Oracle® MICROS Simphony

Loyalty/Stored Value Command Module Interface Reference Guide

Release 2.6 and Later
F75916-01
March 2023

ORACLE

Oracle MICROS Simphony Loyalty/Stored Value Command Module Interface Reference Guide, Release 2.6 and Later

F75916-01

Contents

Preface

Purpose

This document provides the information necessary to develop a communications interface between the third-party loyalty/stored value vendor and the Oracle MICROS Simphony system for the purpose of processing transactions.

Audience

This document is intended for developers, installers, and MIS or IT personnel.

Third Party Driver Package Distribution for Vendors

If a third-party has developed a driver, use the driver configuration settings provided by the third-party. Third party drivers may have unique fields. Complete the steps beginning on page D-1 of **Appendix D** for each third-party driver.

Customer Support

To contact Oracle Customer Support, access My Oracle Support at the following URL:

https://support.oracle.com

When contacting Customer Support, please provide the following:

- Product version and program/module name
- Functional and technical description of the problem (include business impact)
- Detailed step-by-step instructions to re-create
- Exact error message received and any associated log files
- Screenshots of each step you take

Documentation

Product documentation is available on the Oracle Help Center at https://docs.oracle.com/en/industries/food-beverage/pos.html

Revision History

Date	Description of Change
March 2023	Initial publication.

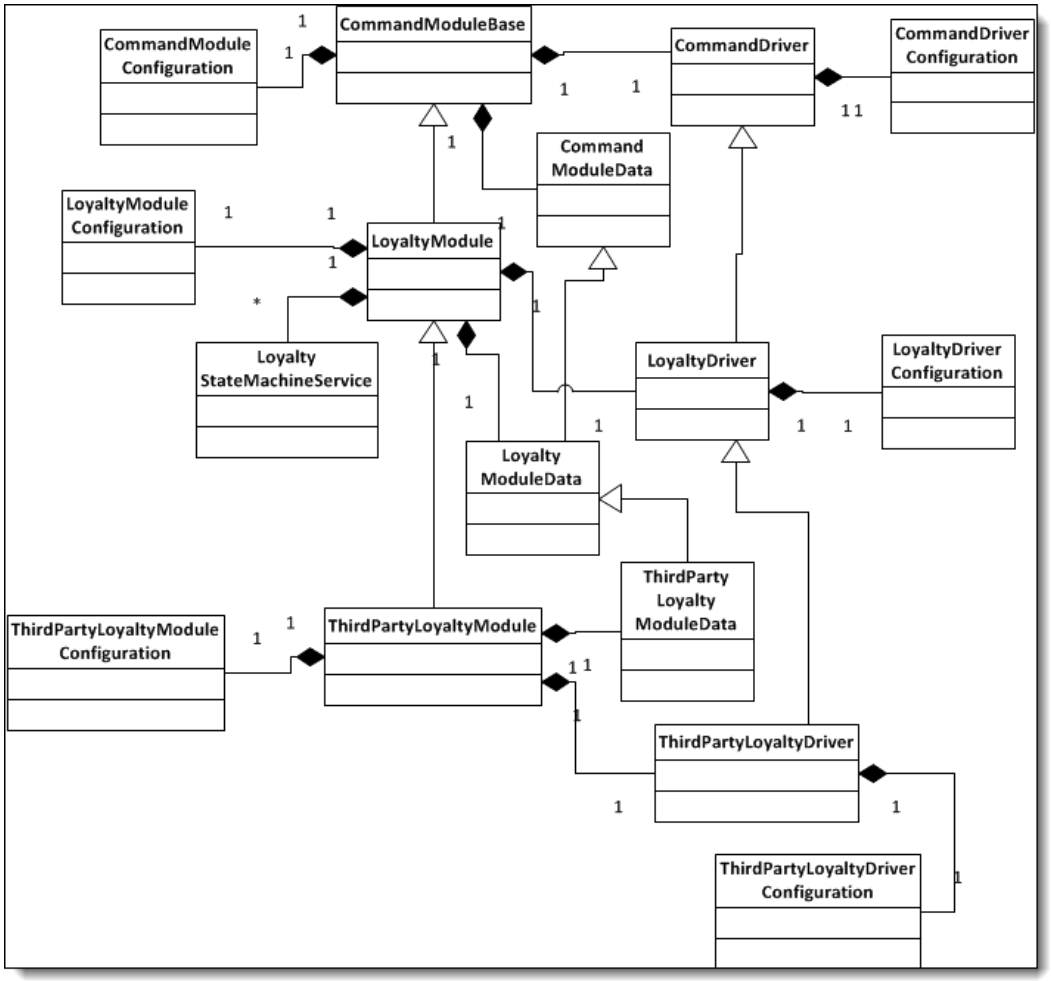
1

Loyalty/Stored Value Command Module Interface

Command Module Architecture

Command modules consist of a module/driver pair inherited from the following class hierarchy. The module contains business logic, while the driver communicates with a third-party provider via some communications mechanism (for example, a port or a web service). The command module layer is the base class of all command modules and provides module loading and configuration functionality. All command modules can be dynamically loaded and configured through the Enterprise Management Console (EMC) within Simphony.

Figure 1-1 - Loyalty/SVC Architecture Diagram



The second layer is the functional layer (either Loyalty or Stored Value). All of the business logic and display logic for data collection and check manipulation is found here. The Loyalty and Stored Value modules both have the same structure.

The third layer is the Third-Party layer, which is the first non-abstract layer in the architecture. This is the layer third parties will implement to provide their own driver functionality. The third-party module is very thin; most of the business logic for the module already exists in the Functional layer. There are some places where screens or printouts can be formatted specially via overrides, but the logic flow is fixed by Simphony requirements. The third-party driver, however, contains most of the driver functionality since the entire communications process is left to the third-party.

2

Specifications

Error Response Formatting

All operations display an error if the driver returns a Failure from the internal operation method. Set a CommandModuleException in the LastError property of the Data object with a CommandModuleError.DriverError and include an appropriate Message.

“%s –‘ %s’”, CommandModuleError.DriverError, LastError.Message Usually this formats to something like “Driver Error – ‘Port not found.’”

Display Results Formatting

Many operations displays the results of the operation to the user and optionally allow a version of the results to be printed. Generally, the state machine logic formats the results for the screen and printer so that all drivers present the same UI in Simphony. However, it is possible to override the display and printer if needed.

Standard Display Results

For any operation that supports a Display Results state, the standard method allows the state machine logic to build the display body, while the driver has the option of providing a header and a footer. The third-party driver can set the following properties:

- UIDisplayMessageHeader: header lines that are displayed before the body
- UIDisplayMessageFooter: footer lines that are displayed after the body
- PrinterMessageHeader: header lines that are printed before the body
- PrinterMessageFooter: footer lines that are printed after the body

This is the recommended method for customizing screens and printouts.

Overridden Display Results

The third-party driver may override the entire message displayed to the user or printed by setting the following properties:

- UIDisplayMessageOverride: the entire message displayed to the user for the operation
- PrinterMessageOverride: the entire message printed for the operation

Account Number Lookup

If the customer cannot locate their stored value card, the system allows the account number to be looked up by either Guest Name or Phone Number. Some operations allow only one or the other, while some operations allow neither. Each operation defines which lookups are valid for that operation. If a lookup is enabled for the operation, the account number may be a phone number or guest name lookup string.

The AccountNumberContentValue property indicates this, as described in the account number data discussion in **Appendix A** on page A-3.

If after processing, the driver determines that the guest name or phone number chosen is ambiguous (that is, cannot be used to pick one and only one account number), it may return UserInputRequired to instruct the state machine to transition to the Select Customer Name State. The driver must load the account numbers and other information for the selection state to display to the user. In the Loyalty Module use the LookupResults array, while for the Stored Value module use the AccountLookupDetailsResults. The state machine displays the lookup results and allows the user to pick one account number from the list. The operation state calls the driver again with the same collected data, but with the new filled-in account number.

If the driver does not support account or guest name lookup, disable the Guest Name and Phone Number lookup buttons by having users set the Remove Guest Name Lookup Button and Remove Phone Number Lookup Button value to True in the Loyalty or Stored Value settings. If the driver returns a Failure instead of UserInputRequried, the lookup screen is never presented to the user. The operation just fails without an account number.

Offline Mode

Offline mode allows transactions under a floor limit to automatically succeed without contacting the Loyalty or Stored Value provider. In order for offline mode to be active, the Enable Offline Support option in the Loyalty and Stored Value settings must be enabled. Disabling this option turns off offline support entirely.

If offline mode has been enabled and the driver returns a CommunicationsFailure result for an operation, the system shifts into offline mode for the following operations:

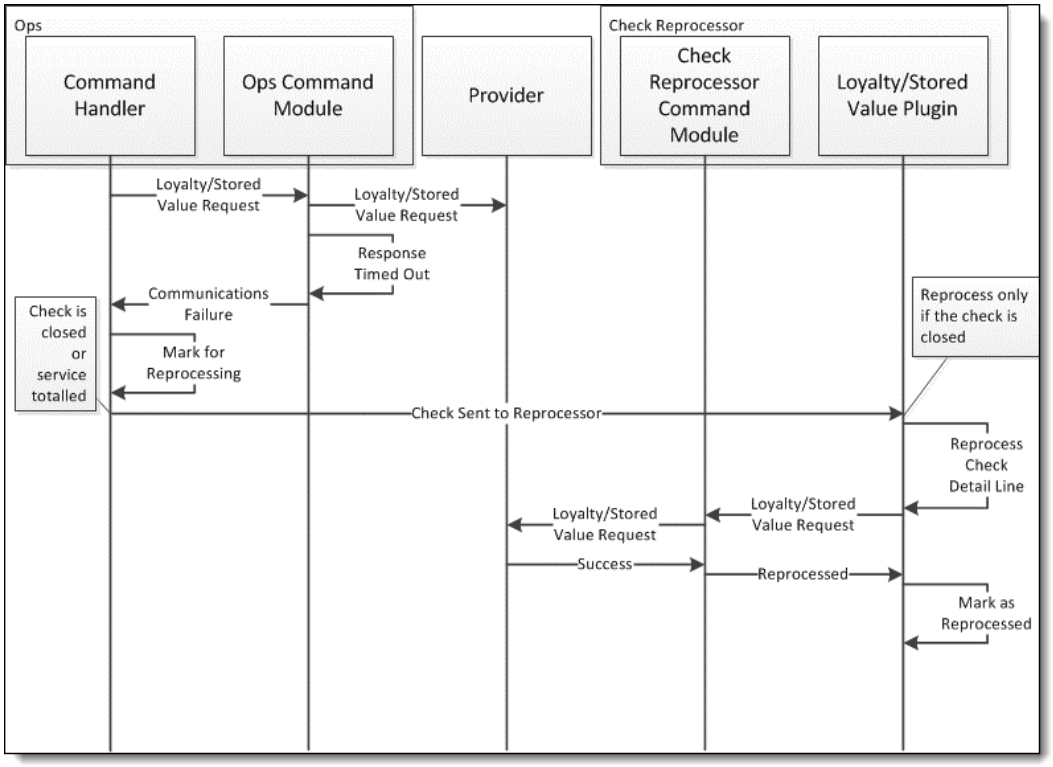
- Stored Value Authorization
- Stored Value Void Authorization
- Stored Value Redemption
- Stored Value Void Redemption
- Stored Value Reload
- Stored Value Void Reload
- Loyalty Issue Points
- Loyalty Void Issue Points
- Accept Coupon
- Void Accept Coupon

Logic Flow

If the driver does not support offline for any of these operations, ensure that it never returns CommunicationsFailure, but Failure as the result of an operation. Returning Failure will fail the transaction but not trigger offline mode.

When in offline mode, any transaction from the above list that is under the floor limit is blindly accepted as successful. Once the check is closed it is sent to the check reprocessor, where the offline transactions are attempted repeatedly until it either succeeds or the number of retries is exceeded. The following diagram shows the flow for this logic:

Figure 2-1 - Logic Flow



Notice that there are two copies of the Command Module: one in Ops and one in the Check Reprocessor. These are both instances of the Loyalty or Stored Value module. The module is asked to process the transaction using the Module Data that was serialized onto the check. There is no difference in the call from the state machine to the module and driver whether it has been instantiated in Ops or in the Check Reprocessor.

There is no need for special driver code to handle the Check Reprocessor. In fact, the driver is not aware that it is running in the Check Reprocessor and not Ops.

There is, however, a need for the driver to determine how it handles being in an offline state. If the communications timeout is set for 30 seconds, then every failed attempt at communication while offline halts Simphony for 30 seconds while it waits for a response. Because of this, most drivers implement some type of method to attempt communications while offline without delaying the main thread for every attempt. The iCare driver goes into offline mode when communication fails, and then keeps a user-configurable counter

in the iCare Driver Configuration to determine how many attempts to skip while online. For example, by default the iCare driver only attempts to communicate every 5 transactions while offline. Once one transaction succeeds, it shifts back into online mode, communicating for every transaction. The implementation of this mechanism is entirely the purview of the driver. Other methods might include timers or ping systems, depending on the communications media used.

Data Model

In offline mode the following properties are used by the state machine to handle adding items to the check. These values are visible in the Driver Configuration in EMC, but they are read-only on that screen. They need to be populated by querying them from the provider, hard coding them, or having them update hourly using the Data Transfer Service, described in the following section.

Table 2-1 – Data Flow

Property	Type	Content	Description
OfflineFloorLimit	Decimal	Special	The minimum value of a check item that will not be processed offline.
OfflineCouponCeiling	Decimal	Special	The maximum value of a coupon that will be processed offline.
OfflinePointsIssueTender	String	Special	Unused
OfflineAcceptCouponTender	String	Special	A check detail item to be placed on the check when a coupon is
OfflineRedemptionTender	String	Special	A check detail item to be placed on the check when an SVC is redeemed.
OfflineRedemptionAuthTender	String	Special	A check detail item to be placed on the check when an SVC is authorized.
OfflineReloadTender	String	Special	A check detail item to be placed on the check when an SVC is reloaded.

The tender strings are of the format: “{0}-{1}”, ItemType, ObjectNumber where:

- ItemType is one of the following letters:
  - T – the item is a Tender (Payment)
  - M – the item is a Menu Item (not supported by Simphony)
  - S – the item is a Service Charge
  - D – the item is a Discount
- ObjectNumber is the object number of the item to be placed on the check.

Data Transfer Service

These offline tenders allow the state machine to put an item on the check while communications with the provider are offline. They need to be read from the provider while communications are up, so that they are available when communications are down.

The Data Transfer Service is a Windows service that runs a job called Command Module Maintenance. This job runs every hour by default and instantiates a copy of the module/driver combination. It then calls the PerformDTSMaintenance() method on the driver. The driver should use this method to communicate with the provider service and request the floor limits and tenders, format them to the above-mentioned format, and write them into the Driver Configuration. If the method returns Success, then the driver is indicating that the data have changed and should be persisted to the database. If the driver returns Failure, no changes are made to the configuration in the database.

3 Loyalty Specifications

The Loyalty functional layer provides data collection and check manipulation for each of the Loyalty Operations. The following operations are supported:

- Issue Points
- Void Issue Points
- Request Available Balance
- Request Available Coupons
- Accept Coupon
- Void Accept Coupon
- Issue Coupon

For each of the non-void operations, the Loyalty Module queries the user for all the required data, fills in more data from the check and the Simphony Ops environment, and finally calls the driver to perform the operation. The data is passed around in the LoyaltyModuleData object. When the driver retrieves the result, it places the result in the LoyaltyModuleData object using the provided Data Model. The Loyalty Module then interprets the Data Model and displays and/or prints the results and places check items on the check as requested by the driver.

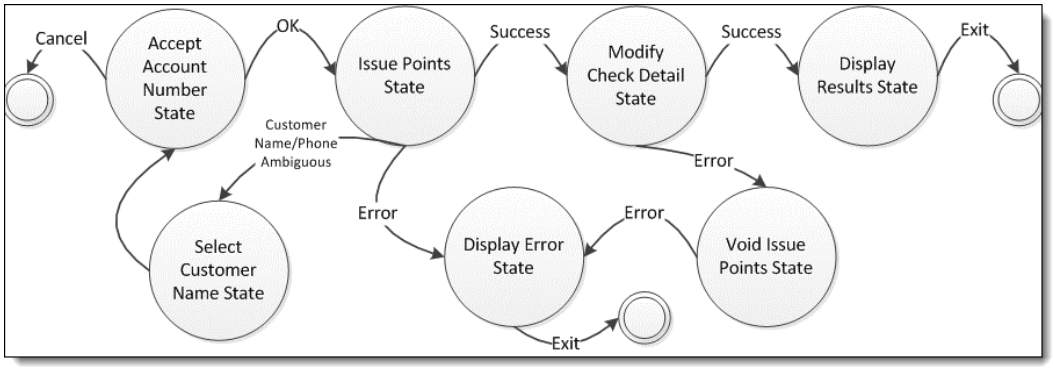
Before each request the base classes populate the LoyaltyModuleData object with general data. All the data is described in the Appendices of this document. This general loyalty data is given below:

- The bits from the LoyaltySettings are copied into the data object for easy access: SupportOfflineTransactions, AlwaysPrintLoyaltyResults, NeverDisplayLoyaltyResults, RemoveLoyaltyResultsPrintButton, RemoveGuestNameLookupButton, RemovePhoneNumberLookupButton. Most of these are handled by the module business logic, but they are also visible to the driver.
- Some generic Ops data is populated: TerminalID, TerminalType, BusinessDate, LocalCurrency, Language, LocalDateTime, TransactionEmployee, RevenueCenter.
- The check data is summarized for the loyalty provider. The following check summary information is populated: CheckNumber, IsCheckActive, CheckSummaryCheckOpenTime, CheckDetailMenuItem, CheckDetailSalesItemizer, CheckDetailDiscount, CheckDetailServiceCharge, CheckDetailPayment, CheckSummaryTotalsDiscounts, CheckSummaryTotalsAmountDue, CheckSummaryTotalsPayments, CheckSummaryTotalsServiceCharges, CheckSummaryTotalsTaxes.

Issue Points

The issue points operation submits a check to the loyalty service for evaluation and reward of points. This operation also allows items to be added to the check if requested by the loyalty service. The issue points state machine is described below.

Figure 3-1 - Issue Points Operations



- **Accept Account Number State**
  This state displays the account number entry screen, allowing the user to enter either an account number or phone number (if phone number lookup is allowed). If the user presses Cancel, the state machine exits immediately. Fields populated in the Data object:
  - AccountNumber
  - AccountNumberEntry
  - Track1-Track4 (if swiped)
  - AccountNumberMasked
  - PrintableAccountNumberMasked

Select Customer Name State
The state machine transitions to this state when the driver returns UserInputRequired for the issue points operation and the LookupResults property has been populated with customer name/account number pairs. See the section titled

- Account Number Lookup for more information.
- **Retrieve Patron List State**
  This state calls the LookUpPatronInternal() operation after the LookUp has been performed by the driver. The data returned includes the account holder's name (GuestName) or phone number (PhoneNumber). See the section titled More Detail on the Response Properties for additional information.

  The only data that must be returned is GuestName. Simphony uses this data to attach the loyalty information to the guest check. If the account holder name is not available, you can substitute any value, such as the last 4 digits of the account number. It is important that this value is not null.

- **Select Patron State**
  This state instructs the driver to perform the LoyaltyAccountPatronInternal(). This operation allows the system to show a list of patrons when more than one patron is returned from the lookup request.
- **Request Loyalty POS Options State**
  This state calls the driver to perform the LoadConfigurationInternal() operation. The data available includes optional settings that the provider may return, such as Prompt for Amount, Encrypt PIN, Program Code, Prompt for Coupon, Prompt for PIN, and the Request code.
- **Issue Points State**
  This state calls the driver to perform the IssuePointsInternal() operation. The available data includes the general loyalty data, as well as whatever the Account Number Entry State collected.
  - On a success, return Success and populate the LocalBalance list with a list of point balances, one per program. The state machine uses this list to build the on-screen points issued/point totals that are displayed to the user.
  - On a failure, return Failure and populate the LastError with an exception representing the error case. Use the exception message to add text to the error.
  - If there is a communications failure and you want the system to shift into offline mode, return CommunicationFailure. See the section titled Offline Mode for more information.
- **Modify Check Detail State**
  The issue points operation is added to the check as a detail item so that it may be later voided. The LoyaltyModuleData object is serialized onto the check. If there is any information the driver needs for a void, make sure it is stored in the leaf-class LoyaltyModuleData and it is serialized.
  - The driver may also request that other check detail items be added to the check. It does this by populating the LoyaltyPostingItems list.

> **NOTE:**
>
> If an item cannot be added to the check, the entire issue points operation fails, and the issue points operation is auto-voided (that is, Void Issue Points are called automatically).

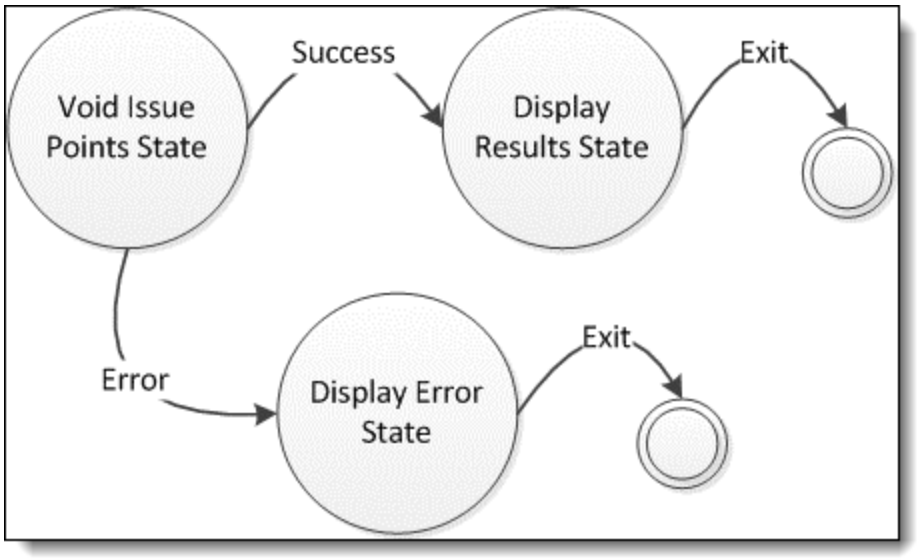
- **Void Issue Points St**ate
  This state is used for an auto-void. The LoyaltyModuleData object that was serialized onto the check is given to the driver in order to perform the void.
  - On a success, return Success. An error appears indicating that the coupon could not be added to the check.
  - On a failure, return Failure. An error appears indicating that the coupon could not be added to the check. The fact that the auto-void failed is logged.
  - If there is a communications failure and you want the system to shift into offline mode, return CommunicationFailure. See the section titled Offline Mode for more information.

- **Display Results State**
  The state machine builds the body of the message displaying or printing the balances that were returned. The driver may customize this message with a header and footer as normal.
- **Display Error State**
  If the operation failed, the LastError is displayed to the user, and nothing is placed on the check.

Void Issue Points

The void issue points operation voids loyalty points that have been issued based upon a check. The void issue points state machine is described below.

Figure 3-2 - Void Issue Points



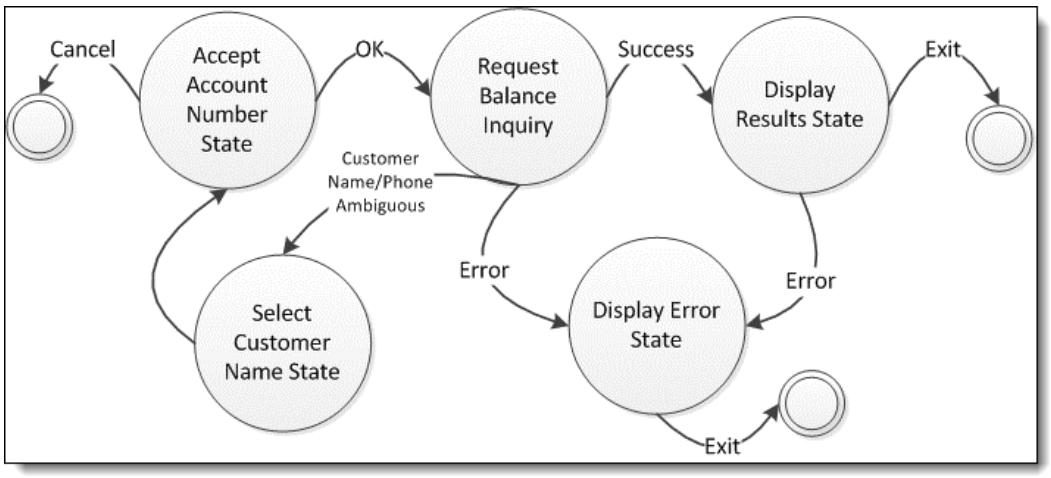
- **Void Issue Points State**
  This state calls the driver to perform the VoidIssuePointsInternal() operation. The available data includes all the data that was serialized onto the check (which is the entire Data object).
  - On a success, return Success and populate the LocalBalance list with a list of point balances, one per program. The state machine uses this list to build the on-screen issue points voided message that is displayed to the user.
  - On a failure, return Failure and populate the LastError with an exception representing the error case. Use the exception message to add text to the error.
  - If there is a communications failure and you want the system to shift into offline mode, return CommunicationFailure. See the section titled Offline Mode for more information.
- **Display Results State**
  The state machine builds the body of the message displaying or printing the balances that were returned. The driver may customize this message with a header and footer as normal.

- **Display Error State**
  If the operation failed, the LastError is displayed to the user, and nothing is placed on the check.

Request Available Balance

The request available balance operation requests program balances from the loyalty service for a given account number. The request available balance state machine is described below.

Figure 3-3 - Request Available Balance



- **Accept Account Number State**
  This state displays the account number entry screen allowing the user to enter either an account number or phone number/guest name (if phone number/guest name lookup is allowed). If the user presses Cancel, the state machine exits immediately. Fields populated in the Data object:
  - AccountNumber
  - AccountNumberEntry
  - Track1-Track4 (if swiped)
  - AccountNumberMasked
  - PrintableAccountNumberMasked

Select Customer Name State
The state machine transitions to this state when the driver returns UserInputRequired for the request available balance operation, and the LookupResults property has been populated with customer name/account number pairs. See the section titled

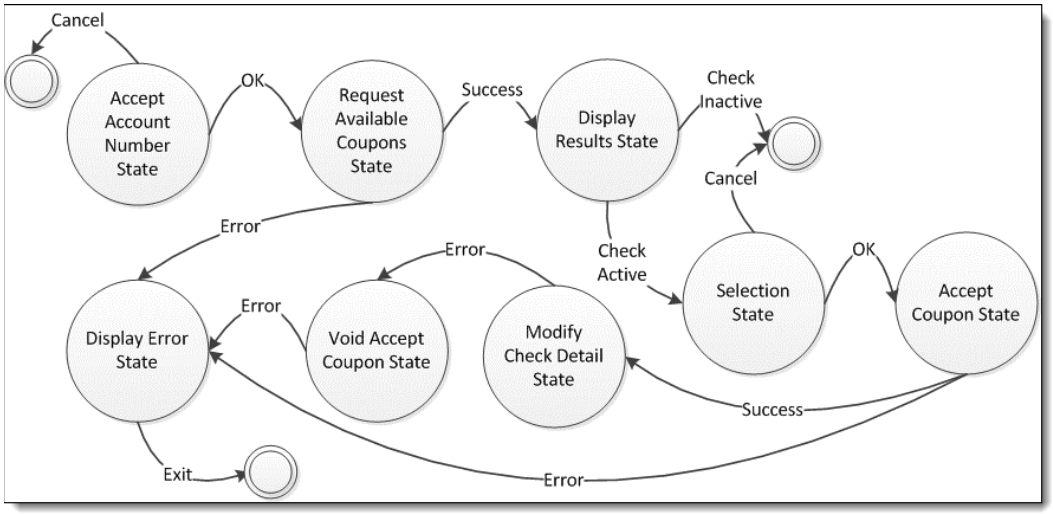
- Account Number Lookup for more information.
- **Retrieve Patron List State**
  This state calls the LookUpPatronInternal() operation after the LookUp has been performed by the driver. The data returned includes the account holder's name (GuestName) or phone number (PhoneNumber). See the section titled More Detail on the Response Properties for additional information.

  The only data that must be returned is GuestName. Simphony uses this data to attach the loyalty information to the guest check. If the account holder name is not available, you can substitute any value, such as the last 4 digits of the account number. It is important that this value is not null.
- **Select Patron State**
  This state instructs the driver to perform the LoyaltyAccountPatronInternal(). This operation allows the system to display a list of patrons when more than one patron is returned from the Lookup request.
- **Request Loyalty POS Options State**
  This state calls the driver to perform the LoadConfigurationInternal() operation. The data available includes optional settings that the provider may return, such as Prompt for Amount, Encrypt PIN, Program Code, Prompt for Coupon, Prompt for PIN, and the Request code.
- **Request Available Balance**
  This state calls the driver to perform the PointBalanceInquiryInternal() operation. The available data includes the general loyalty data, as well as whatever the Account Number Entry State collected.
  - On a success, return Success and populate the LocalBalance list with a list of point balances, one per program. The state machine uses this list to build the on-screen points issued/point totals that are displayed to the user.
  - On a failure, return Failure and populate the LastError with an exception representing the error case. Use the exception message to add text to the error.
- **Display Results State**
  The state machine builds the body of the message displaying or printing the balances that were returned. The driver may customize this message with a header and footer as normal.
- **Display Error State**
  If the operation failed, the LastError is displayed to the user, and nothing is placed on the check.

Request Available Coupons

The request available coupon operation requests coupons from the loyalty service for a given account number. The request available coupon state machine is described below.

Figure 3-4 - Request Available Coupons



- **Accept Account Number State**
  This state displays the account number entry screen allowing the user to enter an account number. If the user presses Cancel, the state machine exits immediately. Fields populated in the Data object:
  - AccountNumber
  - AccountNumberEntry
  - Track1-Track4 (if swiped)
  - AccountNumberMasked
  - PrintableAccountNumberMasked
- **Request Available Coupons**
  This state calls the driver to perform the CouponInquiryInternal() operation. The available data includes the general loyalty data, as well as whatever the Account Number Entry State collected.
  - On a success, return Success and populate the LoyaltyActionItems list with a list of coupons. If a check is not active, the module uses this list to build a display and present it to the user. If the check is active, the module uses this list to build a selection list that allows the user to choose one of the coupons and add it to the check.
  - On a failure, return Failure and populate the LastError with an exception representing the error case. Use the exception message to add text to the error.
- **Retrieve Patron List State**
  This state calls the LookUpPatronInternal() operation after the LookUp has been performed by the driver. The data returned includes the account holder's name

(GuestName) or phone number (PhoneNumber). See the section titled More Detail on the Response Properties for additional information.

The only data that must be returned is GuestName. Simphony uses this data to attach the loyalty information to the guest check. If the account holder name is not available, you can substitute any value, such as the last 4 digits of the account number. It is important that this value is not null.

- **Select Patron State**
  This state instructs the driver to perform the LoyaltyAccountPatronInternal(). This operation allows the system to display a list of patrons when more than one patron is returned from the Lookup request.
- **Request Loyalty POS Options State**
  This state calls the driver to perform the LoadConfigurationInternal() operation. The data available includes optional settings that the provider may return, such as Prompt for Amount, Encrypt PIN, Program Code, Prompt for Coupon, Prompt for PIN, and the Request code.
- **Selection State**
  This state presents the coupon list to the user and allows them to choose one to apply to the check. If the user presses OK, the selected action item is placed in the SelectedActionItem of the Data object and that action item's Value property is placed in the CouponCode property of the Data object. If the user presses Cancel, the state machine exits.
- **Accept Coupon State**
  If the user chooses a coupon to apply to the check from the Selection State, this state calls the driver AcceptCouponInternal() operation. The Selection State will have populated the Data object with the appropriate information to perform this operation.
  - On a success, return Success, and place the item to be placed on the check for the coupon by filling in the ItemType and ItemNumber pair on the Data object.
  - On a failure, return Failure and populate the LastError with an exception representing the error case. Use the exception message to add text to the error.
- **Modify Check Detail State**
  Modifies the check detail by applying the coupon to the check. The item that is added to the check is described by the ItemType and ItemNumber pair found in the Data object.
- **Void Accept Coupon State**
  If the coupon has already been accepted by the loyalty system, but the modify check detail failed, this state voids the Accept Coupon operation by calling the VoidAcceptCouponInternal() operation on the driver. The available data includes all the data that was serialized onto the check (which is the entire Data object).
  - On a success, return Success. An error indicating that the coupon could not be added to the check appears.
  - On a failure, return Failure. An error indicating that the coupon could not be added to the check appears. The fact that the auto-void failed is logged.
- **Display Results State**
  If the check is not active, the state machine builds the body of the message displaying or printing the coupons that were returned. If the check is active, the state

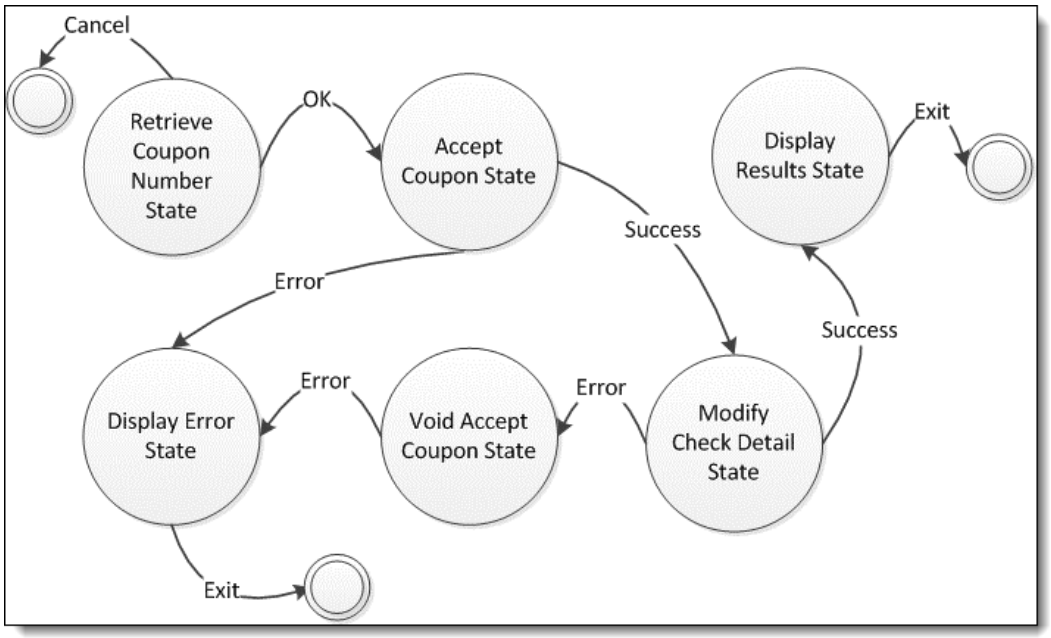
machine builds a message indicating the acceptance of the coupon was successful. The driver may customize either message with a header and footer as normal.

- **Display Error State**
  If the operation failed, the LastError is displayed to the user, and nothing is placed on the check.

Accept Coupon

The accept coupon operation allows the user to enter a coupon code to apply a coupon to the check. The accept coupon state machine is described below.

Figure 3-5 - Accept Coupon



- **Retrieve Coupon Number State**
  This state displays the coupon number entry screen, allowing the user to enter an alphanumeric coupon code. If the user presses Cancel, the state machine exits immediately. Fields populated in the Data object:
  - CouponCode
- **Accept Coupon State**
  If the user chooses a coupon to apply to the check from the Selection State, this state calls the driver AcceptCouponInternal() operation. The available data is the standard loyalty data plus the CouponCode.
  - On a success, return Success and place the item to be placed on the check for the coupon by filling in the ItemType and ItemNumber pair on the Data object.
  - On a failure, return Failure and populate the LastError with an exception representing the error case. Use the exception message to add text to the error.

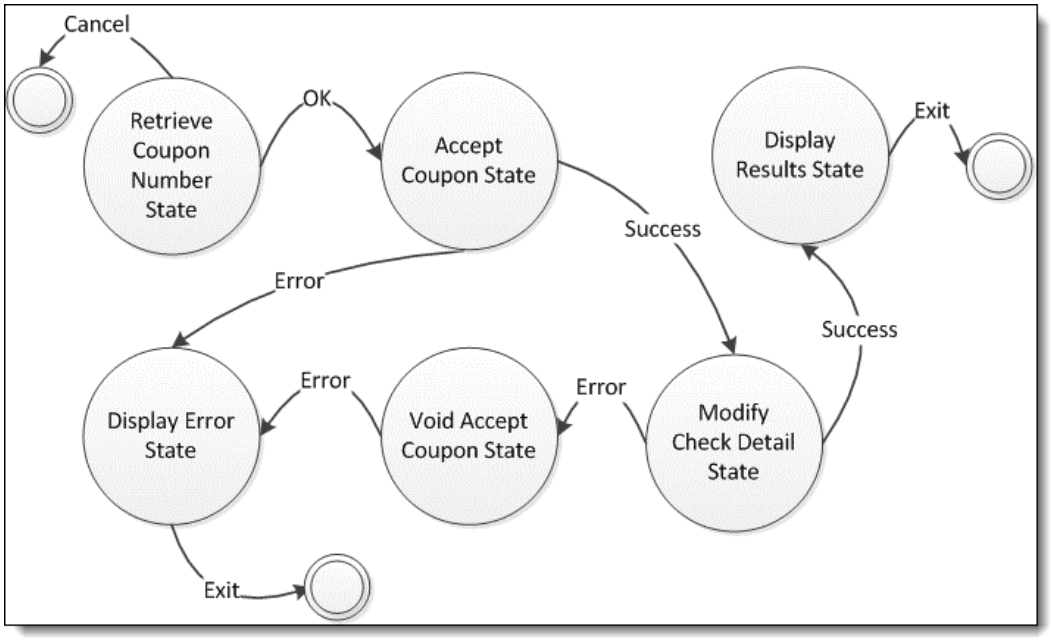
- If there is a communication failure, and you want the system to shift into offline mode, return CommunicationFailure. See the section titled Offline Mode for more information.

- **Modify Check Detail State**
  Modifies the check detail by applying the coupon to the check. The item that is added to the check is described by the ItemType and ItemNumber pair found in the Data object.
- **Display Results State**
  The state machine builds a message indicating the acceptance of the coupon was successful. The driver may customize the message with a header and footer as normal.
- **Display Error State**
  If the operation failed, the LastError is displayed to the user and nothing is placed on the check.

Void Accept Coupon

The void accept coupon operation allows the user to void a coupon that has been applied to the check. The void accept coupon state machine is described below.

Figure 3-6 - Void Accept Coupon



- **Void Accept Coupon State**
  This state calls the driver VoidAcceptCouponInternal() operation.
  - On a success, return Success.
  - On a failure, return Failure and populate the LastError with an exception representing the error case. Use the exception message to add text to the error.

- If there is a communication failure and you want the system to shift into offline mode, return CommunicationFailure. See the section titled Offline Mode for more information.

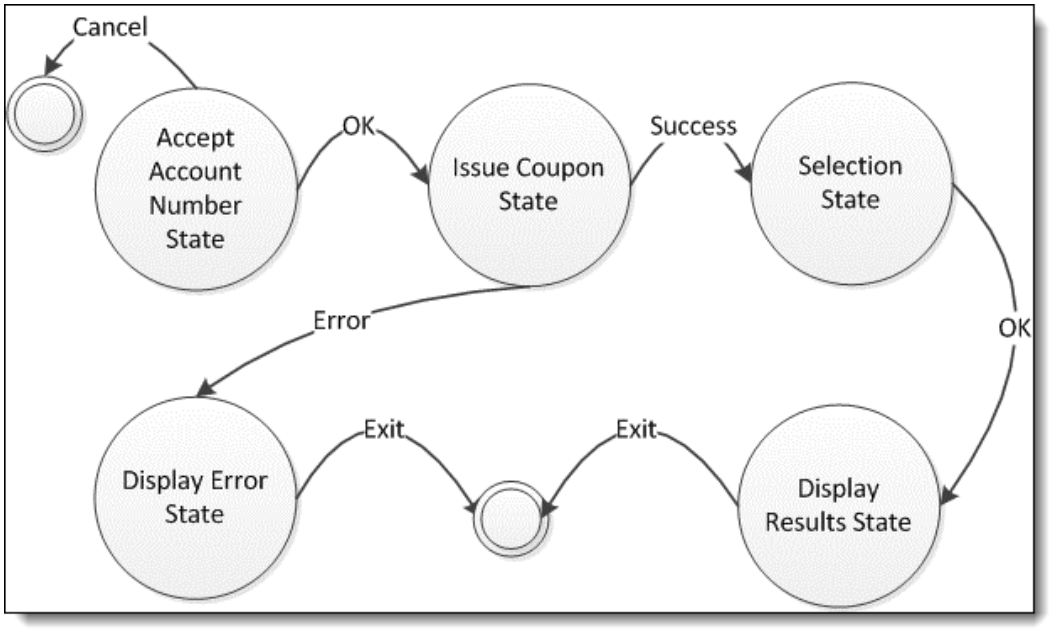
- **Display Results State**
  The state machine builds a message indicating the coupon was voided. The driver may customize the message with a header and footer as normal.

- **Display Error State**
  If the operation failed, the LastError is displayed to the user, and nothing is placed on the check.

Issue Coupon

The issue coupon operation allows a manager to issue coupons to a customer by printing them out. The issue coupon state machine is described below.

Figure 3-7 - Issue Coupon



- **Accept Account Number State**
  This state displays the account number entry screen, allowing the user to enter an account number. If the user presses Cancel, the state machine exits immediately. Fields populated in the Data object:

  - AccountNumber
  - AccountNumberEntry
  - Track1-Track4 (if swiped)
  - AccountNumberMasked
  - PrintableAccountNumberMasked

- **Issue Coupon State**
  This state calls the driver to perform the IssueCouponInternal() operation. The available data includes the general loyalty data, as well as whatever the Account Number Entry State collected.
  - On a success, return Success and populate the LoyaltyActionItems list with a list of coupons available to be issued.
  - On a failure, return Failure and populate the LastError with an exception representing the error case. Use the exception message to add text to the error.
- **Retrieve Patron List State**
  This state calls the LookUpPatronInternal() operation after the LookUp has been performed by the driver. The data returned includes the account holder's name (GuestName) or phone number (PhoneNumber).

  See the section titled More Detail on the Response Properties for additional information.

  The only data that must be returned is GuestName. Simphony uses this data to attach the loyalty information to the guest check. If the account holder name is not available, you can substitute any value, such as the last 4 digits of the account number. It is important that this value is not null.
- **Select Patron State**
  This state instructs the driver to perform the LoyaltyAccountPatronInternal(). This operation allows the system to display a list of patrons when more than one patron is returned from the Lookup request.
- **Request Loyalty POS Options State**
  This state calls the driver to perform the LoadConfigurationInternal() operation. The data available includes optional settings that the provider may return, such as Prompt for Amount, Encrypt PIN, Program Code, Prompt for Coupon, Prompt for PIN, and the Request code.
- **Selection State**
  This state presents the coupon list to the user and allows them to choose one to issue to the customer. If the user presses OK, the selected action item is placed in the SelectedActionItem of the Data object and that action item's Value property is placed in the CouponCode property of the Data object. If the user presses Cancel, the state machine exits.
- **Display Results State**
  The state machine builds the body of the message displaying or printing the coupon that was issued. The coupon is formatted with a standard coupon header and the coupon code is printed as the coupon serial number. The driver may customize the message with a header and footer as normal.
- **Display Error State**
  If the operation failed, the LastError is displayed to the user, and nothing is placed on the check.

Stored Value Specification

The Stored Value functional layer provides data collection and check manipulation for each of the Stored Value Operations. The following operations are supported:

- Issue Card
- Void Issue Card
- Balance Inquiry
- Activate Card
- Void Activate Card
- Reload Card
- Void Reload Card
- Cash Out Card
- Authorize Card
- Void Authorize Card
- Redeem Card
- Void Redeem Card

For each of the non-void operations, the Stored Value Module queries the user for all the required data, fills in more data from the check and the Simphony Ops environment, and finally calls the driver to perform the operation. The data is passed around in the StoredValueModuleData object. When the driver retrieves the result, it places the result in the StoredValueModuleData object using the provided Data Model. The Stored Value Module then interprets the Data Model and displays and/or prints the results and places check items on the check as requested by the driver.

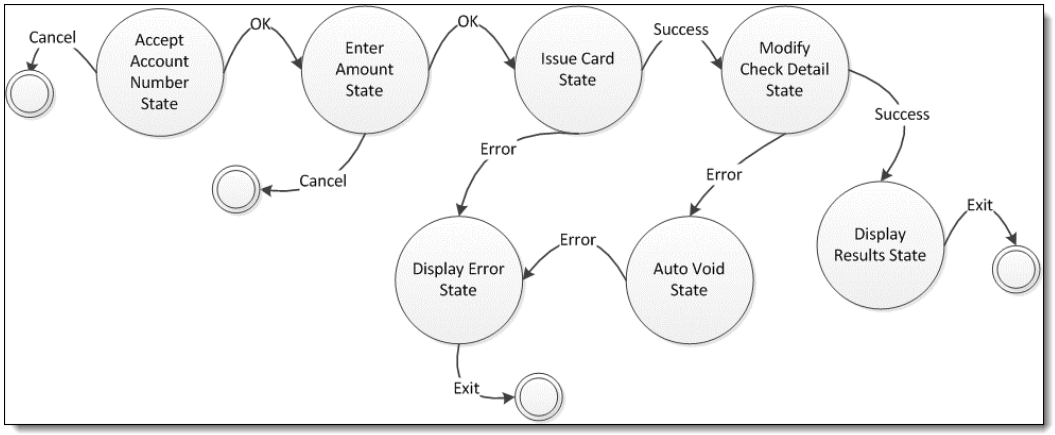
Before each request the base classes populate the StoredModuleData object with general data. All the data is described in the Appendices. This general stored value data is provided below:

- The bits from the StoredValueSettings are copied into the Data object for easy access: SupportOfflineTransactions, AlwaysPrintStoredValueResults, NeverDisplayStoredValueResults, ShowVoidOnGuestCheck, RemoveStoredValueResultsPrintButton, RemoveGuestNameLookupButton, and RemovePhoneNumberLookupButton. Most of these are handled by the module business logic, but they are visible to the driver as well.
- Some generic Ops data is populated: TerminalID, TerminalType, BusinessDate, LocalCurrency, Language, LocalDateTime, TransactionEmployee, and RevenueCenter.
- The check data is summarized for the stored value provider. The following check summary information is populated: IsCheckActive, CheckNumber, and CheckSummary.

Issue Card

The issue points operation issues a stored value card for a certain amount. The Issue Card state machine is described below.

Figure 3-8 - Issue Card



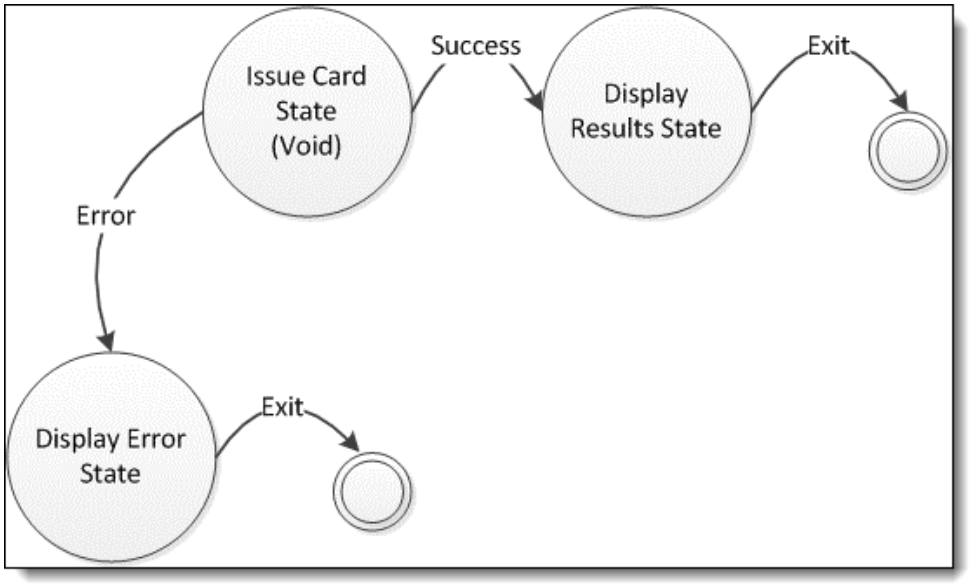
- **Account Number Entry State**
  This state displays the account number entry screen, allowing the user to enter an account number. If the user presses Cancel, the state machine exits immediately. Properties populated in the Data object:
  - AccountNumber
  - AccountNumberEntry
  - Track1-Track4 (if swiped)
  - AccountNumberMasked
  - PrintableAccountNumberMasked
- **Enter Amount State**
  This state displays a numeric entry pad to accept the amount to issue the card for. If the user presses Cancel, the state machine exits immediately. Properties populated in the Data object:
  - Amount
- **Issue Card State**
  This state calls the driver IssueCardInternal() operation. The available data includes the general stored value data, as well as whatever the Account Number Entry State and Enter Amount State collected.
  - On a success, return Success and populate the ProgramName and AccountBalance properties with the balance of the newly issued card. Populate the ItemType and ItemNumber fields with the item to place on the check representing the activation (usually a service charge). The state machine uses this list to build the success message that is displayed to the user.
  - On a failure, return Failure and populate the LastError with an exception representing the error case. Use the exception message to add text to the error.

- **Modify Check Detail State**
  The ItemType and ItemNumber properties are used to add an item to the check. The StoredValueModuleData object will be serialized onto the check.
- **Auto-Void State**
  This state is used for an auto-void, calling the VoidIssueCardInternal() operation on the driver to do so. The StoredValueModuleData object that was serialized onto the check is given to the driver in order to perform the void.
  - On a success, return Success.
  - On a failure, return Failure and populate the LastError with an exception representing the error case. Use the exception message to add text to the error.
- **Display Results State**
  The state machine builds the body of the message displaying or printing the balances that were returned. The driver may customize this message with a header and footer as normal.
- **Display Error State**
  If the operation failed, the LastError is displayed to the user and nothing is placed on the check.

Void Issue Card

The void issue card operation voids a stored value card issue. The Void Issue Card state machine is described below.

Figure 3-9 - Void Issue Card



- **Issue Card State (Void)**
  This state calls the driver to perform the VoidIssueCardInternal() operation. The available data includes the serialized StoredValueModuleData object on the issue card detail item that is voided.
  - On a success, return Success.

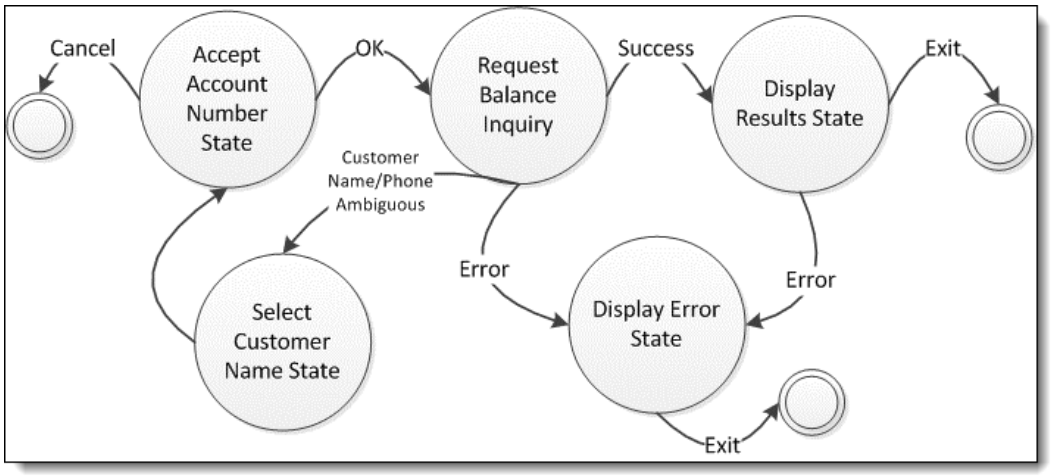
- On a failure, return Failure and populate the LastError with an exception representing the error case. Use the exception message to add text to the error.

- Display Results State. The state machine builds the body of the message displaying or printing the balances that were returned. The driver may customize this message with a header and footer as normal.
- Display Error State. If the operation failed, the LastError is displayed to the user, and nothing is placed on the check.

Request Available Balance

The request available balance operation requests program balances from the stored value service for a given account number. The request available balance state machine is described below.

Figure 3-10 - Request Available Balance



- **Accept Account Number State**
  This state displays the account number entry screen, allowing the user to enter either an account number or phone number/guest name (if phone number/guest name lookup is allowed). If the user presses Cancel, the state machine exits immediately. Fields populated in the Data object:
  - AccountNumber
  - AccountNumberEntry
  - Track1-Track4 (if swiped)
  - AccountNumberMasked
  - PrintableAccountNumberMasked
- **Select Customer Name State**
  The state machine transitions to this state when the driver returns UserInputRequired for the request available balance operation and the LookupResults property has been populated with customer name/account number pairs. See the section titled “Account Number Lookup” on page 10 for more information.

- **Retrieve Patron List State**
  This state calls the LookUpPatronInternal() operation after the LookUp has been performed by the driver. The data returned includes the account holder's name (GuestName) or phone number (PhoneNumber). See the section titled More Detail on the Response Properties for additional information.

  The only data that must be returned is GuestName. Simphony uses this data to attach the loyalty information to the guest check. If the account holder name is not available, you can substitute any value, such as the last 4 digits of the account number. It is important that this value is not null.

- **Select Patron State**
  This state instructs the driver to perform the LoyaltyAccountPatronInternal(). This operation allows the system to display a list of patrons when more than one patron is returned from the Lookup request.

- **Request Loyalty POS Options State**
  This state calls the driver to perform the LoadConfigurationInternal() operation. The data available includes optional settings that the provider may return, such as Prompt for Amount, Encrypt PIN, Program Code, Prompt for Coupon, Prompt for PIN, and the Request code.

- **Balance Inquiry State**
  This state calls the driver to perform the BalanceInquiryInternal() operation. The available data includes the general loyalty data, as well as whatever the Account Number Entry State collected.

  - On a success, return Success and populate the ProgramName and AccountBalance properties with a list of balances, one per program. The state machine uses this list to build the on- screen amount issued/balance totals that are displayed to the user.
  - On a failure, return Failure and populate the LastError with an exception representing the error case. Use the exception message to add text to the error.

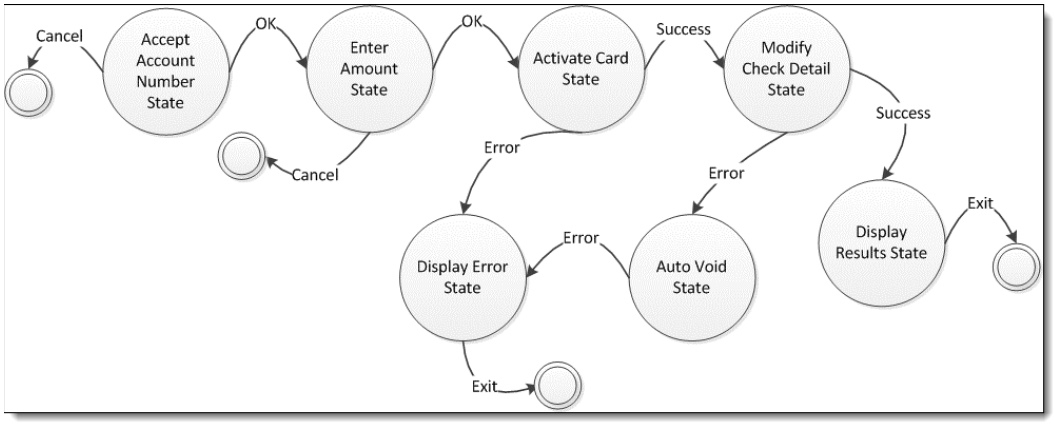
- **Display Results State**
  The state machine builds the body of the message displaying or printing the balances that were returned. The driver may customize this message with a header and footer as normal.

- **Display Error State**
  If the operation failed, the LastError is displayed to the user, and nothing is placed on the check.

Activate Card

The activate card operation activates a stored value card for a certain amount. The Activate Card state machine is described below.

Figure 3-11 - Activate Card



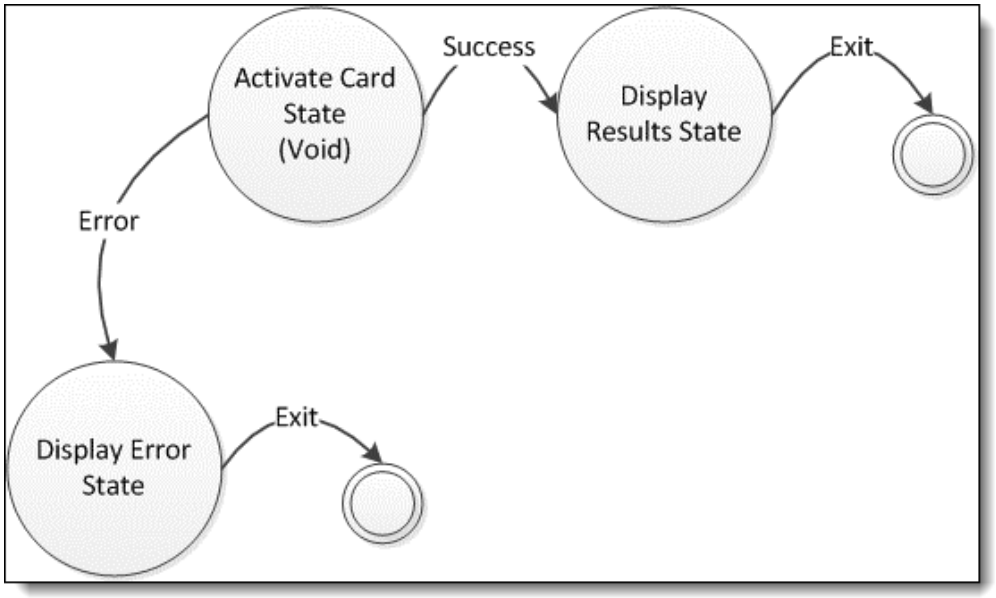
- **Account Number Entry State**
  This state displays the account number entry screen, allowing the user to enter an account number. If the user presses Cancel, the state machine exits immediately. Properties populated in the Data object:
  - AccountNumber
  - AccountNumberEntry
  - Track1-Track4 (if swiped)
  - AccountNumberMasked
  - PrintableAccountNumberMasked
- **Enter Amount State**
  This state displays a numeric entry pad to accept the amount to activate the card for. If the user presses Cancel, the state machine exits immediately. Properties populated in the Data object:
  - Amount
- **Activate Card State**
  This state calls the driver ActivateCardInternal() operation. The available data includes the general stored value data, as well as whatever the Account Number Entry State and Enter Amount State collected.
  - On a success, return Success and populate ProgramName and AccountBalance properties with the balance of the newly activated card. Populate the ItemType and ItemNumber fields with the item to place on the check representing the activation (usually a service charge). The state machine uses this list to build the success message that is displayed to the user.
  - On a failure, return Failure and populate the LastError with an exception representing the error case. Use the exception message to add text to the error.

- **Modify Check Detail State**
  The ItemType and ItemNumber properties are used to add an item to the check. The StoredValueModuleData object will be serialized onto the check.
- **Auto-Void State**
  This state is used for an auto-void, calling the VoidActivateCardInternal() operation on the driver to do so. The StoredValueModuleData object that was serialized onto the check is given to the driver in order to perform the void.
  - On a success, return Success.
  - On a failure, return Failure and populate the LastError with an exception representing the error case. Use the exception message to add text to the error.
- **Display Results State**
  The state machine builds the body of the message, displaying or printing the balances that were returned. The driver may customize this message with a header and footer as normal.
- **Display Error State**
  If the operation failed, the LastError is displayed to the user, and nothing is placed on the check.

Void Activate Card

The void activate card operation voids a stored value card activation. The Void Activate Card state machine is described below.

Figure 3-12 - Void Activate Card



- **Activate Card State (Void)**
  This state calls the driver to perform the VoidActivateCardInternal() operation. The available data includes the serialized StoredValueModuleData object on the issue card detail item that is voided.

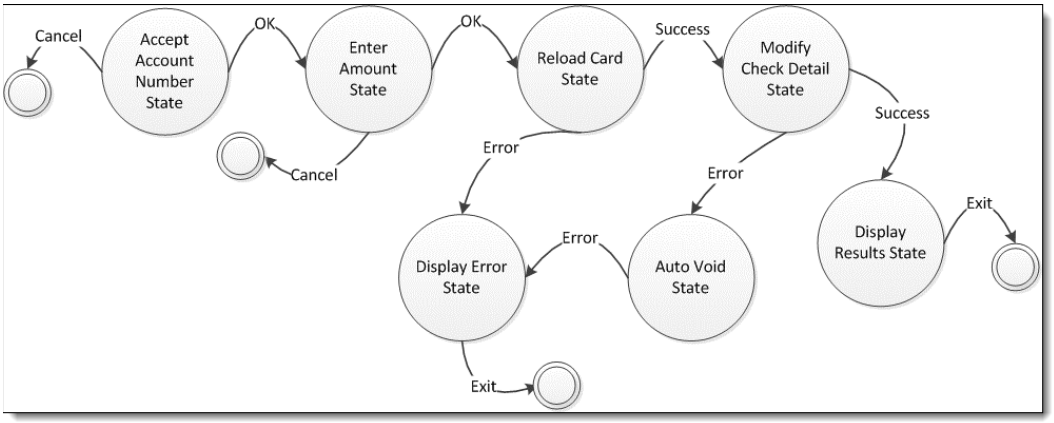
- On a success, return Success.
- On a failure, return Failure and populate the LastError with an exception representing the error case. Use the exception message to add text to the error.

- **Display Results State**
  The state machine builds the body of the message that a void was processed. The driver may customize this message with a header and footer as normal.
- **Display Error State**
  If the operation failed, the LastError is displayed to the user, and nothing is placed on the check.

Reload Card

The reload card operation reloads a stored value card for a certain amount. The Reload Card state machine is described below.

Figure 3-13 - Reload Card



- **Account Number Entry State**
  This state displays the account number entry screen, allowing the user to enter an account number. If the user presses Cancel, the state machine exits immediately. Properties populated in the Data object:
  - AccountNumber
  - AccountNumberEntry
  - Track1-Track4 (if swiped)
  - AccountNumberMasked
  - PrintableAccountNumberMasked

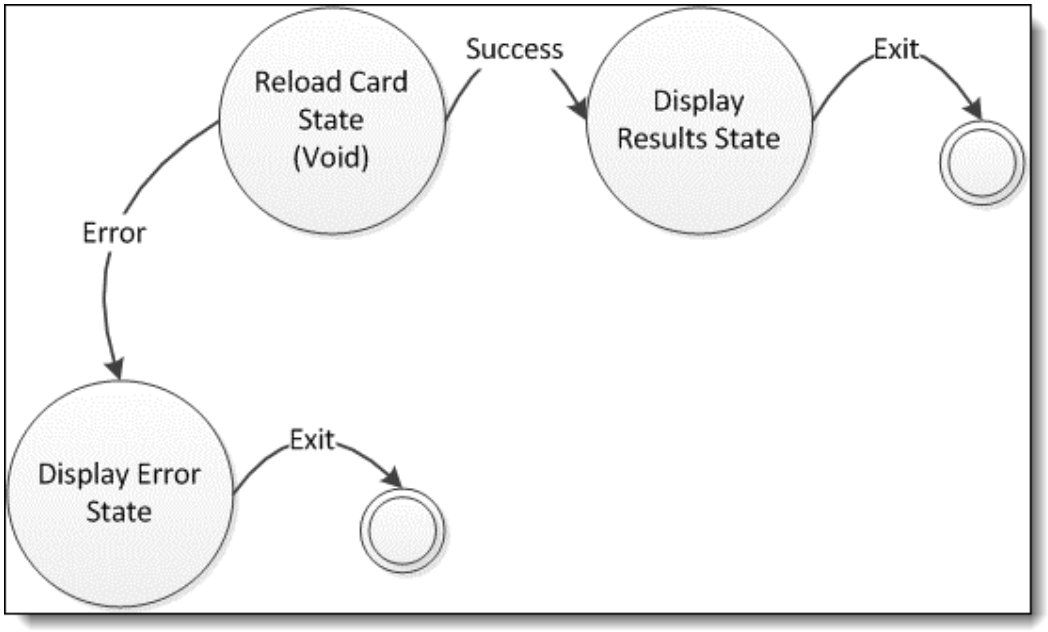
- **Enter Amount State**
  This state displays a numeric entry pad to accept the amount to reload the card for. If the user presses Cancel, the state machine exits immediately. Properties populated in the Data object:
  - Amount
- **Reload Card State**
  This state calls the driver ReloadCardInternal() operation. The available data includes the general stored value data, as well as whatever the Account Number Entry State and Enter Amount State collected.
  - On a success, return Success and populate the ProgramName and AccountBalance properties with the balance of the newly reloaded card. Populate the ItemType and ItemNumber fields with the item to place on the check representing the activation (usually a service charge). The state machine uses this list to build the success message that is displayed to the user.
  - On a failure, return Failure and populate the LastError with an exception representing the error case. Use the exception message to add text to the error.
  - If there is a communication failure and you want the system to shift into offline mode, return CommunicationFailure.

  See the Offline Mode section for more information.
- **Modify Check Detail State**
  The ItemType and ItemNumber properties are used to add an item to the check. The StoredValueModuleData object will be serialized onto the check.
- **Auto-Void State**
  This state is used for an auto-void, calling the VoidReloadCardInternal() operation on the driver to do so. The StoredValueModuleData object that was serialized onto the check is given to the driver in order to perform the void.
  - On a success, return Success.
  - On a failure, return Failure and populate the LastError with an exception representing the error case. Use the exception message to add text to the error.
  - If there is a communication failure and you want the system to shift into offline mode, return CommunicationFailure. See the section titled “Offline Mode” on page 11 for more information.
- **Display Results State**
  The state machine builds the body of the message, displaying or printing the balances that were returned. The driver may customize this message with a header and footer as normal.
- **Display Error State**
  If the operation failed, the LastError is displayed to the user, and nothing is placed on the check.

Void Reload Card

The void reload card operation voids a stored value card reload. The Void Activate Card state machine is described below.

Figure 3-14 - Void Reload Card



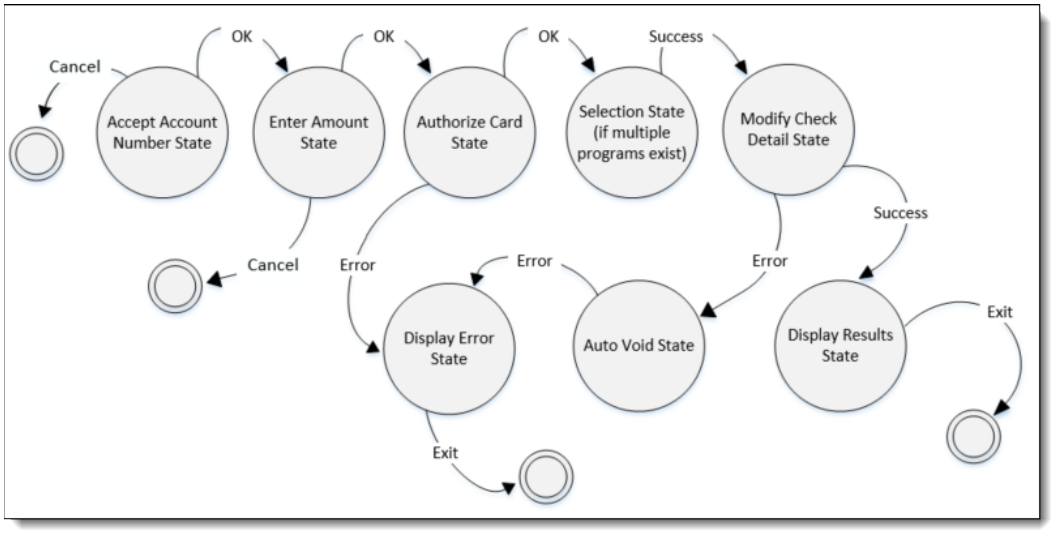
- **Reload Card State (Void)**
  This state calls the driver to perform the VoidReloadCardInternal() operation. The available data includes the serialized StoredValueModuleData object on the issue card detail item that is voided.
  - On a success, return Success.
  - On a failure, return Failure and populate the LastError with an exception representing the error case. Use the exception message to add text to the error.
  - If there is a communication failure and you want the system to shift into offline mode, return CommunicationFailure.

  See the Offline Mode section for more information.
- **Display Results State**
  The state machine builds the body of the message that a void was processed. The driver may customize the message with a header and footer as normal.
- **Display Error State**
  If the operation failed, the LastError is displayed to the user, and nothing is placed on the check.

Authorize Card

The authorize card operation authorizes a stored value card for a certain amount to pay on the check. Authorization allows a stored value redemption to occur with a tip, just like a credit card transaction. The Authorize Card state machine is described below.

Figure 3-15 - Authorize Card



- **Account Number Entry State**
  This state displays the account number entry screen, allowing the user to enter an account number. If the user presses Cancel, the state machine exits immediately. Properties populated in the Data object:
  - AccountNumber
  - AccountNumberEntry
  - Track1-Track4 (if swiped)
  - AccountNumberMasked
  - PrintableAccountNumberMasked
- **Enter Amount State**
  This state displays a numeric entry pad to accept the amount to authorize the card for. If the user presses Cancel, the state machine exits immediately. Properties populated in the Data object:
  - Amount
- **Authorize Card State**
  This state calls the driver AuthorizeCardInternal() operation. The available data includes the general stored value data, as well as whatever the Account Number Entry State and Enter Amount State collected.
  - On a success:

    If no program was submitted as part of the request, and multiple programs are configured for the account, return Success and populate a list of programs and current balances for user selection in Selection State.

If a program was submitted as part of the request or the account only has a single program configured, return Success and populate ProgramName and AccountBalance properties with the balance on the card, and populate the Amount in the Data object with the amount actually authorized, and the AuthorizationCode with the authorization code. If the amount that was authorized is less than the amount requested, set the IsPartialRedemption property to True. The state machine will use this list to build the success message that is displayed to the user.

- On a failure:

  Return Failure and populate the LastError with an exception representing the error case. Use the exception message to add text to the error.

  If there is a communication failure and you want the system to shift into offline mode, return CommunicationFailure.

See the Offline Mode section for more information.

- **Selection State**

  This state shows the list of programs and balances returned if multiple programs are configured on an account. Once selected, the program code is added to the request and Authorize Card State is actioned again. Properties populated in the Data object:

  - ProgramCode

- **Modify Check Detail State**

  The ItemType and ItemNumber properties is used to add an item to the check. The StoredValueModuleData object is serialized onto the check.

- **Auto-Void State**
  This state is used for an auto-void, calling the VoidAuthorizeCardInternal() operation on the driver to do so. The StoredValueModuleData object that was serialized onto the check is given to the driver in order to perform the void.

  - On a success, return Success.
  - On a failure, return Failure and populate the LastError with an exception representing the error case. Use the exception message to add text to the error.
  - If there is a communication failure and you want the system to shift into offline mode, return CommunicationFailure.

  See the Offline Mode section for more information.

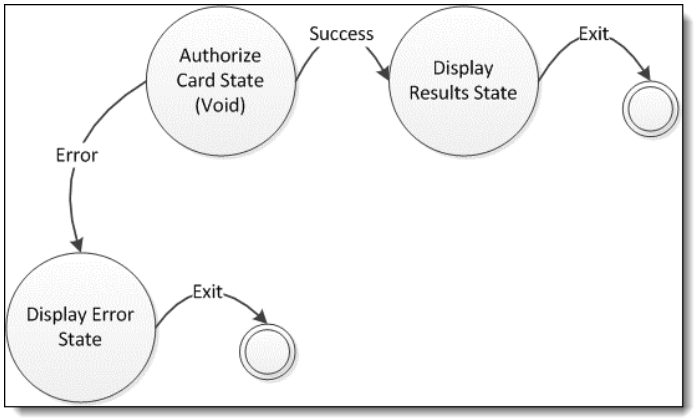
- **Display Results State**
  The state machine will build the body of the message, displaying or printing the balances that were returned and the amount redeemed. The driver may customize this message with a header and footer as normal.

- **Display Error State**
  If the operation failed, the LastError is displayed to the user, and nothing is placed on the check.

Void Authorize Card

The void authorize card operation voids a stored value card authorization. The Void Authorize Card state machine is described below.

Figure 3-16 - Void Authorize Card

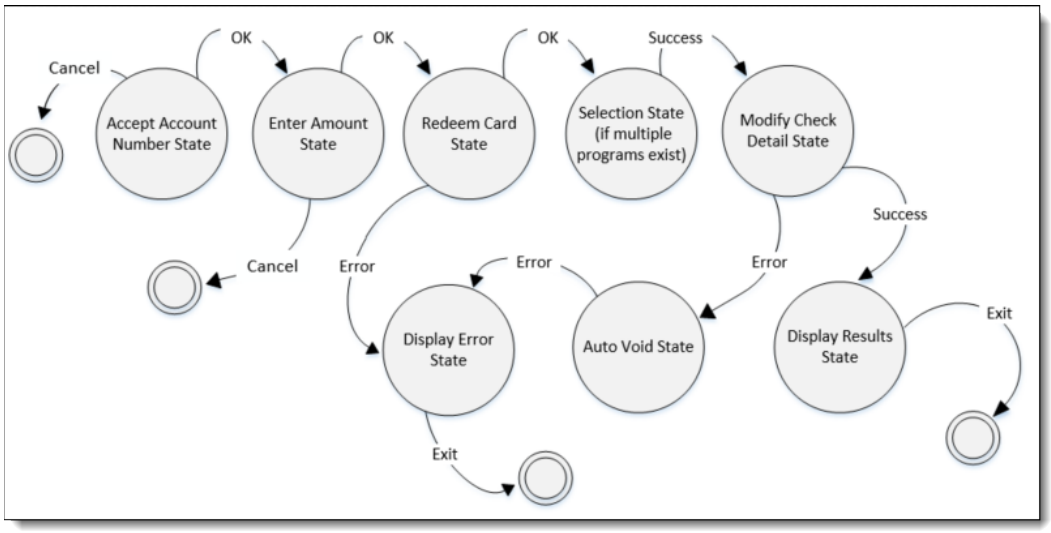


- **Authorize Card State (Void)**
  This state calls the driver to perform the VoidAuthorizeCardInternal() operation. The available data includes the serialized StoredValueModuleData object on the issue card detail item that is voided.
  - On a success, return Success.
  - On a failure, return Failure and populate the LastError with an exception representing the error case. Use the exception message to add text to the error.
  - If there is a communication failure and you want the system to shift into offline mode, return CommunicationFailure. See the section titled Offline Mode for more information.
- Display Results State
  The state machine will build the body of the message that a void was processed. The driver may customize this message with a header and footer as normal.
- Display Error State
  If the operation failed, the LastError is displayed to the user, and nothing is placed on the check.

Redeem Card

The redeem card operation redeems a stored value card for a certain amount to pay on the check. Redeem can be used either after an authorization or stand-alone. The Redeem Card state machine is described below.

Figure 3-17 - Redeem Card



- **Account Number Entry State**
  This state displays the account number entry screen, allowing the user to enter an account number. If the user presses Cancel, the state machine exits immediately. Properties populated in the Data object:
  - AccountNumber
  - AccountNumberEntry
  - Track1-Track4 (if swiped)
  - AccountNumberMasked
  - PrintableAccountNumberMasked
- **Enter Amount State**
  This state displays a numeric entry pad to accept the amount to redeem the card for. If the user presses Cancel, the state machine exits immediately. Properties populated in the Data object:
  - Amount
- **Redeem Card State**
  This state calls the driver RedeemCardInternal() operation. The available data includes the general stored value data, as well as whatever the Account Number Entry State and Enter Amount State collected.
  - On a success:

    If no program was submitted as part of the request and multiple programs are configured for the account, return Success and populate a list of programs and current balances for user selection in Selection State.

If a program was submitted as part of the request or the account only has a single program configured, return Success and populate ProgramName and AccountBalance properties with the balance on the card, and populate the Amount in the Data object with the amount actually redeemed, and the AuthorizationCode with the authorization code. If the amount that was redeemed is less than the amount requested, set the IsPartialRedemption property to **True**. The state machine uses this list to build the success message that is displayed to the user.

- On a failure, return Failure and populate the LastError with an exception representing the error case. Use the exception message to add text to the error.
- If there is a communication failure and you want the system to shift into offline mode, return CommunicationFailure.

  See the Offline Mode section for more information.

- **Selection State**

  This state shows the list of programs and balances returned if multiple programs are configured on an account. Once selected, the program code is added to the request and Redeem Card State is actioned again. Properties populated in the Data object:

  - ProgramCode

- **Modify Check Detail State**
  The ItemType and ItemNumber properties are used to add an item to the check. The StoredValueModuleData object is serialized onto the check.

- **Auto-Void State**
  This state is used for an auto-void, calling the VoidAuthorizeCardInternal() operation on the driver to do so. The StoredValueModuleData object that was serialized onto the check is given to the driver in order to perform the void.

  - On a success, return Success.
  - On a failure, return Failure and populate the LastError with an exception representing the error case. Use the exception message to add text to the error.
  - If there is a communication failure and you want the system to shift into offline mode, return CommunicationFailure.

  See the Offline Mode section for more information.

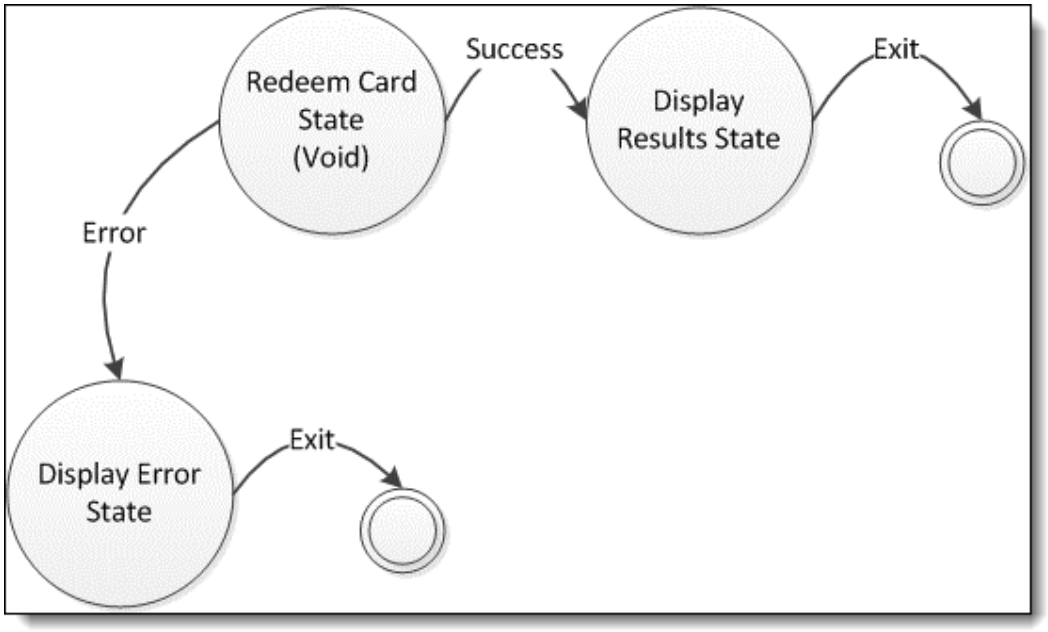
- **Display Results State**
  The state machine will build the body of the message, displaying or printing the balances that were returned and the amount redeemed. The driver may customize this message with a header and footer as normal.

- **Display Error State**
  If the operation failed, the LastError is displayed to the user, and nothing is placed on the check.

Void Redeem Card

The void redeem card operation voids a stored value card redemption. The Void Redeem Card state machine is described below.

Figure 3-18 - Void Redeem Card

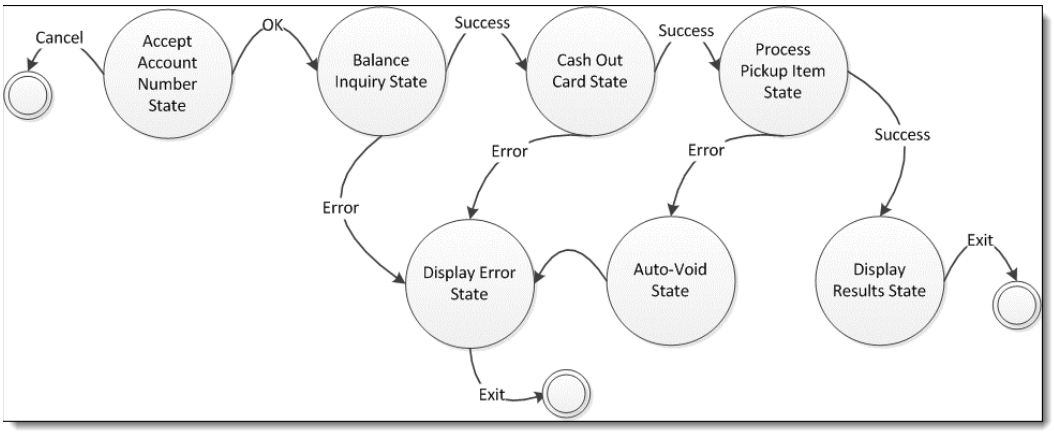


- **Redeem Card State (Void)**
  This state calls the driver to perform the VoidRedeemCardInternal() operation. The available data includes the serialized StoredValueModuleData object on the issue card detail item that is voided.
  - On a success, return Success.
  - On a failure, return Failure and populate the LastError with an exception representing the error case. Use the exception message to add text to the error.
  - If there is a communication failure and you want the system to shift into offline mode, return CommunicationFailure. See the section titled Offline Mode for more information.
- **Display Results State**
  The state machine will build the body of the message that a void was processed. The driver may customize this message with a header and footer as normal.
- **Display Error State**
  If the operation failed, the LastError is displayed to the user, and nothing is placed on the check.

Cash Out Card

The cash out card operation cashes out a stored value card. The Cash Out Card state machine is described below. It is possible to alter the flow of this operation by setting a special instruction for the state machine. If the StateMachineSpecialInstructions is set to SkipCashoutBalanceInquiry, then the state machine will transition directly to the Cash Out Card State. This value can be set in the GetNewDataObject() method of the Third-Party Stored Value Module. It must be set on every Data object that desires this behavior for the Cash Out operation.

Figure 3-19 - Cash Out Card



- **Account Number Entry State**
  This state displays the account number entry screen, allowing the user to enter an account number. If the user presses Cancel, the state machine exits immediately. Properties populated in the Data object:
  - AccountNumber
  - AccountNumberEntry
  - Track1-Track4 (if swiped)
  - AccountNumberMasked
  - PrintableAccountNumberMasked
- **Balance Inquiry State**
  This state retrieves the balance for the account using the BalanceInquiryInternal() method so that we know the exact amount to cash out for. If the balance is zero, the state machine exits.
  - On a success, return Success and populate the AccountBalance and ProgramName properties.
  - On a failure, return Failure and populate the LastError with an exception representing the error case. Use the exception message to add text to the error.
- **Cash Out Card State**
  This state calls the driver CashOutCardInternal() operation. The available data includes the general stored value data, as well as whatever the Account Number Entry State and Enter Amount State collected.

  - On a success, return Success and populate the Amount property with the amount to cash out and the ItemNumber with the object number of the Pickup Tender that will be used for the cashout. The tender in this case must be a pickup tender or the cash out will fail. The state machine will use this list to build the success message that is displayed to the user.
  - On a failure, return Failure and populate the LastError with an exception representing the error case. Use the exception message to add text to the error.

- **Process Pickup Item State**
  The ItemNumber and Amount properties will be used to add a pickup item to the check. The StoredValueModuleData object will be serialized onto the check.
- **Display Results State**
  The state machine will build the body of the message displaying or printing the amount cashed out. The driver may customize this message with a header and footer as normal.
- **Display Error State**
  If the operation failed, the LastError is displayed to the user, and nothing is placed on the check.

Appendix A
Command Module Data

The following table lists the data properties found in the Command Module layer, including the CommandModuleData class and the MagneticStripeCardModuleData class, which both the Loyalty and StoredValue modules use.

Table A-1 - Command Module Data

Property	Type	Content	Description
CheckExpirationDate	Bool	Infrastructure	Should the account number entry screen check the expiration date? (always False)
ConfigSettingsID	Int	Infrastructure	The ID of the Loyalty or Stored Value settings row
Created	DateTime	Infrastructure	The time the module was created
CurrentPropertyName	String	Infrastructure	The current restaurant property name
DisplayGuestNameLookup	Bool	Infrastructure	Should the guest name lookup button be displayed?
DisplayPhoneNumberLookup	Bool	Infrastructure	Should the phone number lookup button be displayed?
DriverOperationDone	AutoResetEvent	Infrastructure	Set by the driver monitor when the driver thread terminates
EncryptData	Bool	Infrastructure	Encrypt data flag (unused)
GuestNameLookup	OperationCommandDelegate	Infrastructure	Delegate that handles the guest name lookup screen
LanguageId	Int	Infrastructure	The current language in use
MaskAccountNumber	String	Infrastructure	Unused
MaskCharacter	String	Infrastructure	Unused
ModuleDriverPreamble	String	Infrastructure	The preamble of the current module
ModuleName	CommonModuleName	Infrastructure	The compound name of the module
PhoneNumberLookup	OperationCommandDelegate	Infrastructure	Delegate that handles the phone number lookup screen

Property	Type	Content	Description
RequireIssueNumber	Bool	Infrastructure	Should the account number entry screen require the issue number? (always False)
RequireStartDate	Bool	Infrastructure	Should a start date be required? (always False)
ResultCode	Int	Infrastructure	The internal result code of the last operation
ResultObject	Object	Infrastructure	Last UI result object
ResultPropertyName	String	Infrastructure	Name of the property to store the UI result object
Version	Decimal	Infrastructure	The module version (unused)
AccountNumber	SafeByteArray	Request	The account number of the magnetic stripe card
AccountNumberRetention	SafeByteArray	Request	Reserved for the iCare driver
AccountNumberContentValue	Enum	Request	Indicates that the account number field holds either: AccountIsAccountNumber, AccountIsPhoneNumber, or AccountIsGuestName
AccountNumberEntry	AccountNumberEntryMethod	Request	How was the account number collected?
AccountNumberEntryRetention	AccountNumberEntryMethod	Request	Reserved for the iCare driver
AccountNumberMasked	String	Request	The masked account number
InitiatingData	String	Request	The data from the button that triggered this operation
IsManualAuthorization	Bool	Request	The authorization code has been entered manually (unused)
PrintableAccountNumberMasked	String	Request	The printable masked account number
Track1	SafeByteArray	Request	Track 1 data
Track1Retention	SafeByteArray	Request	Reserved for the iCare driver
Track2	SafeByteArray	Request	Track 2 data
Track2Retention	SafeByteArray	Request	Reserved for the iCare driver
Track3	SafeByteArray	Request	Track 3 data
Track3Retention	SafeByteArray	Request	Reserved for the iCare driver

Property	Type	Content	Description
Track4	SafeByteArray	Request	Track 4 data
Track4Retention	SafeByteArray	Request	Reserved for the iCare driver
DriverResultCode	CommandDriverResultCode	Response	Result of the current driver operation
LastError	Exception	Response	The error exception to be displayed to the user

More Details on the Properties

1. ModuleName. Infrastructure, Read Only. This is a compound name made up of four components:
    a. ModuleClass. The class of command module. Currently only Loyalty and StoredValue are valid options.
    b. ModuleCategory. The category of module. Currently only LoyaltyOperation and StoredValueOperation are valid options.
    c. ModuleID. The ID of the module used throughout the system to identify the module.
    d. Description. The description of the module.
2. InitiatingData. Infrastructure, Read Only. The data from the button that was just pressed. This data is a string in the form "ModuleID:FunctionName". The ModuleID is the id of the module that will handle the button press. The function name is converted into the LoyaltyFunction or StoredValueFunction enumerations that form the operations available in the Loyalty or StoredValue driver interface.
3. DriverResultCode. Response, Writable, Special. An enumeration of possible return values of the driver operation. This is normally set by returning the value from the specific driver operation upon completion and is not to be set directly in the Data object.
    a. Success. The operation was successful.
    b. Failure. The operation was a failure. Place the error to display in the LastError property of the Loyalty or Stored Value Module Data (described below).
    c. CommunicationFailure. Return this to indicate a communication failure and switch the system to offline mode.
    d. UserInputRequired. Return this to indicate in some logic flows that the user needs to make a selection based upon a response from the provider. The logic will describe exactly when this value is to be used.
4. AccountNumber. Request, Read Only. A SafeByteArray holding the account number. Use the CommandDriver.SafeByteArrayToString() method to unencrypt the string. Account numbers can also store guest name or phone number for lookups. They are encoded in the following way:
    a. Starts with "#" – phone number

b. Contains any alpha character – guest name

c. All digits – account number

5. AccountNumberEntry. Request, Read Only. An enumeration with the following values:

   a. None – the user pressed Cancel

   b. Manual – the user manually entered the account number

   c. Automatic – the user swiped the card

6. AccountNumberMasked. Request, Read Only. The masked account number.

7. PrintableAccountNumberMasked. Request, Read Only. The printable masked account number. Currently the same as the AccountNumberMasked.

8. Track1 – Track4. Request, Read Only. These SafeByteArray values hold the tracks of a swiped card. They will only be populated if AccountNumberEntry is Automatic.

Appendix B
Loyalty Module Data

Table B-2 - Loyalty Module Data

Property	Type	Content	Description
AlwaysPrintLoyaltyResults	Bool	Infrastructure	Always print the loyalty results?
CurrentFunction	LoyaltyFunction	Infrastructure	The current function being executed
CurrentState	LoyaltyState	Infrastructure	The current state of the state machine
CurrentTransition	LoyaltyTransitions	Infrastructure	The current transition of the state machine being executed
FormattedResults	String	Infrastructure	The display message built by the infrastructure from result data
IsFinal	Bool	Infrastructure	Reserved for the iCare driver
LoyaltyNumberEntryResult	Decimal	Infrastructure	The amount entered from the retrieve number prompt
NeverDisplayLoyaltyResults	Bool	Infrastructure	Never display the loyalty results?
OfflineSerialized DriverConfiguration	String	Infrastructure	Used by the offline subsystem
OfflineSerilaized ModuleConfiguration	String	Infrastructure	Used by the offline subsystem
PrinterMessageBody	List<String>	Infrastructure	The internal representation of the message body for printing
RemoveGuestNam e LookupButton	Bool	Infrastructure	Don't allow lookup by guest name?
RemoveLoyaltyResult s PrintButton	Bool	Infrastructure	Don't allow a print button on the results screen?
RemovePhoneNumb er LookupButton	Bool	Infrastructure	Don't allow lookup by phone number?

Property	Type	Content	Description
ReprocessRequired	Bool	Infrastructure	If the system goes offline, and the request can be performed offline, the infrastructure will mark it for reprocessing
ShowVoidOnGuestCheck	Bool	Infrastructure	Show voids on the guest check?
SupportOfflineTransactions	Bool	Infrastructure	Are offline transactions supported?
UIDisplayMessageBody	List<String>	Infrastructure	The internal representation of the message body for screen display
Amount	Decimal	Request	The amount of the current request
BusinessDate	DateTime	Request	The system business date
CheckDetailDiscount	List<LoyaltyCheckSummaryDetailDiscount>	Request	A list of discount items from the current check (if any)
CheckDetailMenuItem	List<LoyaltyCheckSummaryDetailMenuItem	Request	A list of menu items from the current check (if any)
CheckDetailPayment	List<LoyaltyCheckSummaryDetailPayment>	Request	A list of payment items from the current check (if any)
CheckDetailSalesItemizer	List<LoyaltyCheckSummaryDetailSalesItemi	Request	A list of sales itemizers from the current check (if any)
CheckDetailServiceCharge	List<LoyaltyCheckSummaryDetailServiceCharge>	Request	A list of service charge items from the current check (if any)
CheckNumber	Int	Request	If the operation was performed inside a check, this is the check number
CheckSummaryCheckOpenTim	DateTime	Request	The time the check was opened
CheckSummaryTotalsServic-eCharges	String	Request	The total service charge amount on the check, as a string
CheckSummaryTotalsTaxes	String	Request	The total taxes on the check, as a string
CheckSummaryTotalsAmountDue	String	Request	The total amount due on the check, as a string
CheckSummaryTotalsDiscounts	String	Request	The total discount amount on the check, as a string
CheckSummaryTotalsPayments	String	Request	The total payment amount on the check, as a string

Property	Type	Content	Description
CouponCode	String	Request	The coupon code of the current coupon that has either been entered by the user (Accept Coupon) or chosen by the user from a list (Coupon Inquiry)
CouponValueAmount	Decimal	Request	The amount of the coupon referenced in the CouponCode property
IsCheckActive	Bool	Request	Is there a check for this operation?
Language	String	Request	The ISO 3166 language code of the current language
LocalCurrency	String	Request	The ISO 4217 currency code of the current currency
LocalDateTime	DateTime	Request	The date/time of the current request
PointsToRedeem	Decimal	Request	The number of points to redeem
PreviousTotalDue	Decimal	Request	The total on the check before it was closed (this is mostly for the Check Reprocessor, where all checks are closed and thus all check total due values are 0).
RevenueCenter	Int	Request	The revenue center where the request was made
SelectedActionItem	LoyaltyActionItem	Request	The action item chosen by the user from a selection prompt
SelectedCoupons	List<int>	Request	The list of coupons selected by the user (only single select for now)
TerminalID	Int	Request	The terminal number of the terminal making the request
TerminalType	String	Request	The terminal type of the terminal making the request
TransactionEmployee	Long	Request	The employee ID of the employee who requested the request
ItemNumber	Int	Response	The object number of the item to put on the check

Property	Type	Content	Description
ItemType	String	Response	The type of the item to put on the check
LocalBalance	List<LoyaltyLocalBalance>	Response	Stores the results of a balance inquiry for multiple programs
LookupResults	List<LoyaltyLookup>	Response	The driver fills this in with guest account numbers for guest name or phone number lookup selection display
LoyaltyActionItems	List<LoyaltyActionItem>	Response	Used to return lists of items (like coupons) to the state machine for display/processing.
LoyaltyPostingItems	List<LoyaltyPostingItem>	Response	A list of items to be added to the check
LoyaltyUniqueItems	List<LoyaltyUniqueItem>	Response	Reserved for the iCare driver
PrinterMessageHeader	List<String>	Response	The header of the screen display message
PrinterMessageFooter	List<String>	Response	The footer of the screen display message
PrinterMessageOverride	List<String>	Response	Override the entire screen display message (header, footer, and body are ignored if this is not null)
PromptForRedeemSVC	Bool	Response	Reserved for the iCare driver
RedeemPointIncrement	Decimal	Response	Reserved for the iCare driver
UIDisplayMessageHeader	List<String>	Response	The header of the screen display message
UIDisplayMessageFooter	List<String>	Response	The footer of the screen display message
UIDisplayMessageOverride	List<String>	Response	Override the entire screen display message (header, footer, and body are ignored if this is not null)

More Detail on the Request Properties

- CheckDetailDiscount (List<LoyaltyCheckSummaryDetailDiscount>) – Request, Read Only. Discount items on the check:
  - StatusBits (string) – the status bits from the check. Use the BitUtility to decode the bit values from the string.
  - DisplayName (string) – the name the item places on the check
  - Number (int) – the object number of the discount
  - Quantity (int) – the quantity

- Total (int) – the total of the discount

- CheckDetailMenuItem (List<LoyaltyCheckSummaryDetailMenuItem>) – Request, Read Only. Menu items on the check:
  - StatusBits (string) – the status bits from the check. Use the BitUtility to decode the bit values from the string.
  - DisplayName (string) – the name the item places on the check
  - Number (int) – the object number of the menu item
  - Quantity (int) – the quantity
  - Total (int) – the total of the menu item
  - MainLevel (int) – the main level of the menu item
  - SubLevel (int) – the sub level of the menu item
- CheckDetailPayment (List<LoyaltyCheckSummaryDetailPayment>) Request, – Read Only. Payments on the check:
  - StatusBits (string) – the status bits from the check. Use the BitUtility to decode the bit values from the string.
  - DisplayName (string) – the name the item places on the check
  - Number (int) – the object number of the discount
  - Quantity (int) – the quantity
  - Total (int) – the total of the discount
- CheckDetailSalesItemizer (List<LoyaltyCheckSummaryDetailSalesItemizer>) – Request, Read Only. Sales itemizers on the check:
  - StatusBits (string) – the status bits from the check. Use the BitUtility to decode the bit values from the string.
  - DisplayName (string) – the name the item places on the check
  - Number (int) – the object number of the discount
  - Total (int) – the total of the discount
- CheckDetailServiceCharge (List<LoyaltyCheckSummaryDetailServiceCharge>) – Request, Read Only. Service charges on the check:
  - StatusBits (string) – the status bits from the check. Use the BitUtility to decode the bit values from the string.
  - DisplayName (string) – the name the item places on the check
  - Number (int) – the object number of the discount
  - Quantity (int) – the quantity
  - Total (int) – the total of the discount

More Detail on the Response Properties

- ItemType – Response, Writable. The type of item to be placed on the check.
  - D – Discount
  - T – Payment Tender
  - M – Menu Item
  - S – Service Charge
- LocalBalance (List<LoyaltyLocalBalance>) – Response, Writable. Stores the point balance for a program.
  - ProgramType (ProgramType) – whether this is a loyalty or stored value balance
  - ProgramName (string) – the name of the program
  - ProgramCode (string) – unused
  - Balance (decimal) – the balance of the account for this program
  - LastTransactionAmount (decimal) – the amount of the last transaction against this program
- LookupResults (List<LoyaltyLookup>) – Response, Writable. The lookup items from a guest name/phone number lookup.
  - AccountNumber (SafeByteArray) – the account number
  - GuestName (string) – the guest name
  - PhoneNumber (string) – the phone number
  - PostalCode (string) – unused
- LoyaltyActionItems (List<LoyaltyActionItem>) – Response, Writable. Stores an action (coupons only in a third-party driver).
  - LoyaltyActionType (LoyaltyActionType) – unused
  - Text (string) – the text to display for the action
  - Code (string) – unused
  - DataItems(List<LoyaltyActionData>) – a list that is always of size 1 used to store the data associated with this action
    - Transaction (ActionTransaction) – AcceptCoupon is the only valid value for third-party implementations
      - PID (string) – unused
      - Prompt (string) – unused
    - Value (string) – for AcceptCoupon transactions this is the coupon code that will be used to post a coupon to the check
- LoyaltyPostingItems (List<LoyaltyPostingItem>) – Response, Writable. Additional items to be placed on the check by the operation (in addition to whatever the operation places on the check).
  - ItemType (string) – The type of item (see above)

- ItemNumber (int) – the object number of the item
- Amount (decimal) – the amount of the item

Appendix C
Stored Value Data

Table C-3 - Stored Value Data

Property	Type	Content	Description
AlwaysPrintStoredValueResults	Bool	Infrastructure	Always print the stored value results?
CardQuantity	Int	Infrastructure	The number of cards to issue/ activate in a multi-issue/activate operation
CurrentFunction	StoredValueFunction	Infrastructure	The current function being executed
CurrentState	StoredValueState	Infrastructure	The current state of the state machine
CurrentTransition	StoredValueTransitions	Infrastructure	The current transition of the state machine being executed
FormattedResults	String	Infrastructure	The display message built by the infrastructure from result data
IsLoyaltyInitiatedOperation	Bool	Infrastructure	Reserved for the iCare driver
NeverDisplayStoredValueResults	Bool	Infrastructure	Never display the stored value results?
OfflineSerialized DriverConfiguration	String	Infrastructure	Used by the offline subsystem
OfflineSerilaized ModuleConfiguration	String	Infrastructure	Used by the offline subsystem
RemoveGuestName LookupButton	Bool	Infrastructure	Don’t allow lookup by guest name?
RemovePhoneNumber LookupButton	Bool	Infrastructure	Don’t allow lookup by phone number?
RemoveStoredValueResults PrintButton	Bool	Infrastructure	Don’t allow a Print button on the results screen?
ReprocessRequired	Bool	Infrastructure	If the system goes offline, and the request can be performed offline, the infrastructure will mark it for reprocessing
ShowVoidOnGuestCheck	Bool	Infrastructure	Show voids on the guest check?
SupportOfflineTransactions	Bool	Infrastructure	Are offline transactions supported?

Property	Type	Content	Description
BusinessDate	DateTime	Request	The system business date
CheckNumber	Int	Request	If the operation was performed inside a check, this is the check number
CheckSummary	StoredValueCheckSummary	Request	The summary of the check
GuestCheckTotalAmountDue	Decimal	Request	The total amount due on the check
IsCheckActive	Bool	Request	Is there a check for this operation?
Language	String	Request	The ISO 3166 language code of the current language
LocalCurrency	String	Request	The ISO 4217 currency code of the current currency
LocalDateTime	DateTime	Request	The date/time of the current request
RevenueCenter	Int	Request	The revenue center where the request was made
TerminalID	Int	Request	The terminal number of the terminal making the request
TerminalType	String	Request	The terminal type of the terminal making the request
TransactionEmployee	Long	Request	The employee ID of the employee who requested the request
Amount	Decimal	Request/ Response	The amount of the current request to make and the amount returned from the driver
AccountBalance	Decimal	Response	The balance of the current stored value account
AccountLookupDetailsResults	IList<AccountLookupDetails>	Response	A list of account details for selecting from multiple accounts
AuthorizationCode	String	Response	The authorization code for a redemption authorization operation
IsPartialRedemption	Bool	Response	Is the current Stored Value redemption for the partial amount of the check?
ItemNumber	Int	Response	The object number of the item to put on the check
ItemType	String	Response	The type of the item to put on the check
PrinterMessageHeader	List<String>	Response	The header of the screen display message

Property	Type	Content	Description
PrinterMessageFooter	List<String>	Response	The footer of the screen display message
PrinterMessageOverride	List<String>	Response	Override the entire screen display message (header, footer, and body are ignored if this is not null)
ProgramName	String	Response	The name of the stored value program for the card
StateMachineSpecialInstructions	Enum	Response	Special instructions for the state machine
UIDisplayMessageHeader	List<String>	Response	The header of the screen display message
UIDisplayMessageFooter	List<String>	Response	The footer of the screen display message
UIDisplayMessageOverride	List<String>	Response	Override the entire screen display message (header, footer, and body are ignored if this is not null)

More Detail on Request Properties

- CheckSummary – Request, Read Only. The check summary, described below:
  - CheckOpenTime (DateTime) – The time stamp when the check was opened.
  - DiscountCheckSummaryDetails (StoredValueCheckSummaryDetails) – discounts on the check
    - StatusBits (string) – the status bits from the check. Use the BitUtility to decode the bit values from the string.
    - DisplayName (string) – the name the item places on the check
    - Number (int) – discount object number
      - Quantity (int) – quantity
    - Total (decimal) – total of the discount
  - ServiceChargeCheckSummaryDetails (StoredValueCheckSummaryDetails) – service charges on the check (see above)
  - PaymentCheckSummaryDetails (StoredValueCheckSummaryDetails) – payment items on the check (see above)
  - MenuItemCheckSummaryDetails (StoredValueCheckSummaryMenuItemDetails) – menu items on the check
    - StatusBits (string) – the status bits from the check. Use the BitUtility to decode the bit values from the string.
    - DisplayName (string) – the name the item places on the check
    - Number (int) – menu item object number

        - Quantity (int) – quantity
        - Total (decimal) – total of the menu item
        - MainLevel (int) – main level of the item
        - SubLevel (int) – sublevel of the item
- CheckSummaryTotalDiscounts (string) – the total of discounts on the check
- CheckSummaryTotalAmountDue (string) – the total amount due
- CheckSummaryTotalPayments (string) – the total of payments on the check
- CheckSummaryTotalServiceCharges (string) – the total of service charges on the check
- CheckSummaryTotalTaxes (string) – the total taxes on the check
- SalesItemizerCount (int) – the number of sales itemizers on the check

More Detail on Response Properties

- AccountLookupDetailsResults (List<AccountLookupDetails>) – When a guest name or phone number lookup returns multiple results, they are placed here by the driver.
    - AccountNumber (SafeByteArray) – the account number
    - GuestName (string) – the guest name
    - PhoneNumber (string) – the phone number
    - PostalCode (string) – unused
- ItemType – Response, Writable. The type of item to be placed on the check.
    - D – Discount
    - T – Payment Tender
    - M – Menu Item
    - S – Service Charge

Appendix D
Third Party Driver Distribution Procedures

Before developing a driver, complete the steps in this appendix for each third-party driver.

1. After you receive the ThirdPartyDriverPackage.zip package from the third-party vendor, unzip it to a desired location.

   The ThirdPartyDriverPackage.zip contains two identical directories:

   - **\CE\ThirdPartyDrivers2.0**: Place only third-party drivers for clients running Windows CE in this ThirdPartyDrivers2.0 folder.
   - **\WIN32\ThirdPartyDrivers2.0:** Place only third-party drivers for clients running Windows 32 in this ThirdPartyDrivers2.0 folder.

2. Copy all drivers that you want to deploy to Simphony clients to the appropriate ThirdPartyDrivers2.0 folders listed above.
3. Edit the Setup.dat file in each of the ThirdPartyDrivers2.0 folders to ensure that any new drivers will be transferred to the Simphony clients. To edit the Setup.dat file, right-click the file labeled **Setup.dat**, and then select **Open With**, **Notepad**.

> **NOTE:**
>
> The Setup.dat file determines whether a client receives a new package. New packages are those which include a new Setup.dat version number. An example of a Setup.dat file is depicted below.

```
#####################################################################

                    CAL Script to update CAL on WIN32

#####################################################################
VERSION,1.0.0,

NAME,ThirdPartyDrivers

TRANSFERFILE,SkeletonLoyaltyCommandModule.dll,%AppRoot%WebServer\
wwwroot\EGateway\Handlers\SkeletonLoyaltyCommandModule.dll
TRANSFERFILE,SkeletonStoredValueCommandModule.dll,%AppRoot%WebSer
ver\wwwroot\EGateway\Handlers\SkeletonStoredValueCommandModule.dl l
PAUSE, 4000 REBOOT
```

The Setup.dat file contains three important fields:

- **VERSION**: This is the version number of the package that will be distributed to Simphony clients. When Simphony detects a new version number, the driver package will be redistributed.

- **NAME**: The name of the folder in which the driver files reside. In this example, the driver files will reside in the ThirdPartyDrivers folder.
- **TRANSFERFILE**: The first part of the TRANSFERFILE command before the comma tells Simphony where to look for the driver file. In the example above, Simphony will look for the SkeletonLoyaltyCommandModule.dll in the same directory in which the Setup.dat file resides. The portion after the comma tells Simphony where on the client to transfer the file to. Again looking at the example above, the SkeletonLoyaltyCommandModule.dll driver file will be transferred to the %AppRoot%WebServer\wwwroot\EGateway\Handlers\ folder on the client.

> **NOTE:**
>
> Do not change the %AppRoot%WebServer\wwwroot\EGateway\Handlers\ portion of the TRANSFERFILE line as all drivers will reside in the \Handlers folder on the client system.

4. With the Setup.dat file open in Notepad, edit the existing TRANSFERFILE lines to point to your new driver files. Change only the sections in bold text in the TRANSFERFILE command.

   ```
   TRANSFERFILE,SkeletonLoyaltyCommandModule.dll,%Ap
   pRoot%WebServer\wwwroot\EGateway\Handlers\SkeletonLoya
   ltyCommandModule.dll
   ```

   For example, if your driver file is named MyDriver.dll, the TRANSFER file line should look like this:

   TRANSFERFILE,**MyDriver.dll**,%AppRoot%WebServer\wwwr
   oot\EGateway\Handlers**MyDriver.dll**

   If you have more than one file that needs to be transferred to the Simphony client(s), additional TRANSFERFILE lines can be added to the Setup.dat file.

> **NOTE:**
>
> Before saving the final changes to the Setup.dat file, ensure that the following lines no longer exist as these Skeleton files are only used as part of the instructional template:
>
> TRANSFERFILE,SkeletonLoyaltyCommandModule.dll,%
> AppRoot%WebServer\wwwroot\EGateway\Handlers\Skel
> etonLoyaltyCommandModule.dll.
>
> TRANSFERFILE,SkeletonStoredValueCommandModule.d
> ll,%AppRoot%WebServer\wwwroot\EGateway\Handlers\ SkeletonStoredValue
> CommandModule.dll.

5. Save and then close the edited Setup.dat file.
6. Before sending the driver package back to the Oracle Hosting Center, remove the following files from each of the ThirdPartyDrivers2.0 folders. These files are included only as part of the template for instructional purposes and are not necessary for your third-party driver distribution.
   - SkeletonLoyaltyCommandModule.dll

- SkeletonStoredValueCommandModule.dll

After making all of the changes to the Setup.dat file, refer to the MICROS Simphony Client Deployment Guide for instructions to create and upload custom CAL packages.