Microsoft Dynamics® AX

Changes to credit card processing in Microsoft Dynamics AX 2012 R2

Implementation Note

This document explains what has changed in the implementation of credit card processing in Accounts receivable in Microsoft Dynamics AX 2012 R2.

January 2013

www.microsoft.com/dynamics/ax

Karl Gunderson, Senior Software Development Engineer

Send suggestions and comments about this document to adocs@microsoft.com. Please include the title with your feedback.

Table of Contents

Before you begin

This document describes what has changed in the implementation of credit card processing in Accounts receivable in Microsoft Dynamics AX 2012 R2. These changes are driven primarily by the switch from a single credit card processor that uses a proprietary interface in Microsoft Dynamics AX 2012, to support for multiple credit card processors that are based on a published interface in Microsoft Dynamics AX 2012 R2. Support for multiple credit card processors is a benefit of using the Payment SDK. This document assumes that you are familiar with the Payment SDK.

Terminology

- **Payment SDK** – A software toolkit that enables the construction and consumption of services from a payment provider. A connector built with the SDK is the primary means of enabling consumption of services from a payment provider.
- **Connector** – A software component built with the Payment SDK that implements the IPaymentProcessor interface.
- **IPaymentProcessor** – An interface with the fully qualified name Microsoft.Dynamics.Retail.PaymentSDK.IPaymentProcessors. A connector implements this interface, and software that requires the services of a payment provider consumes the interface.

Class model

The following illustration shows the primary components of credit card processing in Microsoft Dynamics AX. **CreditCardProcess** is an abstract class that provides the interface to the rest of Microsoft Dynamics AX. **CreditCardProviderProcess** extends **CreditCardProcess** for connectors that implement the IPaymentProcessor interface. **DynamicsOnlineConnector** is a connector that implements the IPaymentProcessor interface to Payment Service for Microsoft Dynamics ERP and is provided with Microsoft Dynamics AX 2012 R2.

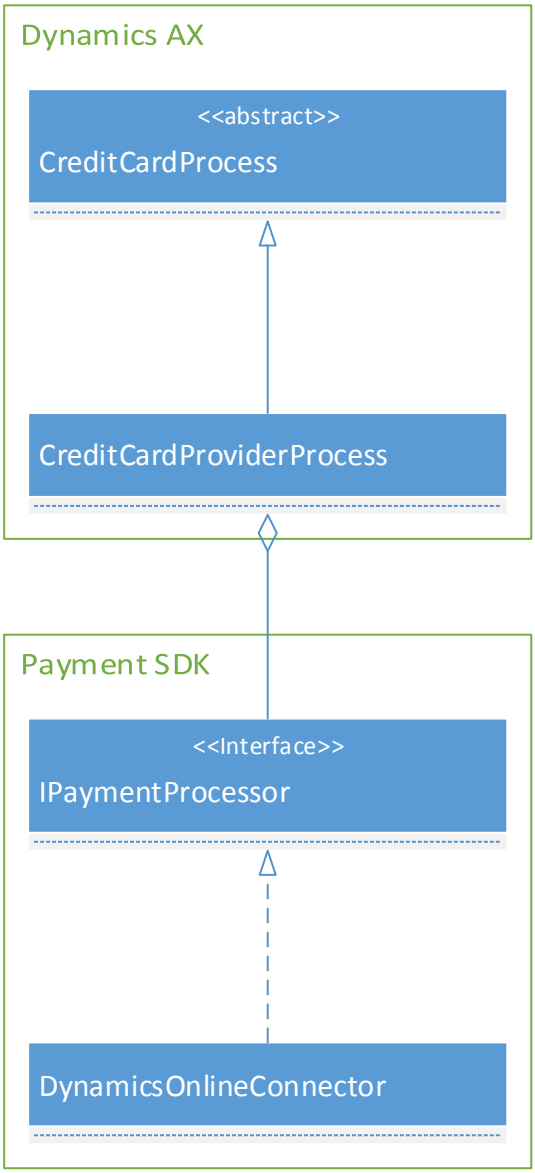

Figure 1 New class model

The following illustration shows the old class model for Microsoft Dynamic AX 2012.

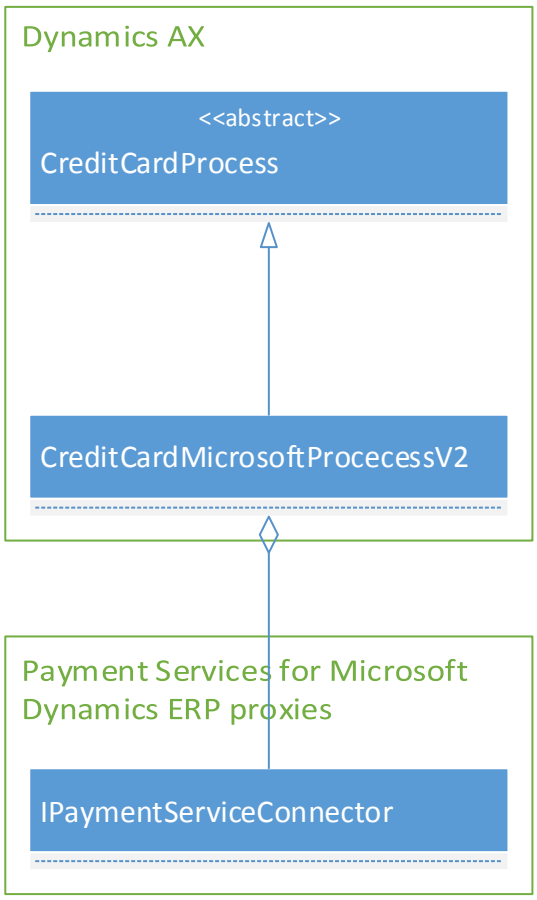

Figure 2 Old class model

There are three changes to the **CreditCardProcess** class:

- The static public constructor method **construct()** no longer takes a parameter to indicate which processor is used. Instead, the processor is determined by the credit card associated with the **_record** parameter.
- The read-only public property method **parmCreditCardProcessor()** returns a **CreditCardProcessorId** string value instead of a **CreditCardProcessor** enumerated value.
- The protected method **creditCardNumber()** has been removed. Instead, use the **CreditCard::creditCardNumber()** method, or retrieve the **Last4Digits** property from the appropriate credit card property collection.

What changed

Changes made to Microsoft Dynamics AX 2012 R2 credit card processing in Accounts Receivable are primarily confined to these areas:

- Payment SDK
- Setup of credit card processors
- Cardholder and card information
- API changes
- Credit card history
- Upgrade

Payment SDK

The basis for changes to Microsoft Dynamics AX 2012 R2 is the Payment SDK. The APIs and data used in the SDK are similar to the previous proprietary interface to the Microsoft Dynamics Online Payment Services web services as made available through the proxies (.NET assemblies or DLLs) provided with Microsoft Dynamics AX 2012. The SDK is publicly available and uses a published interface. One of the biggest differences is that instead of using a fixed API signature, the SDK uses a property collection to send and receive values.

At its most basic, a property collection consists of a collection of properties, and a property is a name/value pair. A property collection can contain nested property collections. A property is self-describing, in that it can have attributes. If this sounds like XML, it should: the physical representation of a property collection is XML.

A property name consists of two dot separated parts, the namespace and the name. An example of a property name is **Connector.ConnectorName**, where Connector is the namespace and ConnectorName is the name.

Property attributes include the following:

- **Value type** – String, decimal, date/time, or property list (a nested property collection).
- **Security level** – One of the following:
  - **None** – Non-sensitive information, such as a transaction ID GUID.
  - **PCI** – Payment card information, such as a credit card number.
  - **PII** – Personally identifiable information, such as a cardholder's name.
  - **HSPII** – Highly sensitive PII, such as a merchant service account ID.
- **Sequence number** – A non-negative integer for display ordering.
- **Is hidden** – A Boolean value indicating a non-displayed property.
- **Is password** – A Boolean value indicating an obscured value.
- **Is encrypted** – A Boolean value indicating an encrypted value (any security level except none).
- **Is read-only** – A Boolean value indicating a non-editable value.
- **Display height** – An integer indicating the display field height in lines.

A property's value is one of the following:

- **Stored string value** – The value of a string type field, possibly encrypted.
- **Decimal value** – The value of a decimal type field.
- **Date value** – The value of a date/time type field.

Setup of credit card processors

Before Microsoft Dynamics AX 2012 R2, credit card processors came from a fixed set and were identified by values of the enumerated type **CreditCardProcessor**. With the Payment SDK, Microsoft Dynamics AX 2012 R2 now supports multiple processors identified by a user-selected name.

In earlier versions of Microsoft Dynamics AX, one of the fixed set of processors was identified as "active." The active processor was the processor associated with a sales order when the sales order was created and a credit card was selected as the payment type. In Microsoft Dynamics AX 2012 R2, the processor is associated with a cardholder's credit card when the card is created. In the **Payment services setup** form, this explains the change of the label for the **Active** check box to **Default processor for new credit cards**.

Because the processor used during a credit card operation is determined by the credit card selected on the sales order, the credit card lookup includes the payment service name.

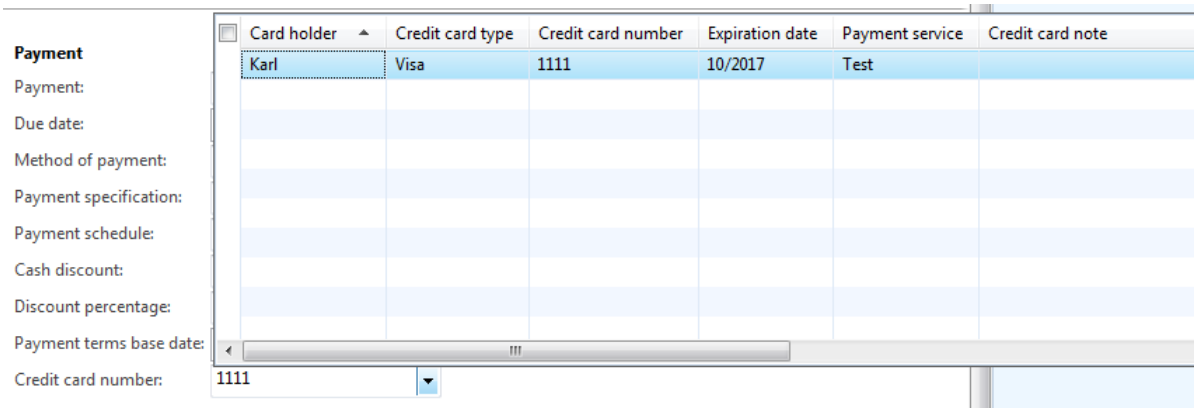

Figure 3 Credit card lookup

The other significant change to the setup of a processor is the **Payment service account** field group in the form. The fields in this section of the form are dynamic and are based on the selected payment connector. Each connector determines the information that is necessary to establish a connection to the payment service. The connector provides this information as a metadata property collection, and the fields in the group are generated dynamically from the metadata.

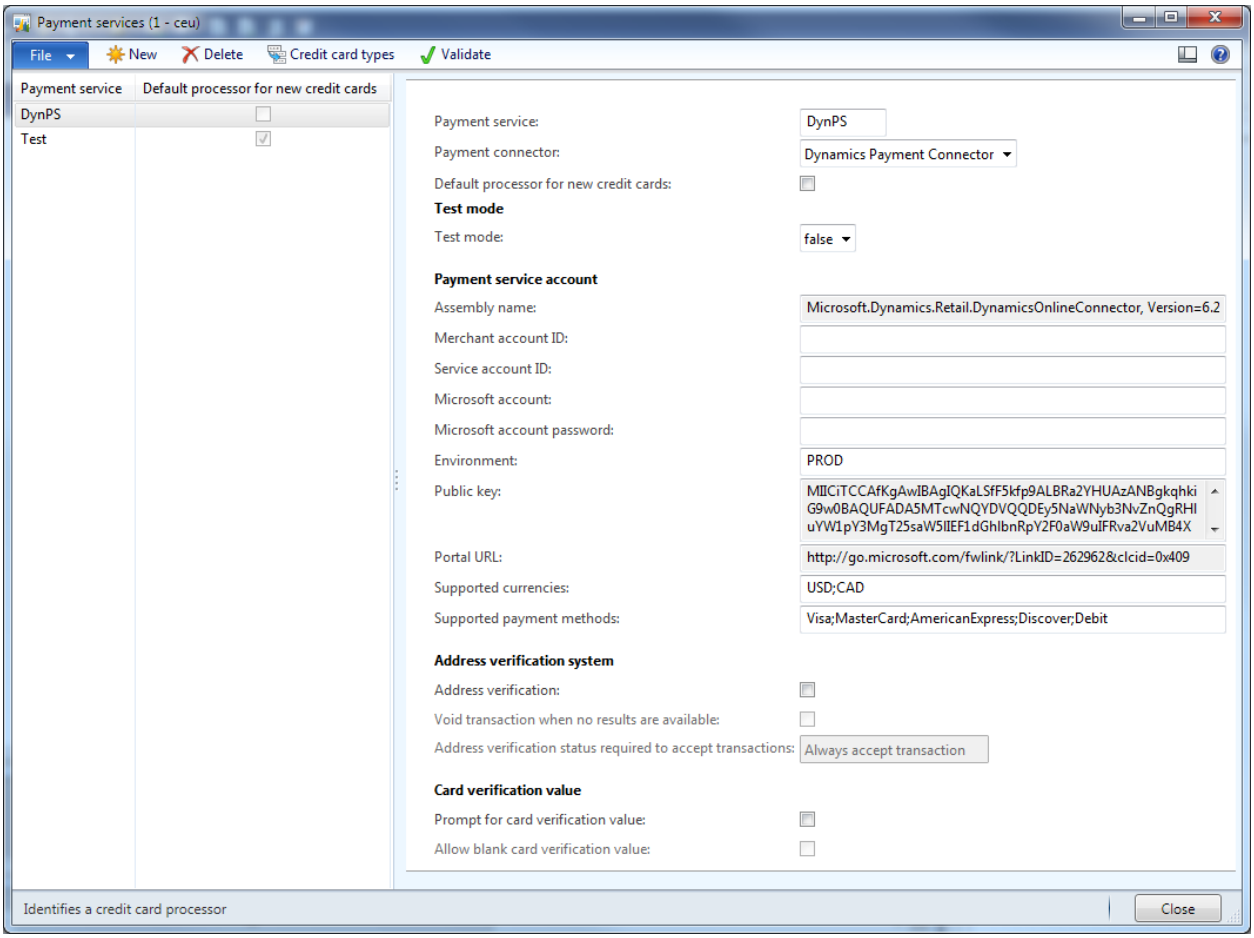

Figure 4 Payment services setup form

Cardholder and card information

Before Microsoft Dynamics AX 2012 R2, all credit cardholder and card information was stored in clear text in application tables. Special provisions were made to ensure that the credit card number was not displayed in the Microsoft Dynamics AX client or even transferred to the client, but this was still a source of concern and prevented Microsoft Dynamics AX 2012 from passing PA-DSS (Payment Application Data Security Standard) certification out of the box.

With Microsoft Dynamics AX 2012 R2, PCI and PII data are encrypted. The credit card number is further tokenized and secured by the connector for Microsoft Dynamics Online Payment Services and never passed to Microsoft Dynamics AX 2012 R2 in plain text. Because of this and other considerations, Microsoft Dynamics AX 2012 R2 is PA-DSS certified. To achieve this level of data security, the connector is responsible for collecting the credit card number and other sensitive data.

Before Microsoft Dynamics AX 2012 R2, Microsoft Dynamics AX forms (that is, the Credit Card Wizard, the **CreditCardWizard** form) collected all the cardholder and card information. In Microsoft Dynamics AX 2012 R2, there is a split in responsibility between Microsoft Dynamics AX and the connector. When collecting information for a new credit card, the connector provides the dialog box with a field for entering the credit card number and other fields.

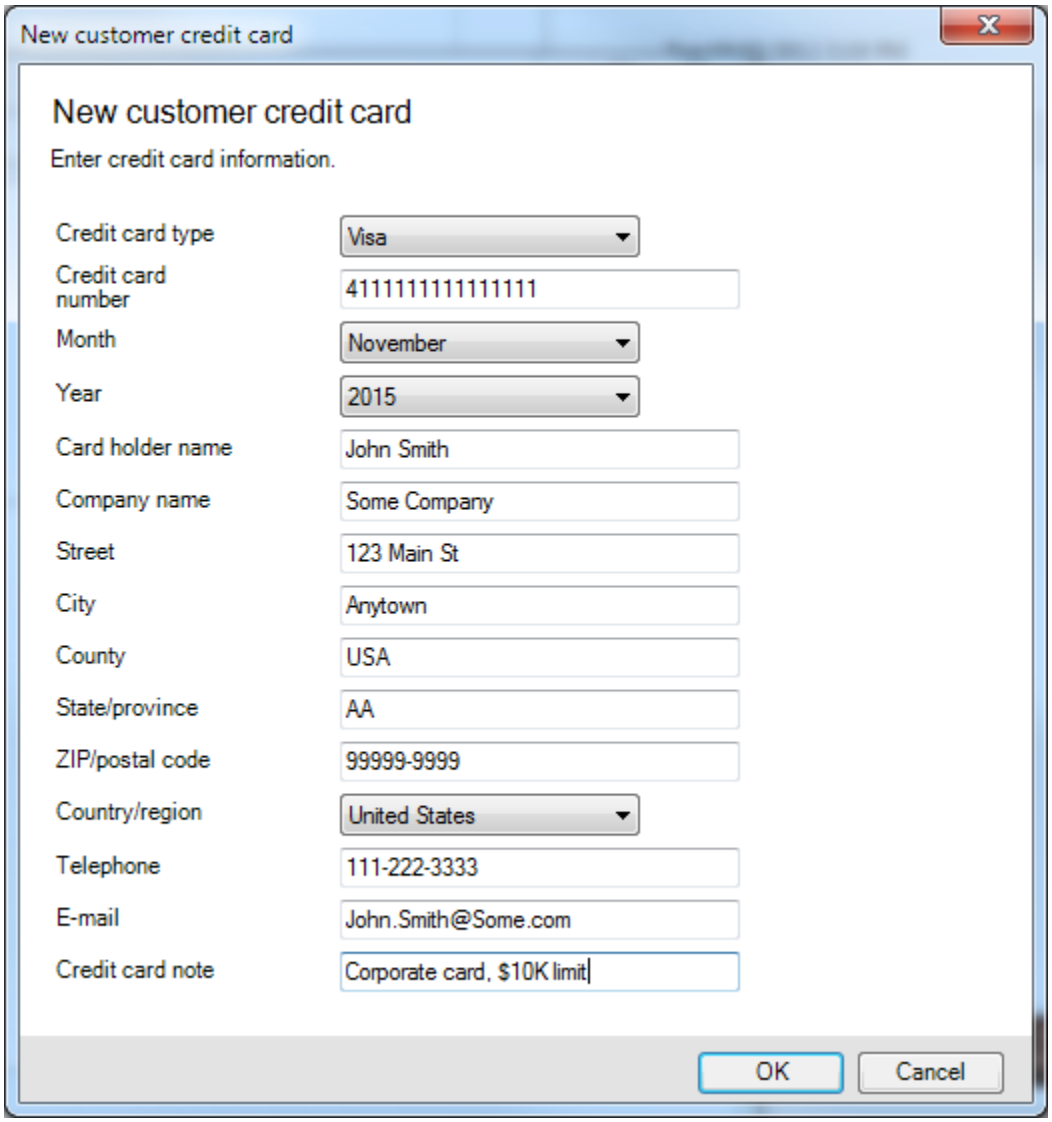

Figure 5 Adding a new credit card

As in Microsoft Dynamics AX 2012, the whole credit card number is never displayed once it is entered and validated, only the last four digits. However, in Microsoft Dynamics AX 2012 R2, if the credit card number or card type is entered incorrectly, the user must delete and recreate the record. After the credit card information has been created, the other fields can still be edited. For example, if the cardholder's address is incorrect, or if the cardholder moves, the address information can be updated.

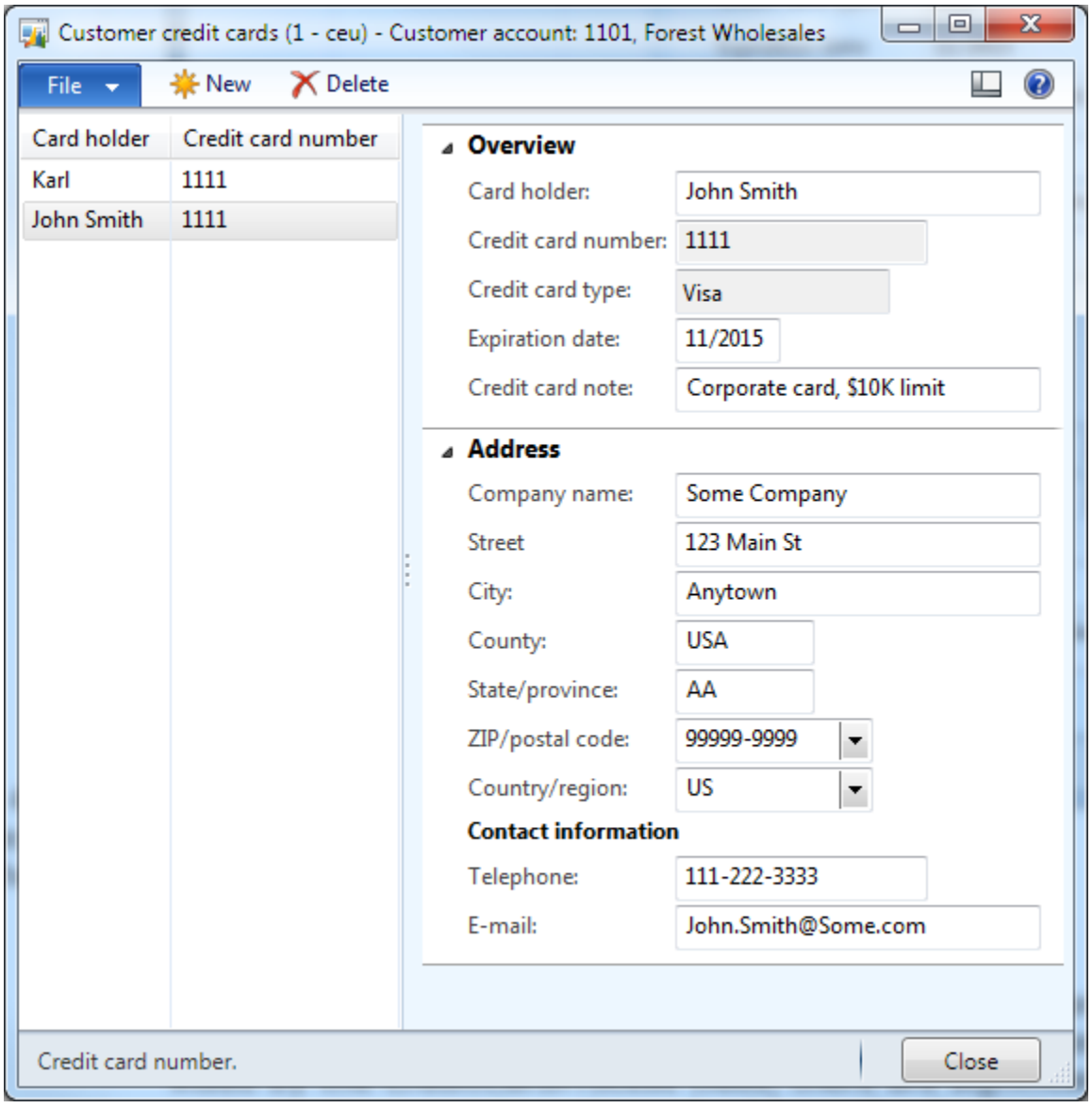

Figure 6 Updating an existing credit card

API changes

As explained previously, the public APIs to Microsoft Dynamics AX credit card processing are largely unchanged. However, at the level below the interface to Microsoft Dynamics AX provided by the **CreditCardProcess** class, there are significant changes. Any customizations or integrations should use the **CreditCardProcess** class as much as possible. If that is not possible, using the types that support the Payment SDK, or using the SDK directly, is relatively safe, because the SDK represents a published, publicly available interface.

Code-level or API changes are dealt with in these sections:

- CreditCardMicrosoftProcessV2 deleted
- CreditCardProviderProcess added
- New CreditCard methods
- New auxiliary classes

CreditCardMicrosoftProcessV2 deleted

Because the interface provided by the Payment SDK is nearly functionally identical to the interface provided by the Microsoft Dynamics Online Payment Services proxies, and the connector supplied with Microsoft Dynamics AX 2012 R2 provides access to the same service, support for the proxies has been dropped. Support for the proxies in Microsoft Dynamics AX 2012 was the responsibility of the **CreditCardMicrosoftProcessV2** class. The equivalent functionality is now provided by the **CreditCardProviderProcess** class. Both are subclasses of **CreditCardProcess**.

CreditCardProviderProcess added

A person who knows the **CreditCardMicrosoftProcessV2** class will find the **CreditCardProviderProcess** class is very familiar. Most methods are still present, and many have either little or no change to their signatures. The class is smaller, because some functionality has been factored out for simplicity and code reuse. For example, if credit card processing takes place externally, there are methods that record those operations in the **CreditCard** class. These methods were built for and used by the **Retail** module for point of sale and the new SharePoint store front end, and are used by the **CreditCardProviderProcess** class. These methods are detailed in the next section.

To understand the **CreditCardProviderProcess** class, it helps to understand the super class **CreditCardProcess**.

The **CreditCardProcess** class has five entry points:

- **main()** – Assumes a SalesTable record and performs an authorize operation.
- **doPreAuth()** – Performs an authorize operation; that is, it reserves credit.
- **doCapture()** – Performs a capture operation; that is, it initiates a transfer of funds from the cardholder's account to the merchant's account, either with or without first authorizing the transfer.
- **doAuthVoid()** – Performs a void operation; that is, it releases credit that was previously reserved.
- **doRefund()** – Performs a refund operation; that is, it initiates a transfer of funds from the merchant's account to the cardholder's account.

After the operation is determined and some values are initialized, the **process()** method is invoked. The **process()** method, in pseudo-code, is as follows:

1. Validate the currency code, throwing an exception if the currency is not supported.
2. If so configured, prompt for a CVV value; exit if the user cancels the operation.

3. Prepare for further work.
4. Invoke the abstract **submit()** method.
5. Record the result of the credit card operation (see the next section).
6. Check the CVV and AVS results.
7. Update the SalesTable with the results of the credit card operation.
8. Display a message indicating the result of the credit card operation.
9. Update related credit card history records.

The heart of the **CreditCardProviderProcess** class is the **submit()** method as inherited from **CreditCardProcess**. The **submit()** method, in pseudo-code, is as follows:

1. If the operation is not a refund, and the amount of the credit card transaction is less than 0 (zero), terminate the operation.
2. If the operation is authorize, send an authorize request.
3. If the operation is capture for a previously authorized sales order, send a capture request.
4. If the operation is capture, but no previous authorization exists, send an authorize request; if the request is successful, send a capture request.
5. If the operation is refund, send a refund request.
6. If the operation is void, send a void request.

Sending a request involves several steps:

1. Create a property collection with the necessary values for the request type.
2. Add merchant account information to the property collection.
3. Create a request object with the property collection.
4. Invoke the connector operation, passing it the request object.
5. Check the generic results of the operation for errors.
6. Construct a result object of the appropriate type, a subclass of **CreditCardProviderResult**.
7. Check the operation-specific results for errors.

One area of customization or integration of particular interest might be with level-2 (header) and level-3 (line) details. During the creation of the property collection for each request type, the **setPurchaseLevelTypeAndDetails()** method is called. Overriding this method, or the **setLevel2Data()** or **setLevel3Data()** method that it calls, should provide plenty of opportunity to introduce changed or new functionality.

New CreditCard methods

Several methods have been added to the **CreditCard** class to support integration when credit card processing occurs outside Microsoft Dynamics AX:

- **addCreditCard()** – Adds a new credit card to a customer for a given credit card processor.
- **recordAuthorization()** – Records the results of an authorize operation.
- **recordCapture()** – Records the results of a capture operation.
- **recordRefund()** – Records the results of a refund operation.
- **recordVoid()** – Records the results of a void operation.

New auxiliary classes

A number of classes have been added to factor out reusable functionality:

- **CreditCardConnectorSetup** – Handles the dynamic portion of the connector setup; see the **Payment service account** field group in Figure 4.
- **CreditCardPaymentProperty** – Represents one property in a property collection.
- **CreditCardPaymentProperties** – Holds a collection of payment properties—that is, zero or more CreditCardPaymentProperty objects.
- **CreditCardPaymentCardTokenize** – Handles adding a new credit card.
- **CreditCardPaymentError** – Represents one error in an error collection.
- **CreditCardPaymentErrors** – Holds a collection of payment errors—that is, zero or more CreditCardPaymentError objects.
- **CreditCardProviderResult** – An abstract class representing the result of a credit card operation.
- **CreditCardProviderAuthorizationResult** – The result from an authorize operation.
- **CreditCardProviderCaptureResult** – The result from a capture operation.
- **CreditCardProviderRefundResult** – The result from a refund operation.
- **CreditCardProviderVoidResult** – The result from a void operation.

Credit card history

The credit card operation history is stored in the CreditCardAuthTrans table. History records are associated with a specific sales order, the SalesTable table, via the **SalesId** field found in both tables. Most fields still exist, but some are now held in one of the two property collections stored in the table.

The following fields have been deleted:

- **ApprovalAmountMST** – Renamed **DEL_ApprovalAmountMST**.
- **AuthorizeNetTransId** – Renamed **DEL_AuthorizeNetTransId**.
- **CreditCardProcessor** – Renamed **DEL_CreditCardProcessor**.
- **CreditCardRefRecId** – Renamed **DEL_CreditCardRefRecId**.
- **SecureCardData** – Renamed **DEL_SecureCardData**.

The following fields have been added:

- **CardTokenRequest** – The property collection as sent to the credit card operation.
- **CardTokenResult** – The property collection as returned from the credit card operation.
- **CreditCardNumber** – Extracted from one of the property collections for display purposes.
- **CreditCardProcessors** – A foreign key to the CredCardProcessors table, a RecId.
- **CreditCardTypeName** – Extracted from one of the property collections for display purposes.
- **UniqueCardId** – Extracted from one of the property collections for comparison and lookup purposes.

Some other fields are now also extracted from one of the property collections (**CardTokenRequest** or **CardTokenResult**); however, because these fields are never updated, they are extracted only for performance or display purposes.

Upgrade

Beyond the usual considerations for upgraded data, there are several less obvious points of interest:

- Before Microsoft Dynamics AX 2012 R2, processors were identified by the enumerated type **CreditCardProcessor**. A record is created in the CreditCardProcesors table with the name AuthNet, and any history records for CreditCardProcessor::AuthorizeNet are associated with this processor. The AuthNet processor is marked **NotUsed** and is not visible in the user interface.
- Before Microsoft Dynamics AX 2012 R2, credit card types were identified by the enumerated type **CreditCardType**. In Microsoft Dynamics AX 2012 R2, credit card types are provided as metadata by each connector, where the type is identified by a string. As a result, no special processing is performed in Microsoft Dynamics AX 2012 R2 based on credit card type, except as configured by the user in the **CreditCardTypeSetup** subform of the **CreditCardProcessors** form.

  In particular, note the following points:

  - In Microsoft Dynamics AX 2012, the displayed digits of a credit card number were the last five digits for American Express and the last four for the other card types. Additionally, when a CVV value was prompted, the value was required to be four digits for American Express and three for the other card types.
  - In Microsoft Dynamics AX 2012 R2, the displayed digits of a credit card number are provided by the connector, and when a CVV value is required, the only consideration is whether the value is blank or non-blank. As in Microsoft Dynamics AX 2012, determination of a valid CVV value is ultimately the responsibility of the credit card processor, and no check for correct length is made in Microsoft Dynamics AX 2012 R2. This means that any value with a length of one to four characters is accepted before the value is passed to the connector.

Appendix

The appendix covers these changed code resources:

- Enumerated type changes
- Extended data type (EDTs) changes
- Table changes

Changes not covered here include changes to forms and those resulting from functionality changes that are not directly related to credit card processing.

Enumerated type changes

Base enum	Reason for change
CreditCardAuthorizationResult	New type for an SDK API
CreditCardCaptureResult	New type for an SDK API
CreditCardMSAuthorizeReturnCode	DEL_
CreditCardMSSettlementReturnCode	DEL_
CreditCardMSVoidReturnCode	DEL_
CreditCardProcessor	DEL_
CreditCardPropertyDataType	New type for an SDK payment property
CreditCardPropertySecurityLevel	New type for an SDK payment property
CreditCardRefundResult	New type for an SDK API
CreditCardType	DEL_
CreditCardVoidResult	New type for an SDK API
DEL_CreditCardMSAuthorizeReturnCode	Deprecated; use the **CreditCardAuthorizationResult** enum.
DEL_CreditCardMSSettlementReturnCode	Deprecated; use the **CreditCardCaptureResult** enum.
DEL_CreditCardMSVoidReturnCode	Deprecated; use the **CreditCardVoidResult** enum.
DEL_CreditCardProcessor	Deprecated; use the **CreditCardProcessorId** EDT.
DEL_CreditCardType	Deprecated; use the **CreditCardTypeName** EDT.
FalseTrue	New type for an SDK API and a table field

Extended data type changes

Extended data type	Reason for change
CreditCardAccountConnectorProperties	New type for a table field
CreditCardConnectorName	New type for an SDK API and a table field
CreditCardCountryCodeIso	New type for an SDK API
CreditCardExpiryDate	Length change for four-digit years
CreditCardIsActive	New type for a table field
CreditCardIsTestMode	New type for a table field
CreditCardMicrosoftAuthorityPolicy	DEL_

Extended data type	Reason for change
CreditCardMicrosoftDNSName	DEL_
CreditCardMicrosoftEmailAddress	DEL_
CreditCardMicrosoftEnvironment	DEL_
CreditCardMicrosoftOrganizationId	DEL_
CreditCardMicrosoftPassword	DEL_
CreditCardMicrosoftServiceID	DEL_
CreditCardMicrosoftServiceUrl	DEL_
CreditCardPaymentCardToken	New type for an SDK API and a table field
CreditCardProcessorNotUsed	New type for a table field
CreditCardProcessorsAccessKey	DEL_
CreditCardProcessorsId	New type for a table field
CreditcardProcessorsName	New type for a table field
CreditCardProcessorsSecurityRefRecId	DEL_
CreditCardPropertyIsEncrypted	New type for an SDK API
CreditCardPropertyIsPassword	New type for an SDK API
CreditCardPropertyIsReadonly	New type for an SDK API
CreditcardPropertySequenceNumber	New type for an SDK API
CreditCardProviderResultOK	New type for internal use
CreditCardSecureCardData	DEL_
CreditCardTypeName	New type for an SDK API
CreditCardUniqueCardId	New type for an SDK API
DEL_CreditCardMicrosoftAuthorityPolicy	Deprecated
DEL_CreditCardMicrosoftDNSName	Deprecated
DEL_CreditCardMicrosoftEmailAddress	Deprecated
DEL_CreditCardMicrosoftEnvironment	Deprecated
DEL_CreditCardMicrosoftOrganizationId	Deprecated
DEL_CreditCardMicrosoftPassword	Deprecated
DEL_CreditCardMicrosoftServiceID	Deprecated
DEL_CreditCardMicrosoftServiceUrl	Deprecated
DEL_CreditCardProcessorsAccessKey	Deprecated
DEL_CreditCardProcessorsSecurityRefRecId	Deprecated
DEL_CreditCardSecureCardData	Deprecated

Table changes

Table	Reason for change
CreditCardAccountSetup	New; replaces CreditCardMicrsosoftSetup
CreditCardAuthTrans	Field changes

CreditCardCust	Field changes; includes a processor reference
CreditCardCustNumber	DEL_
CreditCardMicrosoftSetup	DEL_
CreditCardProcessors	Key is a string; it was previously an enum.
CreditCardProcessorsSecurity	DEL_
CreditCardTypeSetup	Card type is a string; it was previously an enum.
DEL_CreditCardCustNumber	Deprecated
DEL_CreditCardMicrosoftSetup	Deprecated
DEL_CreditCardProcessorsSecurity	Deprecated
SalesTable	Field changes

Microsoft Dynamics is a line of integrated, adaptable business management solutions that enables you and your people to make business decisions with greater confidence. Microsoft Dynamics works like and with familiar Microsoft software, automating and streamlining financial, customer relationship and supply chain processes in a way that helps you drive business success.

U.S. and Canada Toll Free 1-888-477-7989

Worldwide +1-701-281-6500

www.microsoft.com/dynamics

Microsoft