTerminalId` changed to `getSelectedTerminalId`, which returns the selected terminal's ID or "null" if not available.

Version	Code Change
2.27	`public final String getActiveTerminalId() { return terminalManager.getActiveTerminalId(); }`
2.30	`public String getSelectedTerminalId() { Timber.tag(BuildConfig.TAG).d("[%s] getSelectedTerminalId()", TAG); return terminalClient.getSelectedTerminalId(); }`

Get All Added Terminal IDs

In version 2.30, `getTerminalIds` changed to `getAddedTerminalIds`, which returns a list of added terminal IDs or an empty list if not available.

Version	Code Change
2.27	`public final List<String> getTerminalIds() { return terminalManager.getTerminalIds(); }`
2.30	`public List<String> getAddedTerminalIds() { Timber.tag(BuildConfig.TAG).d("[%s] getAddedTerminalIds()", TAG); return terminalClient.getAddedTerminalControllerIds(); }`

Initialize Terminal

CommerceDriver™ handles its receivers internally in version 2.30. The parameter method override is now deprecated. The code sample below shows the new method to attempt to connect to a selected terminal.

Version	Code Change
2.27	`public void initializeTerminal(InitializeTerminalReceiver receiver) { initializeTerminal(new HashMap<String, String>(), receiver); } public void initializeTerminal(Map<String, String> params, InitializeTerminalReceiver receiver) { Timber.tag(TAG).i("Initializing Terminal"); Terminal terminal = terminalManager.getTerminal(); if (terminal != null) { terminal.initialize(params, receiver); } }`
2.30	`public ConnectResponse initializeTerminal() { Timber.tag(BuildConfig.TAG).d("[%s] initializeTerminal()", TAG); return connectTerminal(); } public ConnectResponse connectTerminal() { Timber.tag(BuildConfig.TAG).d("[%s] connectTerminal()", TAG); return terminalClient.connectTerminal(); }`

Check Initialization/Connection

`checkTerminalConnectivity` is no longer supported in version 2.30, and terminal initialization and connection have been set as internal CommerceDriver™ flow functions. Instead, checking initialized will invoke `isTerminalConnected`, which will return a Boolean value based on the connection status derived from a connection response.

Version	Code Change
2.27	`public synchronized final void checkTerminalConnectivity(ConnectivityStatusReceiver receiver) { Timber.tag(TAG).i("Checking Connectivity"); Terminal terminal = terminalManager.getTerminal(); if (terminal == null) { Timber.tag(TAG).w("No Active Terminal Controller"); if (receiver != null) {`

	receiver.sendConnectivityResult (ConnectivityStatusReceiver.ConnectivityResult.TERMINAL_MISSING); } return; } Timber.tag(TAG).v("Sending Terminal Request To Active Terminal Controller"); terminal.checkConnectivity(receiver); } public boolean isTerminalInitialized() { Terminal terminal = terminalManager.getTerminal(); return terminal != null && terminal.isInitialized(); }
2.30	public boolean isTerminalInitialized() { Timber.tag(BuildConfig.TAG).d("[%s] isTerminalInitialized()", TAG); return isTerminalConnected(); } public boolean isTerminalConnected() { Timber.tag(BuildConfig.TAG).d("[%s] isTerminalConnected()", TAG); return terminalClient.isTerminalConnected(); }

Reset/Clear Out Connection

`resetTerminal` is no longer supported in version 2.30. Instead the new commands for disconnecting a terminal and shutting down a terminal, respectively, are provided. `Reset` is now an internal CommerceDriver™ operation.

Version	Code Change
2.27	public void resetTerminal() { Timber.tag(TAG).i("Resetting Terminal"); Terminal terminal = terminalManager.getTerminal(); if (terminal != null) { terminal.reset(); } }
2.30	public DisconnectResponse disconnectTerminal() { Timber.tag(BuildConfig.TAG).d("[%s] connectTerminal()", TAG); return terminalClient.disconnectTerminal(); }

```
public ShutdownResponse shutdownTerminal() {
      Timber.tag(BuildConfig.TAG).d("[%s] shutdownTerminal()", TAG);
      return terminalClient.shutdownTerminal();
   }
```

Terminal Response Objects

Below are some of the response objects that have changed since the 2.27 version of CommerceDriver™:

Disconnected Response

2.30

```
public interface DisconnectResponse {
    Result getResult();
    String getErrorMessage();
    enum Result {
        SUCCESS,
        ERROR, INVALID_TERMINAL_ID,
    }
}
```

Connect Response

2.30

```
public interface ConnectResponse {
    Result getResult();
    String getErrorMessage();
    List<AvailableUpdateInfo> getUpdates();
    boolean isUpdateAvailable();
    enum Result {
        CONNECTED,
        INVALID_TERMINAL_ID,
        SESSION_REQUIRED,
        TERMINAL_ERROR,
    }
}
```

Battery Information Response

2.30

```
public interface BatteryInfoResponse {
    String getBatteryPercentage();

    String getErrorMessage();

    Result getResult();
```

	enum Result { SUCCESS, INVALID_TERMINAL_ID, ERROR, NOT_CONNECTED, } }

Terminal Reboot Response

2.30	public interface RebootResponse { Result getResult(); String getErrorMessage(); enum Result { SUCCESS, ERROR, INVALID_TERMINAL_ID, } }

Terminal Shut Down Response

2.30	public interface ShutdownResponse { Result getResult(); String getErrorMessage(); enum Result { SUCCESS, ERROR, INVALID_TERMINAL_ID, } }

Check Battery

`checkBatteryStatus` will now return a `BatteryInfoResponse` object. As with other method changes around terminal, integrators no longer have to supply a receiver in the method parameter.

Version	Code Change
2.27	public synchronized final void checkTerminalBatteryLevel (BatteryLevelReceiver receiver) { Timber.tag(TAG).i("Checking Battery");

	`` Terminal terminal = terminalManager.getTerminal(); if (terminal == null) { Timber.tag(TAG).w("No Active Terminal Controller"); if (receiver != null) { receiver.sendBatteryStatusError (BatteryLevelReceiver.BatteryStatusError.TERMINAL_MISSING); } return; } Timber.tag(TAG).v("Sending Terminal Request To Active Terminal Controller"); terminal.checkBatteryStatus(receiver); } ```
2.30	``` public BatteryInfoResponse checkBatteryStatus() { Timber.tag(BuildConfig.TAG).d("[%s] checkBatteryStatus()", TAG); return terminalClient.checkBatteryStatus(); } ```

CommerceDriver.java (Transactions)

Start Transaction

In version 2.30, `startTransaction` changed to `startTerminalTransaction`. The new `startTerminalTransaction` now takes a `PosRequest` object as its parameter instead of a `POSBankCardRequest`.

Version	Code Change
2.27	``` public synchronized final void startTransaction(POSBankCardRequest request) { Timber.tag(TAG).i("Starting Transaction"); Terminal terminal = terminalManager.getTerminal(); MerchantProfile merchant = platform.getBankCardMerchantProfile(); BankCardService service = platform.getBankCardPaymentService(); // create transaction processor with factory method. // the transaction processor will expose the platform request/reversal BankCardTransactionProcessor bankCardProcessor = BankCardTransactionProcessor.newInstance(request.getOperation(), merchant.getProfileId(), service.getServiceId(), platform.getTPSApiProvider()); // create terminal request TerminalBankCardRequest bankCardRequest = new TerminalBankCardRequest(request); ```

```
        // set currency code from merchant profile
        bankCardRequest.setCurrencyCode(merchant.getCurrencyCode());
        // set txn specific data
        bankCardRequest.setMerchantProfileData(merchant);
        bankCardRequest.setPaymentServiceData(service);
        bankCardRequest.setTransactionProcessor(bankCardProcessor);
        // common txn data
        bankCardRequest.setTransactionDateTime(new Date());
        bankCardRequest.setCustomerPresence(CustomerPresence.PRESENT);
        bankCardRequest.setCardPresent(true);

        try {
            // validate session
            if (!platform.hasSession()) {
                throw new Exception("Session Invalid");
            }
            // validate terminal
            if (terminal == null) {
                throw new Exception("No Terminal Set");
            }
            Timber.tag(TAG).v("Validating Request = %s", bankCardRequest);
            // validate merchant/service ok
            MerchantServiceValidator merchantServiceValidator = new
MerchantServiceValidator();
            merchantServiceValidator.validateRequest(bankCardRequest);
            // validate txn amounts
            TransactionAmountsValidator transactionAmountsValidator = new
TransactionAmountsValidator();
            transactionAmountsValidator.validateRequest(bankCardRequest);
            // pin debit validator
            PinDebitValidator pinDebitValidator = new PinDebitValidator(service,
terminal);
            pinDebitValidator.validateRequest(bankCardRequest);
        } catch (Exception e) {
            Timber.tag(TAG).e(e,"Error Occurred = %s", e.getMessage());
            BankCardTransactionStatusUpdateReceiver statusUpdateReceiver =
request.getTransactionStatusUpdateReceiver();
            if (statusUpdateReceiver != null) {

statusUpdateReceiver.sendTransactionStatusUpdate(BankCardTransactionStatusUpdateRec
eiver.Status.VALIDATION_FAILED);
            }
            BankCardTransactionCompletedReceiver completedReceiver =
request.getTransactionCompletedReceiver();
            if (completedReceiver != null) {
                BankCardTransactionDataPro transactionDataPro =
bankCardRequest.getTransaction().getTransactionData();

completedReceiver.sendTransactionCompletedMessage(TransactionResult.TRANSACTION_ERR
OR, transactionDataPro, null);
            }
            return;
        }
```

	Timber.tag(TAG).v("Sending Bank Card Request To Terminal"); // create interface to wrap the BankCardRequest terminal.startTransaction(bankCardRequest); }
2.30	public void startTerminalTransaction(PosRequest posRequest) throws SnapValidationError { Timber.tag(BuildConfig.TAG).d("[%s] startTerminalTransaction()", TAG); transactionClient.startTerminalTransaction(posRequest); }

Capture

In version 2.30, `captureTransaction` changed to capture, and the returned object changed from a `BankCardCaptureResponse` to a `BankCardCaptureResponsePro` object. This new object has a lot of new added fields along with the relevant getters and setters.

Version	Code Change
2.27	public final BankCardCaptureResponse captureTransaction(BankCardCapturePro differenceData) { Timber.tag(TAG).i("Capturing Transaction"); return platform.requestBankCardCapture(differenceData); }
2.30	public BankCardCaptureResponsePro capture(BankCardCapturePro differenceData) throws IOException { Timber.tag(BuildConfig.TAG).d("[%s] capture()", TAG); return transactionClient.capture(differenceData); }

Undo

In version 2.30, `undoTransaction` changed to undo, and the return object changed from a `BankCardTransactionResponse` to a `BankCardTransactionResponsePro` object. The new response object has a few new fields, along with some removed methods to operate on the object data.

Version	Code Change

2.27	`public final BankCardTransactionResponse undoTransaction(BankCardUndo differenceData) { Timber.tag(TAG).i("Voiding Transaction"); return platform.requestBankCardUndo(differenceData); }`
2.30	`public BankCardTransactionResponsePro undo(BankCardUndo differenceData) throws IOException { Timber.tag(BuildConfig.TAG).d("[%s] undo()", TAG); return transactionClient.undo(differenceData); }`

Return by ID

In version 2.30, `returnTransactionById` changed to `returnById`, along with changes to both the returned response object and the difference data parameter.

Version	Code Change
2.27	`public final BankCardTransactionResponse returnTransactionById(BankCardReturn differenceData) { Timber.tag(TAG).i("Refunding Transaction"); return platform.requestBankCardReturnById(differenceData); }`
2.30	`public BankCardTransactionResponsePro returnById(BankCardReturnPro differenceData) throws IOException { Timber.tag(BuildConfig.TAG).d("[%s] returnById()", TAG); return transactionClient.returnById(differenceData); }`

Email Receipt

In version 2.30, `emailReceipt` changed to `sendReceipt`, and the response is no longer a generic `ApiResponse` object.

Version	Code Change
2.27	`public final ApiResponse emailReceipt(String email, String transactionId) { Timber.tag(TAG).i("Sending Receipt"); return platform.requestEmailReceipt(email, transactionId);`

	}
2.30	public SendReceiptResponse **sendReceipt**(String email, String transactionId) throws IOException { Timber.tag(BuildConfig.TAG).d("[%s] sendReceipt()", TAG); return transactionClient.sendReceipt(email, transactionId); }

Print Receipt

The receipt request object has been largely changed, as the call no longer requires a receiver and it now returns a response object instead of a void. As such, `printReceipt` can now successfully message print failures and errors.

Version	Code Change
2.27	public synchronized final void **printReceipt**(EMVReceipt receipt, PrintReceiptStatusReceiver receiver) { Timber.tag(TAG).i("Print Receipt"); Terminal terminal = terminalManager.getTerminal(); if (terminal == null) { Timber.tag(TAG).w("No Active Terminal Controller"); if (receiver != null) { receiver.sendPrintReceiptStatus(PrintReceiptStatusReceiver.Status.ERROR); } return; } if (!terminal.isPrintingSupported()) { Timber.tag(TAG).w("The Active Terminal Controller Does Not Have Print Capabilities"); if (receiver != null) { receiver.sendPrintReceiptStatus(PrintReceiptStatusReceiver.Status.ERROR); } return; } terminal.printReceipt(receipt, receiver); }

2.30	`public PrintReceiptResponse printReceipt(PrintReceiptRequest request) { Timber.tag(BuildConfig.TAG).d("[%s] printReceipt()", TAG); return terminalClient.printReceipt(request); }`

Query Transactions

The `TransactionDetailFormat` and `includeRelated` fields are no longer required to query for transactions.

Version	Code Change
2.27	`public final TransactionsDetailsResponse queryTransactionDetails(QueryParameters queryParameters, PagingParameters pagingParameters, TransactionDetailFormat format, boolean includeRelated) { Timber.tag(TAG).i("Querying Transaction Details"); return platform.requestTransactionDetails(queryParameters, pagingParameters, format, includeRelated); }`  `public final TransactionsDetailsResponse queryTransactionDetails(QueryParameters queryParameters, PagingParameters pagingParameters, boolean includeRelated) { Timber.tag(TAG).i("Querying Transaction Details"); return platform.requestTransactionDetails(queryParameters, pagingParameters, includeRelated); }`  `public final TransactionsSummaryResponse queryTransactionSummaries(QueryParameters queryParameters, PagingParameters pagingParameters, TransactionDetailFormat format, boolean includeRelated) { Timber.tag(TAG).i("Querying Transaction Summaries"); return platform.requestTransactionSummaries(queryParameters, pagingParameters, format, includeRelated); }`

```
    public final TransactionsSummaryResponse
queryTransactionSummaries(QueryParameters queryParameters,

PagingParameters pagingParameters,
                                                                    boolean
includeRelated) {
        Timber.tag(TAG).i("Querying Transaction Summaries");
        return platform.requestTransactionSummaries(queryParameters,
pagingParameters, includeRelated);
    }

    public final TransactionsFamiliesResponse
queryTransactionsFamilies(QueryParameters queryParameters,

PagingParameters pagingParameters) {
        Timber.tag(TAG).i("Querying Transaction Families");
        return platform.requestTransactionFamilies(queryParameters,
pagingParameters);
    }
```

2.30

```
/**
     * queries transaction detail information
     */

    public QueryTransactionsDetailResponse
queryTransactionsDetail(QueryParameters queryParameters, PagingParameters
pagingParameters)
            throws IOException {
        Timber.tag(BuildConfig.TAG).d("[%s] queryTransactionsDetail()", TAG);
        return queryClient.queryTransactionsDetail(pagingParameters,
queryParameters);
    }




/**
     * queries transaction summary information
     */

    public QueryTransactionsSummaryResponse
queryTransactionsSummary(QueryParameters queryParameters, PagingParameters
pagingParameters)
            throws IOException {
        Timber.tag(BuildConfig.TAG).d("[%s] queryTransactionsSummary()", TAG);
        return queryClient.queryTransactionsSummary(pagingParameters,
queryParameters);
    }

    /**
     * queries transaction family information
     */
```

	`public QueryTransactionsFamilyResponse queryTransactionsFamily(QueryParameters queryParameters, PagingParameters pagingParameters) throws IOException { Timber.tag(BuildConfig.TAG).d("[%s] queryTransactionsFamily()", TAG); return queryClient.queryTransactionFamilies(pagingParameters, queryParameters); }`

CommerceDriver.java (Account Management)

Login

In version 2.30, `startSession` changed to `login`, and there was a minor change in the response object, as shown below.

Version	Code Change
2.27	`public final StartSessionResponse startSession(String username, String password) { Timber.tag(TAG).i("Starting Session"); return platform.startSession(username, password); }`
2.30	`public LoginResponse login(String userName, String password) throws IOException { Timber.tag(BuildConfig.TAG).d("[%s] login()", TAG); return sessionClient.login(userName, password); }`

Logout

In version 2.30, `invalidateSession` changed to `logout`.

Version	Code Change
2.27	`public final void invalidateSession() { Timber.tag(TAG).i("Invalidating Session"); platform.invalidateSession(); }`
2.30	`public void logout() { Timber.tag(BuildConfig.TAG).d("[%s] logout()", TAG); sessionClient.logout(); }`

Change Password

`changePassword` has been deprecated and will now call `changeUserPassword`.

Version	Code Change
2.27	`public final ApiResponse changePassword(String username, String currentPassword, String newPassword) { Timber.tag(TAG).i("Changing Password"); return platform.requestPasswordChange(username, currentPassword, newPassword); }`
2.30	`public ChangeUserPasswordResponse changePassword(String username, String currentPassword, String newPassword) throws IOException { Timber.tag(BuildConfig.TAG).d("[%s] changePassword()", TAG); ChangeUserInformationResponse response = changeUserPassword(username, currentPassword, newPassword); return new ChangeUserPasswordResponse(); }`  `public ChangeUserInformationResponse changeUserPassword(String userName, String userPassword, String newPassword) throws IOException { Timber.tag(BuildConfig.TAG).d("[%s] changeUserPassword()", TAG); return sessionClient.changeUserPassword(userName, userPassword, newPassword); }`

Reset Password

`resetPasswordWithAnswer` and `requestResetPasswordQuestion` have been removed from version 2.30 and replaced with `resetUserPassword` and `forgotUserPassword`.

Version	Code Change
2.27	`public final ApiResponse resetPasswordWithAnswer(String username, String token, String text) { Timber.tag(TAG).i("Resetting Password");`

	`        return platform.resetPasswordWithAnswer(username, token, text);` `    }` `    public final ResetPasswordQuestionResponse requestResetPasswordQuestion(String username) {` `        Timber.tag(TAG).i("Forgot Password");` `        return platform.requestPasswordResetQuestion(username);` `    }`
2.30	`/**` ` * attempts to reset a user password with the answer to a security question` ` */` `    public ResetPasswordResponse resetPassword(String username, String token, String text) throws IOException {` `        Timber.tag(BuildConfig.TAG).d("[%s] resetUserPassword()", TAG);` `        ChangeUserInformationResponse result = resetUserPassword(username, token, text);` `        return new ResetPasswordResponse();` `    }` `    /**` ` * attempts to reset a user password with the answer to a security question` ` */` `    public ChangeUserInformationResponse resetUserPassword(String username, String token, String text) throws IOException {` `        Timber.tag(BuildConfig.TAG).d("[%s] resetUserPassword()", TAG);` `        return sessionClient.resetUserPassword(username, token, text);` `    }` `   /**` ` * requests a security question to answer if the password for a user is forgotten` ` */` `    public ForgotUserPasswordResponse forgotUserPassword(String username) throws IOException {` `        Timber.tag(BuildConfig.TAG).d("[%s] forgotUserPassword()", TAG);` `        return sessionClient.forgotUserPassword(username);` `    }`

Get Security Questions

Version	Code Change
2.27	`    public final SecurityQuestionsResponse getAvailableSecurityQuestions(String username, String password) {` `        Timber.tag(TAG).i("Getting Available Security Questions");` `        return platform.requestAvailableSecurityQuestions(username, password);`

	`}` `public final SecurityQuestionsResponse getSecurityQuestions(String username, String password) {` `Timber.tag(TAG).i("Get Security Questions");` `return platform.requestSecurityQuestions(username, password);` `}`
2.30	`public SecurityQuestionsResponse getAvailableSecurityQuestions() throws IOException {` `Timber.tag(BuildConfig.TAG).d("[%s] getAvailableSecurityQuestions()", TAG);` `return sessionClient.getAvailableSecurityQuestions();` `}` `public SecurityQuestionsResponse getSecurityQuestions() throws IOException {` `Timber.tag(BuildConfig.TAG).d("[%s] getSecurityQuestions()", TAG);` `return sessionClient.getSecurityQuestions();` `}`

Set Security Questions

`saveSecurityQuestions` was updated to `updateSecurityQuestions`, and there is a new returned `UpdateSecurityQuestionsResponse` object.

Version	Code Change
2.27	`public final ApiResponse saveSecurityQuestions(String username, String password, List<SecurityAnswer> answers) {` `Timber.tag(TAG).i("Saving Security Questions");` `return platform.requestSaveSecurityQuestions(username, password, answers);` `}`
2.30	`/**` ` * updates a user's security questions` ` *` ` * @return the update security questions response` ` */` `public UpdateSecurityQuestionsResponse updateSecurityQuestions(String username, String password, List<SecurityAnswer> answers) throws IOException {` `Timber.tag(BuildConfig.TAG).d("[%s] updateSecurityQuestions()", TAG);` `ChangeUserInformationResponse response = updateUserSecurityQuestions(username, password, answers);` `return new UpdateSecurityQuestionsResponse();` `}`

	`/** * updates a user's security questions * * @return the change user information response */ public ChangeUserInformationResponse updateUserSecurityQuestions(String username, String password, List<SecurityAnswer> answers) throws IOException { Timber.tag(BuildConfig.TAG).d("[%s] updateSecurityQuestions()", TAG); return sessionClient.updateSecurityQuestions(username, password, answers); }`

Get Password Expiration

`getPasswordExpiration` changed to `getUserExpiration` in version 2.30.

Version	Code Change
2.27	`public final PasswordExpirationResponse getPasswordExpiration(String username, String password) { Timber.tag(TAG).i("Getting User Expiration"); return platform.requestPasswordExpiration(username, password); }`
2.30	`public PasswordExpirationResponse getUserExpiration(String username, String password) throws IOException { Timber.tag(BuildConfig.TAG).d("[%s] getUserExpiration()", TAG); return sessionClient.getUserExpiration(username, password); }`

Get Merchant Profile

`getBankCardMerchantProfiles` changed to `getBankingMerchant` in version 2.30.

Version	Code Change
2.27	`public final Set<MerchantProfile> getBankCardMerchantProfiles() { return platform.getBankCardMerchantProfiles(); } public final MerchantProfile getBankCardMerchantProfile() { Timber.tag(TAG).d("Getting Active Bank Card Entry"); return platform.getBankCardMerchantProfile(); }`

2.30	`/** * gets the selected merchant profile for this session */ public MerchantProfile getBankingMerchant() { Timber.tag(BuildConfig.TAG).d("[%s] getBankingMerchant()", TAG); return sessionClient.getSelectedMerchantProfile(); } /** * gets the selected merchant profile id for this session */ public String getSelectedMerchantProfileId() { Timber.tag(BuildConfig.TAG).d("[%s] getSelectedMerchantProfileId()", TAG); return sessionClient.getSelectedMerchantProfileId(); } /** * gets the available merchant profile ids for the current session */ public List<String> getAvailableMerchantProfileIds() { return sessionClient.getAvailableMerchantProfileIds(); }`

Set Merchant Profile

`setBankCardMerchantProfile` changed to `selectBankingMerchant`.

Version	Code Change
2.27	`public final boolean setBankCardMerchantProfile(String mpid) { Timber.tag(TAG).i("Setting Bank Card Merchant Profile = %s", mpid); return platform.setBankCardMerchantProfile(mpid); }`
2.30	`public boolean selectBankingMerchant(String mpid) { Timber.tag(BuildConfig.TAG).d("[%s] selectBankingMerchant()", TAG); return sessionClient.selectBankingMerchant(mpid); }`

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