Oracle JD Edwards EnterpriseOne – Mobile Order Entry

Third-Party Credit Card Plug-In

O R ACL E BUSINESS BRIEF | SEPT E M B ER 20 1 6

Disclaimer

JD Edwards EnterpriseOne Mobile Order Entry: Overview of Third-Party Credit Card Plug-In

The JD Edwards EnterpriseOne Order Entry mobile tablet application contains a generic interface for a third-party credit card plug-in and processing device. This document discusses:

» The messaging from the Order Entry tablet application when processing a credit card
» The input messages required for enabling the credit card option within the Order Entry tablet application
» Processing credit card authorization information that is returned from the plug-in

JD Edwards business partners may use this documentation for building the third-party hardware and software plug-in. The third-party credit card plug-in must be written for the specific hardware device and for the mobile platform that the device supports (iOS or Android). The Order Entry tablet application will be made available in the Mobile Application Archive (MAA) format for business partners and for customers. Oracle will not be responsible for supporting the plug-in code and the credit card processes.

Generic Java Script Interface

Within the Order Entry MAA, a generic Java script (creditcardscanner.js) can be used to integrate with the Order Entry tablet application. Only four Java script functions are discussed. Please note that the functions are stubbed out with examples from the prototype completed at Oracle. The Java script interface is generic in nature and can be used by an Android or iOS third-party plug-in. At this time, Oracle has only prototyped one plug-in with iOS device and iOS compatible credit card hardware. No banking transactions from the credit card plug-in proof of concept were completed. All banking and credit card processing should be handled with the third-party plug-in and should be supported by the business partner.

Credit Card Transaction Flow

The intention and design of the credit card payment option for the Order Entry tablet application is to handle all of the credit card processing in a third-party plug-in that can be added to the application via the MAA. Business partners and customers are responsible for the design, build, and support of the third party plug-in.

For the Order Entry tablet application, an internal parameter must be set to denote that the credit card device is connected. If the credit card device is connected, the credit card payment option will be enabled and displayed in the payment option drop-down list.

When the user selects the credit card option, the Submit button is enabled. The user then taps the Submit button, and the application pushes down the Sales Information through the generic Java script interface to the third-party plug-in. No other action occurs after this. The third-party plug-in contains the Sales Order header information. Depending on the design of the third-party plug-in, the credit card is swiped to retrieve the credit card information. The third-party plug-in will then process the sales and credit card information with banking processes.

To complete the transaction, an authorization message must be pushed from the third-party plug-in back into the Order Entry tablet application through the generic Java script interface. The application then takes the Approved authorization and populates the prepayments table with the Authorization ID and Authorization Date. Finally, the sales order is created and a confirmation message is displayed for the sales order creation.

If the Authorization status is “Invalid” then no processing occurs. The third-party plug-in must display an error message from the bank process.

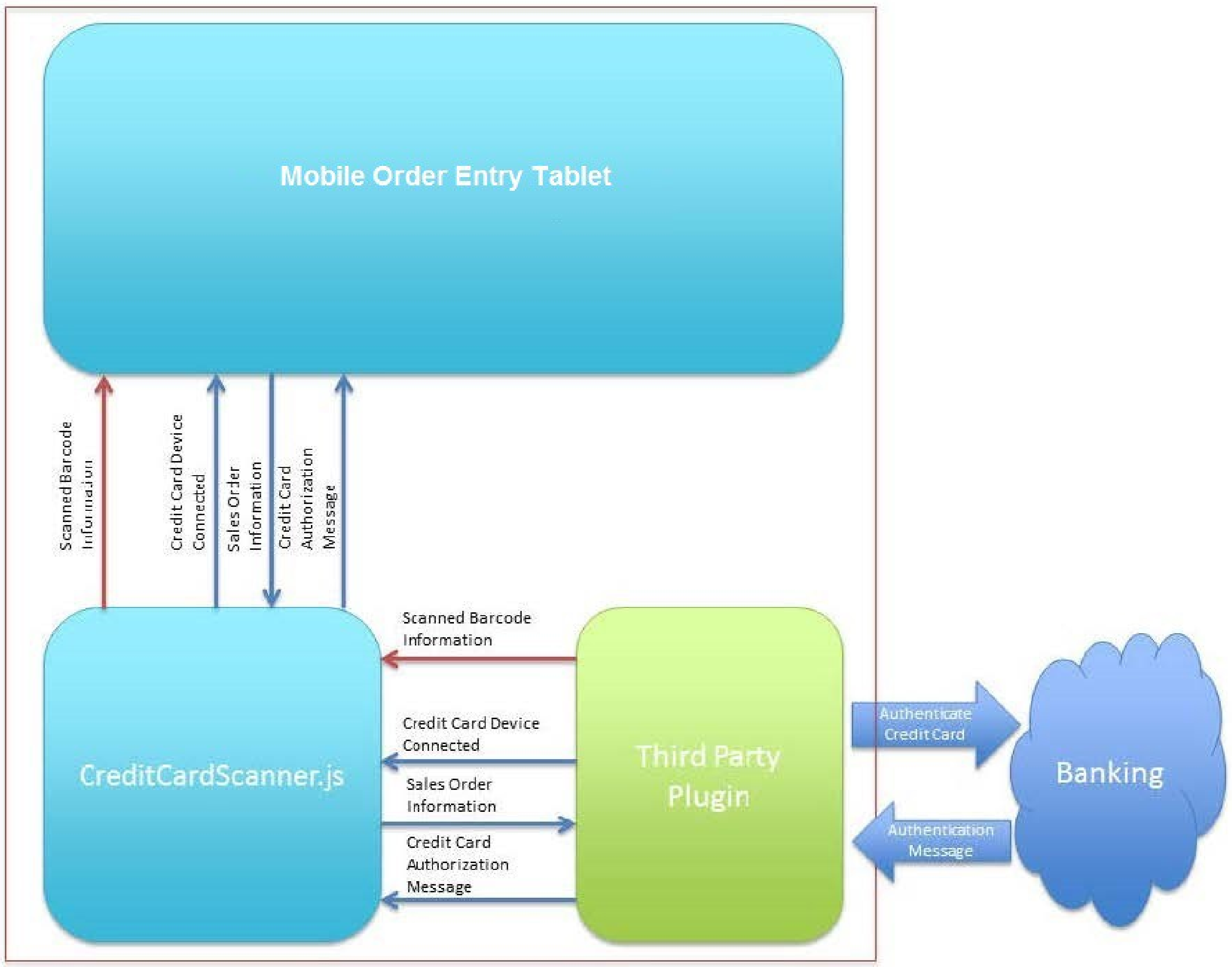
Mobile Order Entry Tablet
Scanned Barcode Information
Credit Card Device Connected
Sales Order Information
Credit Card Authorization Message
CreditCardScanner.js
Scanned Barcode Information
Credit Card Device Connected
Sales Order Information
Credit Card Authorization Message
Third Party Plugin
Authenticate Credit Card
Authentication Message
Banking

Figure 1.Credit card transaction flow

Message and Data Flow

The CreditCardScanner.js Java script is a generic interface that acts as a gateway for messages between the Order Entry tablet application and the third-party plug-in. The four primary messages discussed are:

» Scanned Barcode Information

» Credit Card Device Connected

» Sales Order Information

» Credit Card Authentication Message

All four messages are passed individually as a single text variable within each Java script function. Sales Order Information and Credit Card Authentication Message are stored in the JSON format passed in and out of the Order Entry tablet application.

Java Script Functions

returnedBarcode

The returnedBarcode function passes a text parameter from the third-party barcode scanner to the Order Entry tablet application. The function takes the single parameter and calls setBarcodeResult within the barcode bean. A call to the main application is made to process the barcode and load the item to the shopping cart.

checkConnectionStatus

The checkConnectionStatus function is called from the Order Entry tablet application. It is intended to invoke a plug-in function that will determine if a credit card device is connected. If the device is connected, the internal applicationScope.variable (pageflowScope.scannerReady) must be populated with a 1 to denote that the device is connected. This enables the Credit Card option on the Check Out page within the Order Entry tablet application.

processCreditCardJS

The processCreditCardJS function passes the Sales Order Information (which is already in the JSON format) to the third-party plug-in. The Sales Order Information JSON contains the Sales Order header information and Order Summary Information(Order Totals). These are passed to the plug-in for credit card processing. The plug-in supported by the partner should combine the Sales Order Information and Credit Card information to process the credit card with the bank.

returnedAuthorizationInfo

The returnedAuthorizationInfo function passes the Authorization Information in JSON format from the third party to the Order Entry table application. The function takes a single text parameter and loads it to an internal pageFlowScope (pageFlowScope.authenticationResponse) variable used by the application. The processAuthorizationInfo method in the ccBean is then called to process the authorization. If the Authorization JSON contains the status of “Approved”, the Order Entry tablet application updates the Authorization ID and Date to the prepayments table and creates the sales order. When the order is created the Order Number is displayed in the confirmation message. If the Authorization JSON, contains any other value, no further processing occurs. It is the responsibility of the third-party plug-in to display an error message.

JSON Messages and Required Information

Sale Information Message (Parameters and JSON Example)

Parameter	**Description**
orderType	Order type from the sales order.
orderNumber	Order number from the sales order.
orderCompany	Order company associated with the sales order.
orderKey	Concatenated order number, order type, and order company.

transactionAmount	Sales order transaction amount.
currencyCode	Currency code associated with the transition.
transactionDate	Transaction date from the sales order.
contactName	Contact associated with the sales order.
billToConstantsMailingName	Bill to constant mailing name from address book.
billToMailingName	Bill to mailing name from address book.
billToAddressLine1	Bill to address line 1 from address book.
billToAddressLine2	Bill to address line 2 from address book.
billToAddressLine3	Bill to address line 3 from address book.
billToAddressLine4	Bill to address line 4 from address book.
billToAddressLine5	Bill to address line 5 from address book.
billToAddressLine6	Ship to address line 6 from address book.
billToAddressLine7	Ship to address line 7 from address book.
shipToConstantsMailingName	Ship to constant mailing name from address book.
shipToMailingName	Ship to mailing name from address book.
shipToAddressLine1	Ship to address line 1 from address book.
shipToAddressLine2	Ship to address line 2 from address book.
shipToAddressLine3	Ship to address line 3 from address book.
shipToAddressLine4	Ship to address line 4 from address book.
shipToAddressLine5	Ship to address line 5 from address book.
shipToAddressLine6	Ship to address line 6 from address book.
shipToAddressLine7	Ship to address line 7 from address book.

*All parameters are in the String format.

JSON Example:

```
{
  "billToAddressLine6": " ",
  "billToAddressLine7": " ",
  "orderType": "",
  "orderNumber": "",
  "shipToConstantsMailingName": "Parts Emporium changed",
  "shipToAddressLine7": " ",
  "propertyChangeSupport": {
    ".type": "oracle.adfmf.java.beans.PropertyChangeSupport",
    "propertyChangeListeners": []
  },
  "shipToAddressLine5": " ",
  "shipToAddressLine6": " ",
  "shipToAddressLine3": " ",
  "shipToAddressLine4": " ",
  "shipToAddressLine1": "Denver CO 80205",
  "shipToAddressLine2": " ",
  "paymentType": "4",
  ".type": "com.oracle.e1.jdemf.M42104.application.SalesOrderInformation",
  "transactionAmount": "137.09",
  "billToMailingName": "Capital System DO NOT MODIFY",
  "billToAddressLine1": "Suite 200",
  "billToAddressLine2": "Unit 5",
  "billToAddressLine3": "Atlanta GA 30341",
  "billToAddressLine4": " ",
  "billToAddressLine5": " ",
  "orderKey": "",
  "shipToMailingName": "4022 Walnut Street, Suite 280",
  "contactName": "Jon Stirling",
  "transactionDate": "2015/33/27 13:33:26",
  "billToConstantsMailingName": "Capital System DO NOT MODIFY",
  "orderCompany": "",
  "currencyCode": "USD"
}
```

Credit Card Authorization Message (Parameters and JSON Example)

Parameter	**Description**
authorizationCurrency	Currency associated with the transaction.
cardHolderName	Name on the credit card.
authorizationMessage	Optional message that can come from the bank.
cardType	Type of credit card. Example: MSTR, VISA, etc.
lastFourDigets	Last four digits of credit card number.

authorizeTransactionAmount	Authorized credit card purchase amount.
authorizationStatus	Required: Status must be Approved for the Sales Order Entry table application to continue processing and create the sales order.
authorizeTransactionID	Required: Unique transaction ID sent from the banking process.
authorizationErrorMessage	Error message issued from the banking process.
referenceNumber	Reference number acquired from the banking process.
authorizationExpiration	Credit card expiration date from the credit card.
authorizeDateTime	Required: Authorization date when the credit card was processed and approved. This is the date only, and should be in the default format of the EnterpriseOne system.

*All parameters are in the String format.

JSON Example:

```
{
   "authorizationCurrency": "USD",
   "cardHolderName": "John Doe",
   "authorizationMessage": "Message from the bank.",
   "cardType": "MSTR",
   "lastFourDigets”: "1234",
   "authorizeTransactionAmount": "252.00",
   "authorizationStatus": "Approved",
   "authorizeTransactionID": "123456321",
   "authorizationErrorMessage": "Invalid Card",
   "referenceNumber": "fd1s235s7",
   ".type": "com.oracle.e1.jdemf.M42104.application.CreditCardAuthentication",
   "authorizationExpiration": "12/15/2019",
   "authorizeDateTime": "2015/04/27"
}
```

Oracle JD Edwards EnterpriseOne – Mobile Order Entry Third-Party Credit Card Plug-In
September 2016
Author: Jonathan Stirling