Oracle® Hospitality Simphony

Transaction Services API Document
Release 2.8
E93104-02

July 2018

ORACLE

Contents

Tables

Figures

Preface

This document is intended to be used by software engineers developing applications that will interface to Simphony using Transaction Services (TS) API.

Audience

This document is intended for the following audiences:

- Installers/Programmers
- Dealers
- Customer Service
- Training Personnel
- MIS or IT Personnel

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
- Screen shots of each step you take

Prerequisite Knowledge

This document assumes the reader has the following knowledge or expertise:

- Operational understanding of PCs
- Understanding of basic network concepts
- Experience in configuring workstation clients in the Simphony EMC

Documentation

Oracle Hospitality product documentation is available on the Oracle Help Center at http://docs.oracle.com/en/industries/hospitality/

Glossary

The following acronyms and abbreviations are used within this document:

Acronym / Abbreviation	Full Text
TS	Transaction Services
API	Application Programming Interface
POS	Point of Sale
OPS	Operations Software (aka, POS client)
CAL	Client Application Loader
CAPS	Check and Posting Service
EMC	Enterprise Management Console
IIS	Microsoft Internet Information Services
SQL	Structured Query Language
SVC	Stored Value Card
PDA	Personal Digital Assistance
PC	Personal Computer
RVC	Revenue Center
DB	Database

Revision History

Date	Description of Change
September 2012	• Initial publication
December 2012	• Minor edits
August 2015	• Minor edits
January 2018	• Formatting changes only
July 2018	• Removed the License Requirement section that applied to version 2.7 MR3. This section is not valid for version 2.8

1 Introduction and Overview

Transaction Services (TS) API is a web service that helps integrating a third-party application to Simphony Point-Of-Sales (POS) system. It leverages core business functionalities of Simphony POS system and exposes them via web methods for third-party integration. This web service provides methods to perform following POS operations.

1. Calculate total amounts of a transaction
2. Create a guest check for a transaction in Simphony POS database
3. Add one or more items like menu, discount, service charge, tender etc. to any existing open guest check
4. Void all items of an open guest check
5. Retrieve summary of all open guest checks from a specific or all revenue centers of a property
6. Retrieve printed version of a guest check (i.e. print receipt for a guest check)

All guest checks posted via Transaction Services API can be opened with OPS user interface that are configured to run on any workstation from the same property.

Given below are a few sample business scenarios in which Transaction Services API will be of useful to integrate a third-party application with Simphony POS system.

- Remote ordering from a kiosk or mobile phone application
- Remote ordering or centralized order dispatch via a web application
- Guest payment approval using mobile phones or PDAs

Transaction Services web service is installed on each workstation where Simphony CAL package is executed. This CAL package takes care of downloading required file contents from Simphony application server to the actual workstation and installs it. As part of installation steps, CAL package creates a shortcut to ServiceHost.exe at desktop. Launching ServiceHost.exe will make sure that TS web service is hosted (along with other services and OPS user interface) as well, and ready to process request from any client.

2 Simphony Architecture

As mentioned in the previous section, Transaction Services API is a web service and is hosted by ServiceHost.exe that's designed to run on each POS workstation of Simphony system. The Service Host application is a custom built web server by Oracle Hospitality that host various services like transaction services, print controller, cash management, check and posting along with OPS user interface that are required on a workstation for end-to-end POS operations. As mentioned above, Service Host runs OPS user interface as well, but user can configure (using EMC tool) to remove OPS user interface from Service Host in case only services are required to be hosted for third-party integration.

OPS (i.e. Operational Software) provides user interface for all POS operation in Simphony system. The order taker can logon to OPS with valid credentials and ring in a transaction to create a guest check. As part of a transaction, he/she can add menu items, discounts, service charge & tender details to the guest check and save it to POS database. The guest checks that are created on one workstation of the property can be accessed from other workstations of the same property with the help of CAP (Check and Posting) service.

TS web service uses the same underlying business libraries that are being used by OPS for all POS operation. So, technically TS web service is a headless (i.e. UI less) version of OPS.

The guest check created via Transaction Services API will be posted to enterprise database (MCRSPOS) for reporting purpose via CAPs. All transactions made via TS web service or OPS UI will be stored in local POS database named DataStore first. The CAPS service that owns CheckAndPostingDB database will post all transactional data to enterprise database (MCRSPOS) with the help of EGateway service that's hosted on Simphony application server.

Any configuration changes made by EMC application at enterprise server would get downloaded to local POS database of each workstation with the help of DB Sync component. By default, the DB Sync operation runs for every 30 minutes. Thus, any configuration changes made at enterprise server for a specific property/workstation will be available for Transaction Services and OPS within 30 minutes. The DB Sync can be initiated anytime manually with the help of a function button in OPS too.

This diagram shows the architecture of Simphony system that includes Transaction Services API/web service as well.

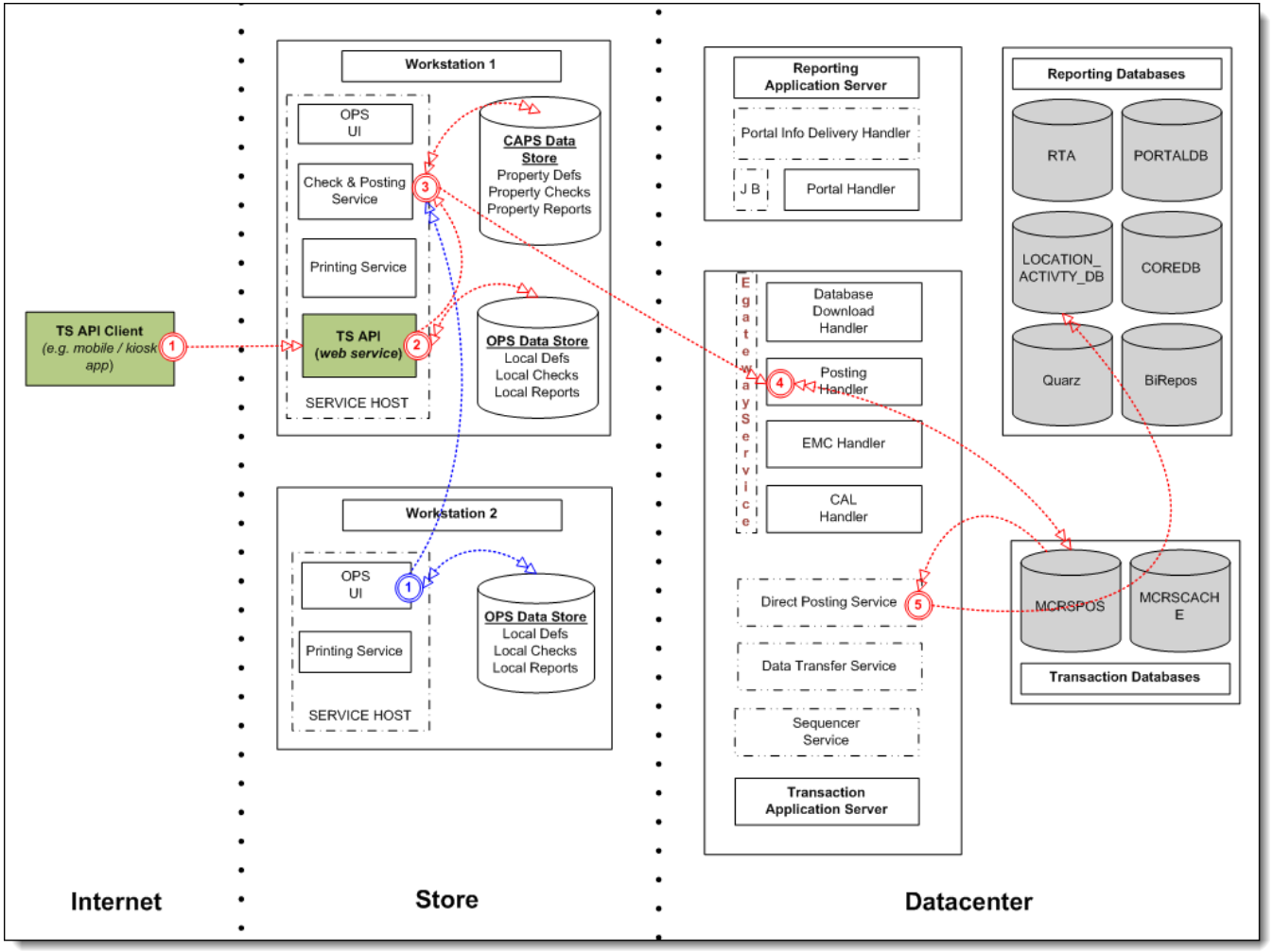

Figure 2-1 Transaction Services Data Flow

Direct Posting Services is a windows service that runs on the application server to upload data from transaction database (i.e. MCRSPOS) to one of the reporting database named LOCATION_ACTIVITY_DB. Any workstation in a store can be configured to host TS API. In this example, **Workstation 1** hosts the TS API

3 Hosting Method

Transaction Services web service is pre-installed on each workstation or application server where the Simphony installation media or CAL package was executed. Configuring a proper workstation client of type POS API from EMC application will allow for the successful hosting of Transaction Services web service at POS workstation.

The Service Host that's configured via EMC to run on a workstation will host the Transaction Services API as a web service by default at port number 8080.

The sample URL of web service is:

http://*WorkstationIPAddress*:8080/EGateway/SimphonyPosApiWeb.asmx

So, any client machine that has access to given POS workstation can consume Transaction Services straightaway by referring to the correct URL of TS web service.

Below are the steps to configure a POS API Client for TS web service from EMC.

1. Log onto the EMC.
2. Select a property that needs to be configured.
3. Click the **Setup** tab.
4. Click the **Workstation** link under Hardware/Interfaces.
5. Add a new workstation for a POS API client, named for example, TS API.
6. Open the new workstation record in Form view.
7. Select **2 - POSAPI Client** from the **Type** dropdown under the General Settings section of **General** tab.
8. If the OPS user interface is not needed to be hosted by Service Host, then click the **Remove OPS From Service Host** link. This ensures that only services are hosted by a Service Host when it's launched from a POS workstation.
9. If the OPS user interface is needed, then select the Service Host ID of corresponding OPS from **Service Host ID** dropdown field of Service Host Fields section.
10. **Save** all changes.

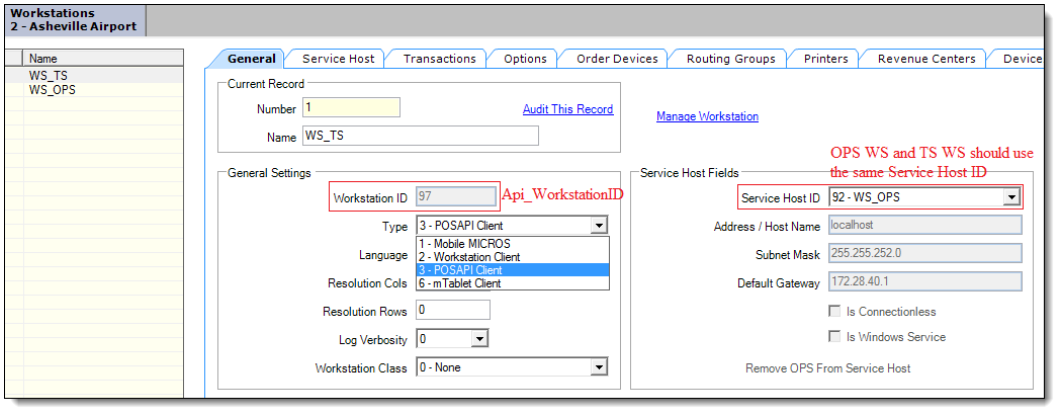

Figure 3-1 Workstation Module - POS API Client

To verify whether or not TS web service is hosted correctly, first launch ServiceHost.exe on the POS workstation and then navigate to the web service URL given above using any web browser. If WSDL details are displayed in the web browser then it means web service is hosted properly.

Apart from above method (that is, hosting TS web service on Service Host), TS web service can be hosted on IIS as well. By default, the Simphony application server will have TS web service installed and hosted after successful execution of Simphony installation media. That web service can be accessed with right URL to application server. It's normally hosted on following URL. Replace the placeholder with the IP Address of application server below:

http://SimphonyAppServerIPAddress:8080/EGateway/SimphonyPosApiWeb.asmx

To create a stub or proxy for client application in order to integrate with TS web service, software engineers can add a reference to this web service. Later, the URL can be changed to point at the instance that's hosted on workstation.

4 Quick Installation Roadmap

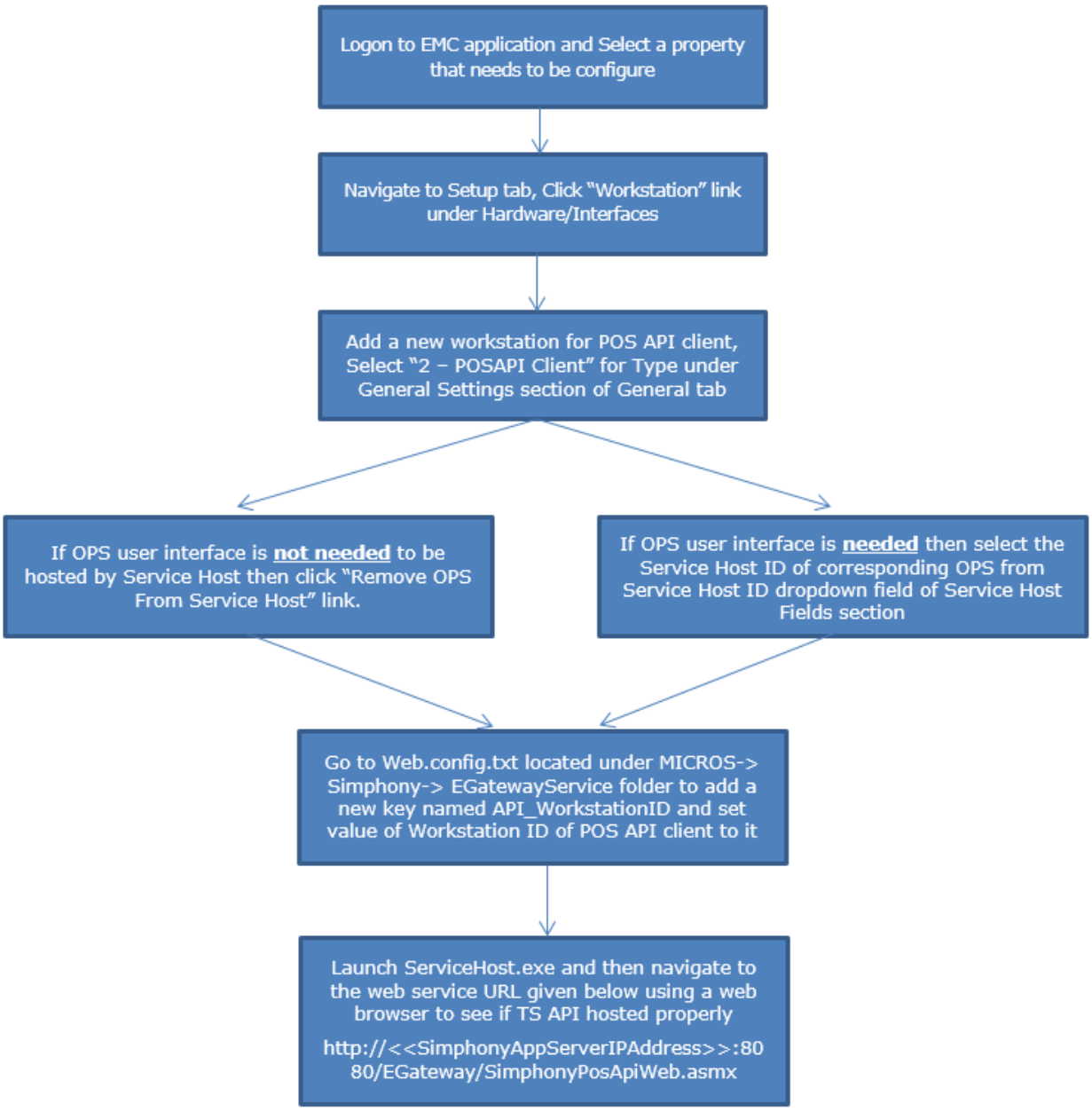

Figure 4-1 Quick Installation Roadmap

5 Error Logging

Transaction Services API writes all errors and other informational messages to a flat file under following folder of POS workstation for diagnostics purpose.

<*ROOT_INSTALL_DRIVE*>\MICROS\Simphony\WebServer\wwwroot\EGateway\EgatewayLog\

Example path: C:\MICROS\Simphony\WebServer\wwwroot\EGateway\EgatewayLog\

6 TS API Class Hierarchy

Table 6-1 Transaction Services Class Hierarchy

ITransactionServices Interface	The main interface that supports transaction related operations (e.g., calculating transaction totals, posting a transaction to create a guest check in the POS database, adding items to an existing guest check, voiding a transaction/check, and retrieving status of any print job already submitted)
IGetCheckInfo Interface	Interface that provides support for fetching all open guest checks, retrieving printed lines of guest check and configuration related information from the POS database
SimphonyPosAPI_CheckSummary Structure	Structure that defines summary details of a guest check
SimphonyPosAPI_OpenChecks Structure	Structure that represents check summary of all open checks
SimphonyPosAPI_GuestCheck Structure	Structure that defines Guest Check details like check id, order type, guest count, table number etc.
SimphonyPosAPI_MenuItem Structure	Structure that holds the definition of Menu Item and it's Condiments
SimphonyPosAPI_MenuItemDefinition Structure	Structure that defines details of a menu item like menu item object number, price, discount etc.
SimphonyPosAPI_ComboMeal Structure	Structure that defines a Combo Meal (main and side items)
SimphonyPosAPI_Discount Structure	Structure used to represent a discount in the Simphony POS system
SimphonyPosAPI_SvcCharge Structure	Structure used to represent a Service Charge in the Simphony POS system. This has details like service charge amount or percentage etc.
SimphonyPosAPI_EPayment Structure	Structure that defines Advanced Electronic Payment details like credit card account, tip amount, cash back amount etc.
SimphonyPosAPI_TmedDetailItemEx Structure	Structure to represent a tender media which has mode of payment
SimphonyPosAPI_TotalsResponse Structure	Structure that holds various totals like subtotal, due amount, tax amount and automatic service charges etc. of a transaction.
SimphonyPosApi_ConfigInfoResponse Structure	Structure that holds configuration details of items like menu item definitions, menu item price, currency, discounts, employees, order type, revenue centers, tender media, service charge etc.
SimphonyPosApi_CheckPrintResponse Structure	Structure that holds the response of Get Printed Check method call. This structure holds operation (i.e., success/failure) results along with printed check lines.
SimphonyPrintApi_PrintJobStatus Structure	Structure that holds the response of Get Printed Job Status method call. This structure holds the operation result (i.e., success/failure) along with details of print jobs and status code and error/success message.
SimphonyPosAPI_OperationalResult Structure	Structure used to represent result (i.e., success or failure) of a method call. In case of failure, this structure will provide error code along with error message that tells the cause of failure.

7 Supported Operations

Transaction Related Operations

TS API provides support for four different transaction related POS operations. One web method is exposed to support each of those operations. All those web methods are explained in the table below.

Table 7-1 Transaction Related Operations Public Member Functions

void **CalculateTransactionTotals** (string vendorCode, ref ARRAY(SimphonyPosAPI_MenuItem) ppMenuItems, ref ARRAY(SimphonyPosAPI_ComboMeal) ppComboMeals, ref SimphonyPosAPI_SvcCharge pServiceChg, ref SimphonyPosAPI_Discount pSubTotalDiscount, int revenueCenterObjectNum, short orderType, int employeeObjectNum, ref SimphonyPosAPI_TotalsResponse pTotalsResponse) *Calculates total amounts of a transaction by considering all items (e.g. menu items, service charge, item discounts and check level discounts etc.) added to the transaction. This method can be used to fetch price of any menu item from Simphony POS database too.*
void **PostTransactionEx** (string vendorCode , ref SimphonyPosAPI_GuestCheck pGuestCheck, ref ARRAY(SimphonyPosAPI_MenuItem) ppMenuItems, ref ARRAY(SimphonyPosAPI_ComboMeal) ppComboMeals, ref SimphonyPosAPI_SvcCharge pServiceChg, ref SimphonyPosAPI_Discount pSubTotalDiscount, ref SimphonyPosAPI_TmedDetailItemEx pTmedDetailEx, ref SimphonyPosAPI_TotalsResponse pTotalsResponse, ref ARRAY(string) ppCheckPrintLines, ref ARRAY(string) ppVoucherOutput) *Creates a guest check in the Simphony POS database for a given transaction. One or more items like menu, discount, service charge and tender can be added to the created guest check later with the help of AddToExistingCheckEx method explained below.*
void **AddToExistingCheckEx** (string vendorCode , ref SimphonyPosAPI_GuestCheck pGuestCheck, ref ARRAY(SimphonyPosAPI_MenuItem) ppMenuItems, ref ARRAY(SimphonyPosAPI_ComboMeal) ppComboMeals, ref SimphonyPosAPI_SvcCharge pServiceChg, ref SimphonyPosAPI_Discount pSubTotalDiscount, ref SimphonyPosAPI_TmedDetailItemEx pTmedDetailEx, ref SimphonyPosAPI_TotalsResponse pTotalsResponse, ref ARRAY(string) ppCheckPrintLines, ref ARRAY(string) ppVoucherOutput) *Adds one or more items (e.g. menu item, discount, service charge, tender media etc.) to an existing open guest check*
void **VoidTransaction** (string vendorCode , ref SimphonyPosAPI_GuestCheck pGuestCheck)

Voids each and every item (e.g. menu, discount, service charge and tender etc) of a given guest check.

void **CheckPrintJobStatus** (string vendorCode, int ppJobId,
ref SimphonyPosApi.SimphonyPrintApi_PrintJobStatus ppJobStatus)

Gets status of a specified print job. The possible statuses of a print job are Pending, Completed, Aborted and Failed.

Details of each of these operations are provided in the following section.

Calculate Totals of a Transaction

Table7-2 Calculating Transaction Totals

```
void CalculateTransactionTotals
(
    String                                  vendorCode,
    ref ARRAY(SimphonyPosAPI_MenuItem)      ppMenuItems,
    ref ARRAY(SimphonyPosAPI_ComboMeal)
                                            ppComboMeals,
    ref SimphonyPosAPI_SvcCharge            pServiceChg,
    ref SimphonyPosAPI_Discount             pSubTotalDiscount,
    int
                                            revenueCenterObjectN
  um,
    short                                   orderType,
    int                                     employeeObjectNum,
    ref SimphonyPosAPI_TotalsResponse       pTotalsResponse
)
```

Business Purpose

User wants to know total amounts of a transaction without creating a guest check in Simphony POS database. This can also be used to fetch price of one or more menu items from POS database.

Method Description

This method determines total amounts of a desired transaction. The caller has to pass the details of menu items, combo meals, service charge and discount that need to be applied on the check along with other supporting details like revenue center id, order type and id of employee who wants to perform this transaction.

This method calculates subtotal, discount, tax, service charges, and due amount to be paid by the customer for given transaction. During calculation, this method will take care of applying applicable automatic discounts and services charges as well. This method will not create any guest check in the POS database.

This method can be used to find out the price of one or more menu items or combo meals too. In this case, parameters related to service charge, discount can be omitted.

Parameter

vendorCode	-	Vendor code for license validation (pass empty value for version 2.7 MR3 or later)
ppMenuItems	-	List of menu items with required condiments and item discount
ppComboMeals	-	List of combo meals (main and side items)
pServiceChg	-	Service charge that needs to applied
pSubTotalDiscount	-	Discount that needs to applied at check level
revenueCenterObjectNum	-	Object number of given revenue center
orderType	-	Type of Order (e.g., Dine-in, Dine-out etc.)
employeeObjectNum	-	Object number of employee who performs this operation
pTotalsResponse	-	Structure to hold the response (output parameter)

Return Value

Void. Result is encapsulated in the *pTotalsResponse* reference parameter mentioned above.

Create a Guest Check

Table 7-3 Create a Guest Check

```
void PostTransactionEx
(
    String                                  vendorCode,
    ref SimphonyPosAPI_GuestCheck           pGuestCheck,
    ref ARRAY(SimphonyPosAPI_MenuItem)      ppMenuItems,
    ref ARRAY(SimphonyPosAPI_ComboMeal)     ppComboMeals,
    ref SimphonyPosAPI_SvcCharge            pServiceChg,
    ref SimphonyPosAPI_Discount
                    pSubTotalDiscount,
    ref SimphonyPosAPI_TmedDetailItemEx     pTmedDetailEx,
    ref SimphonyPosAPI_TotalsResponse
                    pTotalsResponse,
    ref ARRAY(string)                       ppCheckPrintLines,
    ref ARRAY(string)                       ppVoucherOutput
)
```

Business Purpose

Order taker needs to ring in a transaction and create a guest check for a customer by feeding all menu item details, discounts, service charges, tender etc.

Method Description

This method posts a transaction to POS database to create a guest check. This calculates totals of the transaction before it creates guest check in the POS database. The result on transaction totals can be found in pTotalsResponse parameter. Details like menu items, service charge, discount, and tender media that need to be added to the guest check should be passed as input to appropriate parameters. When tender media of type “Service Total” is supplied, system will create a guest check but keep it in “Open” state. However, if a tender media with payment is supplied, then the check will get created and closed at the end of the call. It’s possible to create future check (a.k.a. auto fire check), if needed, by mentioning appropriate value for CheckStatusBits property. The

system will take care of applying all applicable automatic discounts and service charges while creating the guest check. Also, it'll calculate the tax amount based on how order type is configured in EMC. The details like Check Number and Check Sequence Number of the created check will be filled in pGuestCheck parameter. System will not create the guest check if payment or any other interim operation fails. This method will print the guest check and credit voucher if it's configured to do so in EMC for the given workstation.

This method supports printing to local and remote Order Devices. When a check is created by this method, Menu Items will print on local or remote Order Devices based on the default Workstation definition and assigned Menu Item Print Class.

The printed check details will be filled in ppCheckPrintLines while credit voucher details are filled in ppVoucherOutput parameter.

Parameter

vendorCode	-	Vendor code for license validation (pass empty value for version 2.7 MR3 or later)
pGuestCheck	-	The guest check structure to pass input data like Check Date To Fire, Employee object number, guest count, order type, revenue center object number and check table object number. Other properties of this structure like CheckNum, check sequence number, check info lines and operation results will get populated as results by this method.
ppMenuItems	-	List of menu items (with condiments) to be added to given check
ppComboMeals	-	List of combo meals (main and side items) to be added to given check
pServiceChg	-	Service charge to be applied to the guest check
pSubTotalDiscount	-	Discount that needs to be applied at check level
pTmedDetailEx	-	Desired tender and optionally e-payment information to be added to the guest check
pTotalsResponse	-	Structure to hold the response or result (output parameter)
ppCheckPrintLines	-	Printed texts of guest check (output parameter)
ppVoucherOutput	-	Raw credit voucher (output parameter)

Return Value

Void. The *OperationalResult* property of pGuestCheck parameter will hold operation results. Also, *ppCheckPrintLines* will hold printed lines of guest check while *ppVoucherOutput* holds printed lines of credit voucher.

Add Items to Existing Guest Check

Table 7-4 Adding Items to Existing Check

```
void AddToExistingCheckEx
(
    String                                  vendorCode,
    ref SimphonyPosAPI_GuestCheck           pGuestCheck,
    ref     ARRAY(SimphonyPosAPI_MenuItem)
        ppMenuItems,
    ref   ARRAY(SimphonyPosAPI_ComboMeal)
        ppComboMeals,
    ref SimphonyPosAPI_SvcCharge            pServiceChg,
    ref SimphonyPosAPI_Discount
        pSubTotalDiscount,
    ref SimphonyPosAPI_TmedDetailItemEx     pTmedDetailEx,
    ref SimphonyPosAPI_TotalsResponse       pTotalsResponse,
    ref ARRAY(string)                       ppCheckPrintLines,
    ref ARRAY(string)                       ppVoucherOutput
)
```

Business Purpose

This method will be of useful in any of following business scenarios.

- Customer likes to add one or more menu items/combo meals to an existing open guest check
- Customer likes to make partial or full payment on an existing open guest check
- Customer provides a discount coupon that needs to be applied on an existing open guest check
- Order taker likes to apply a service charge on an existing open guest check

Method Description

This method is to add one or more items (i.e., menu item, combo meal, subtotal discount, service charge and tender media) to an existing open guest check. This method cannot add any items to a closed check. This method can be used when the customers like to add one or more menu items to a check that was already created during initial order.

When this method is invoked, Guest Check structure (i.e., *pGuestCheck*) will be interrogated and changed where appropriate. The check sequence number and the check number will not be changed, but the existing Check ID field will always be changed to reflect the new Check ID. The Order Type will also be changed to reflect the new Order Type passed into the parameter.

Provided below are the fields of *SimphonyPosApi_GuestCheck* that may be modified and updated to reflect the new information during the execution of this method:

1. CheckID
2. CheckTableObjectNum (when supported)
3. CheckOrderType
4. CheckEmployeeObjectNum
5. CheckDateToFire
6. pCheckInfoLines

The following fields are not be modified by this method

1. CheckNum
2. CheckSeq
3. CheckRevenueCenterObjectNum

Parameter

vendorCode	-	Vendor code for license validation (pass empty value for version 2.7 MR3 or later)
pGuestCheck		-A structure to pass input data like Employee object number, guest count, Order type, Revenue center object number, Check Date To Fire and dining table object number. Other properties of this structure like CheckNum, check sequence number, check info lines and operation results will get populated as results by this method.
ppMenuItems	-	List of menu items (with condiments) to be added to given guest check
ppComboMeals	-	List of combo meals (main and side items) to be added to given guest check
pServiceChg	-	Service charge that needs to be applied to the check
pSubTotalDiscount	-	Discount that needs to be applied at check level
pTmedDetailEx	-	Desired tender and optionally e-payment information to be added to given check
pTotalsResponse	-	Reference to total response structure that'll hold results (output parameter)
ppCheckPrintLines	-	Printed texts of guest check (output parameter)
ppVoucherOutput	-	Raw credit voucher (output parameter)

Return Value

Void. The *OperationalResult* property of *pGuestCheck* parameter will hold operation results. Also, *ppCheckPrintLines* will hold printed lines of guest check while *ppVoucherOutput* holds printed lines of credit voucher.

Void All Items of an Open Guest Check

Table 7-5 Void All Items of an Open Guest Check

```
void VoidTransaction
(
```

```
        String                              vendorCode,
        ref SimphonyPosAPI_GuestCheck       pGuestCheck
    )
```

Business Purpose

Customer likes to cancel his/her order for any reason (e.g. Incorrect order, took too long to prepare etc.)

Method Description

This method voids all of the items (e.g., menu item, tender media, service charge, discount etc.) in the given guest check and then closes the check. This method works only if the guest check is in open state. This method will throw an exception if the check is found to be closed.

Input data needed for only two properties of *pGuestCheck* (mentioned below) to perform this operation. Other properties of *pGuestCheck* can be left with its default value while invoking this method.

- CheckNum
- CheckSeq

Parameter

vendorCode	-	Vendor code for license validation (pass empty value for version 2.7 MR3 or later)
pGuestCheck	-	Details of the guest check to be voided

Return Value

Void. The *OperationalResult* property of *pGuestCheck* parameter holds operation results.

Check Status of a Print Job

Table 7-6 Check Status of a Print Job

```
    void CheckPrintJobStatus
    (
        string                                          vendorCode,
        int                                             ppJobId,
        ref     SimphonyPosApi.SimphonyPrintApi_PrintJobStatus
                                                        ppJobStatus
    )
```

Business Purpose

The cashier would like to know the status of any specific print job (e.g., guest check print, credit voucher print) of a transaction

Method Description

This method gets the status of a specified print job. This also gets the complete list of print jobs and stores it in *PrintJobList* field of parameter *ppJobStatus*. Below is the exhaustive list of job status:

1. Job Pending

2. Job Complete
3. Job Aborted
4. Job Sent to backup printer
5. Job Failed
6. Job Not found

Parameter

vendorCode - Vendor code for license validation (pass empty value for version 2.7 MR3 or later)
ppJobId - Id of print job for which status needs to be retrieved
ppJobStatus - Status of the print job (output parameter)

Return Value

Void. Result is encapsulated in the *ppJobStatus* reference parameter.

Guest Check and Configuration Related Operations

TS API provides support for two check related and one configuration related operations. One web method is exposed to support each of those operations. All of those web methods are explained in the table below.

Table 7-7 Public Member Functions

void **GetOpenChecks** (string vendorCode, int employeeObjectNum, ref SimphonyPosAPI_OpenChecks openChecks) *Gets summary of all open guest checks from all revenue centers of the property from Simphony POS database*
void **GetOpenChecksEx** (string vendorCode, int employeeObjectNum, ref SimphonyPosAPI_OpenChecks openChecks) *Gets summary of all open guest checks from all revenue centers of the property from Simphony POS database. The only difference between GetOpenChecks and this method is that GetOpenChecks populates CheckRevenueCenterObjectNum member with ID of revenue center while this method populates Object Number of revenue center.*
void **GetOpenChecksByRVC** (string vendorCode, int employeeObjectNum, int revenueCenterObjectNum, ref SimphonyPosAPI_OpenChecks openChecks) *Gets summary of open guest checks for a specific revenue center from Simphony POS database*
void **GetPrintedCheck** (string vendorCode, int CheckSeq, int EmplObjectNum, int TmedObjectNum, ref SimphonyPosApi_CheckPrintResponse ppCheckPrintLines) *Gets printed texts of an open guest check*
void **GetConfigurationInfo** (string vendorCode, int employeeObjectNum, int[] configurationInfoType, int revenueCenter, ref SimphonyPosApi_ConfigInfoResponse configInfoResponse) *Gets configured information for one or more types from POS database*

Details of each of these operations are provided in the following sections.

Get Summary of All Open Guest Checks

Table 7-8 Get Summary of All Open Guest Checks

void GetOpenChecks	
(	
string	***vendorCode,***
int	***employeeObjectNum*,**
ref SimphonyPosAPI_OpenChecks	***openChecks***
)	

Business Purpose

User wants to see summary of all open Guest Checks from all revenue centers of the property.

Method Description

This method gets summary of all open Guest Checks from all revenue centers of the property from the POS database. Guest Checks that are created by a specific employee can be fetched by passing appropriate value to *employeeObjectNum* parameter. However, when 0 is passed to *employeeObjectNum*, it will fetch all the open Guest Checks irrespective who created the check. The *CheckRevenueCenterObjectNum* field of *openChecks.SimphonyPosApi_CheckSummary* structure is mislabeled and it will hold value of Revenue Center ID instead of Revenue Center Object Number. If this field is expected to hold Object Number of Revenue Center then the new method named *GetOpenChecksEx* can be used instead.

Parameter

vendorCode - Vendor code for license validation (pass empty value for version 2.7 MR3 or later)

employeeObjectNum - Object number of employee to filter open Guest Checks based on who created it. Pass specific employee object number to fetch open checks created by that specific employee. Pass zero to fetch all open checks irrespective of who created the check.

openChecks - Holds open checks retrieved from the POS database (output parameter)

Return value

Void. Result is encapsulated in the *openChecks* reference parameter.

Get Open Guest Checks with RVC Object Number

Table 7-9 Get Open Guest Checks with RVC Object Number

void GetOpenChecksEx	
(	
string	***vendorCode,***
int	***employeeObjectNum*,**
ref SimphonyPosAPI_OpenChecks	***openChecks***
)	

Business Purpose

User wants to see summary of all open Guest Checks from all revenue centers of the property, and like to have object number of revenue center (instead of ID) for each guest check. Object number of revenue center is different from that of ID.

Method Description

This method is another version of *GetOpenChecks* method that's explained in the previous section. This was introduced later in version 2.7 MR5 to retrieve all open Guest Checks from all revenue centers of the property. The only difference compared to *GetOpenChecks* method is that the *CheckRevenueCenterObjectNum* property of *openChecks.SimphonyPosApi_CheckSummary* holds the Revenue Center Object Number instead of the Revenue Center ID. All open Guest Checks created by a specific employee can be fetched by passing appropriate value to *employeeObjectNum* parameter. However, when 0 is passed to *employeeObjectNum*, it will fetch all open Guest Checks irrespective who created them.

Parameter

vendorCode - Vendor code for license validation (pass empty value for version 2.7 MR3 or later)

employeeObjectNum - Employee object number to filter open Guest Checks based on who created it. Pass specific employee object number to fetch open checks created by that specific employee. Pass zero to fetch all open checks irrespective of who created them.

openChecks - Holds open checks retrieved from the database (output parameter)

Return Value

Void. Result is encapsulated in the *openChecks* reference parameter.

Get Open Guest Checks from a Specific RVC

Table 7-10 Get Open Guest Checks from a Specific RVC

```
Void    GetOpenChecksEx
(
String                              vendorCode,
Int                             employeeObjectNum,
Int                             revenueCenterObjectNum
ref SimphonyPosAPI_OpenChecks   openChecks
)
```

Business Purpose

User wants to see a summary of the open Guest Checks from a specific revenue center.

Method Description

This method was introduced later in version 2.7 MR4 to get all open Guest Checks from a specific revenue center from Simphony POS database. All open Guest Checks created by a specific employee in a specific revenue center can be fetched by passing appropriate value to *employeeObjectNum* and *revenueCenterObjectNum* parameters. However, when 0 passed for *employeeObjectNum*, it will fetch all the open Guest Checks from specified revenue center irrespective of who created it. Also, note that the other two

related methods named *GetOpenChecks* and *GetOpenChecksEx* returns a summary of all open checks from all revenue centers of the property.

Parameter

vendorCode	- Vendor code for license validation (pass empty value for version 2.7 MR3 or later)
employeeObjectNum	- Employee object number to filter open Guest Checks based on who created it. Pass specific employee object number to fetch open checks created by that specific employee. Pass zero to fetch all open checks irrespective of who created them.
revenueCenterObjectNum	- Object number of revenue center for which checks needs to be retrieved
openChecks	- Holds open checks retrieved from the database (output parameter)

Return value

Void. Result is encapsulated in the *openChecks* reference parameter.

Get Printed Texts of a Guest Check

Table 7-11 Get Printed Texts of a Guest Check

```
void  GetPrintedCheck
(
    string                                          vendorCode,
    int                                             CheckSeq,
    int
                    EmplObjectNum,
    int                                             TmedObjectNum,
    ref SimphonyPosAPI_CheckPrintResponse
                                                    ppCheckPrintLines
)
```

Business Purpose

User wants to reprint a Guest Check for a customer using an external printer.

Method Description

This method gets printed texts of an open Guest Check. This method will work on only open guest check and will throw exception in case of closed guest check.

This method requires the tender media as input because it has several printing options that assist in the formatting of the final Guest Check.

Parameter

vendorCode	-	Vendor code for license validation (pass empty value for version 2.7 MR3 or later)
CheckSeq	-	Check Number of the guest check (and not the check sequence number as the name implies)
EmplObjectNum	-	Object number of the employee who wants to perform this operation
TmedObjectNum	-	Object number of the Tender Media to print the check with. Service Total is the only type of Tender Media that's permitted as of now
ppCheckPrintLines	-	This holds an array of printed lines of the check along with response code

Return value

Void. Result is encapsulated in the *ppCheckPrintLines* reference parameter.

Get Configured Information

Table 7-12 Get Configured Information

```
void  GetConfigurationInfo
(
    string                                  vendorCode,
    int                                     employeeObjectNum,
    ARRAY(int)                              configurationInfoType,
    int                                     revenueCenter,
    ref                         SimphonyPosAPI_ConfigInfoResponse
        configInfoResponse
)
```

Business Purpose

User wants to fetch Configured Information of a particular Revenue Center from Simphony POS database.

Method Description

This method fetches information that is configured for one or more type at enterprise/server side. Example configuration type is Menu Item Definition, Menu Item Price, Menu Item Class, Service Charge, Tender Media, Order Type, Currency, Employee etc.

Parameter

vendorCode	-	Vendor code for license validation (pass empty value for version 2.7 MR3 or later)
employeeObjectNum	-	Object Number of employee who wants to perform this operation
configurationInfoType	-	Array of information types for which details need to be fetched from POS database
revenueCenter	-	Object number of revenue center
configInfoResponse	-	Structure that holds the results. This holds configuration information along with operation result. There's one field for each information type requested.

Return Value

Void. Result is encapsulated in the *configInfoResponse* reference parameter.

8 Structure Reference

SimphonyPosAPI_CheckSummary

This structure is to encapsulate summary details of a check.

Table 8-1 Check Summary Public Attributes

int	**CheckSeq** *Check sequence is a number that identifies a check in the Simphony POS database. The sequence number will be assigned to a check when it's created by the system.*
int	**CheckNum** *Check number is a number to identify a check in a particular workstation. This number will be assigned to a check by the system when the check is created. Minimum and maximum range for check number can be configured in EMC for any workstation.*
int	**CheckEmployeeObjectNum** *ID of employee who created the check*
int	**CheckRevenueCenterObjectNum** *ID of revenue center this check is currently active*
int	**CheckLastWorkstationOwner** *Object number of workstation that owned the check last time*
int	**CheckCurrentlyOpenOnWorkstation** *Object number of workstation that has this check currently opened*
int	**CheckTableObjectNum** *Object number of dining table for which the check created.*
int	**CheckTableGroup** *ID of table group in which dining table of this check falls under*
int	**CheckOrderType** *Order type ID of the check. E.g. Dine In and Eat Out*
string	**CheckID** *Name to identify a check. Duplicate check names on open checks are not allowed.*
string	**CheckTotalDue** *Due or balance amount to be paid by the customer for the check*
DateTime	**CheckLastServiceTime** *The time when the check was submitted last time to POS database*
DateTime	**CheckOpenTime** *The time when the check was opened/created on POS database*
DateTime	**CheckAutoFireTime** *The time when check will be fired*
short	**CheckInTraining**

A flag that indicates whether the check is opened on training mode or not. Always 0 is populated.

short CheckInsufficientBeverage

A flag that indicates whether insufficient beverage found on the check (i.e. beverage count is less than total guest count). Always 0 is populated.

short CheckTransferedToDriver

A flag that indicates status of the check as having been assigned to a driver. This is no longer in use. Always 0 is populated.

short CheckIsDelayedOrder

A flag that indicates status of the check as being a Delayed Order. This is no longer in use. Always 0 is populated.

short CheckIsFutureOrder

A flag that indicates status of this check as having been assigned to a driver. Always 0 is populated.

SimphonyPosAPI_OpenChecks

This structure is to encapsulate details of all open checks

Table 8-2 Open Checks Public Attributes

ARRAY(SimphonyPosAPI_CheckSummary) CheckSummary

A structure to hold summary of all open checks

SimphonyPosAPI_OperationalResult OperationalResult

A structure that indicates whether current operation succeeded or not. In case of failure, this will have appropriate error code and error message

SimphonyPosAPI_GuestCheck

This structure is to encapsulate the details of a guest check.

The Guest Check structure is a collection of elements that are passed as a parameter. This shared structure is used to communicate key elements of the transaction to the API and for the API to return key elements to the API consumer.

Table 8-3 Guest Check Public Attributes

string CheckID

Name to identify a check. Duplicate check names on open checks are not allowed

int CheckTableObjectNum

Object number of dining table for which the check opened

int CheckRevenueCenterID

Object number of revenue center

int CheckOrderType

Order type ID of the check. E.g. Dine In and Eat Out

int CheckEmployeeObjectNum

Object number of employee who opened the check

int CheckSeq

Check sequence is a number that identifies a check in the POS database. The number is assigned to the check is opened. This is used as a parameter while adding items to an existing check.
int CheckNum *Check number is a number to identify a check in a particular workstation. This number will be assigned to a check by the system when the check is created. Minimum and maximum range for check number can be configured in EMC for any workstation.*
DateTime CheckDateToFire *Time when check should fire. This will permit an order to be delayed on the current business date*
int CheckGuestCount *Total number of guest in a transaction*
int CheckStatusBits *Check status identifier. e.g., "Rush Order" or "VIP".*
ARRAY(string) PCheckInfoLines *Information lines that are added to guest check*
ARRAY(int) PPrintJobIds *List of print job ID that resulted from the transaction. The method CheckPrintJobStatus can be used to get the status of any print job later*
SimphonyPosAPI_OperationalResult OperationalResult *A structure that indicates whether current operation succeeded or not. In case of failure, this will have appropriate error code and message*

SimphonyPosAPI_MenuItem

This structure is to encapsulate the details of a menu item along with its condiments.

The Menu Item is comprised of the desired Main item and an array of Condiments. An example may be a Cheeseburger (Main item), Well Done and Extra pickles (Condiment array).

Table 8-4 Menu Item Public Attributes

SimphonyPosAPI_MenuItemDefinition MenuItem *Structure that defines details of a main menu item. The details include object number of menu item, price, discount etc.*
ARRAY(SimphonyPosAPI_MenuItemDefinition) Condiments *List of a structure that defines details of condiment added to menu item*

SimphonyPosAPI_MenuItemDefinition

This structure is to encapsulate the details of a menu item definition.

Table 8-5 Menu Item Definitions Public Attributes

int MiObjectNum *Object number of given menu item*

int MiMenuLevel

Main level to be used while picking up a menu definition from definition list. This must be a value between 1 and 8 (if not 0). When 0 is specified, system will pick up first menu definition irrespective of whether it's active or not on given Main level

int MiSubLevel

Sub level to be used while picking up a menu definition from definition list. This must be a value between 1 and 8 (if not 0). When 0 is specified, system will pick up first menu definition irrespective of whether it's active or not on given Sub level

int MiPriceLevel

Sequence number to be used while picking up a price definition from the list. ***This is not currently supported in Transaction Services web service****. That is, price definition will always be picked up based on the value of Sub level or Main level mentioned above*

string MiOverridePrice

Price to override default value of the item. This field can be left empty if default price is desired. If left empty this will be populated with default price by this method.

string MiWeight

Weight of given item. ***This is not currently supported in the API.***

string MiReference

A text that needs to be added as reference to given menu item

SimphonyPosApi_Discount ItemDiscount

Discount that needs to applied to given menu item

SimphonyPosAPI_ComboMeal

This structure is to encapsulate the details of a combo meal (main and side menus)

When ordering Combo Meals, TS API is very strict in checking all of the Combo Meal linkage. The Combo Meal Menu Item passed along, must be linked to a Combo Meal Object Number. Additionally, the Combo Meal Side Items that are passed along must be correctly linked to a Combo Meal as defined in the target database. This means that side items must be passed in order. All items in orders must be filled correctly for Combo Meals.

Table 8-6 Combo Meal Public Attributes

SimphonyPosAPI_MenuItem ComboMealMenuItem *Combo Meal Menu Item (e.g. Burger Combo)*
SimphonyPosAPI_MenuItem ComboMealMainItem *Combo Meal Main Item (e.g. Hamburger)*
ARRAY(SimphonyPosAPI_MenuItem) SideItems *Combo Meal Side Items (e.g. French Fries, Coke etc.)*
int ComboMealObjectNum *Combo Meal Object Number*

SimphonyPosAPI_Discount

This structure is to encapsulate the details of discounts.

Table 8-7 Discount Public Attributes

int DiscObjectNum *Discount Object Number*
string DiscAmountOrPercent *Amount or Percentage to be discounted. API expects value for this property in case of "Open Discount". However, in case of "Closed Discount", discount amount or percent will be taken from POS database with the help of Discount Object Number.*
string DiscReference *Reference text to be added to given discount for reference purpose*

SimphonyPosAPI_SvcCharge

This structure is to encapsulate the details of service charge to be applied on guest check

Table 8-8 Service Charge Public Attributes

int SvcChgObjectNum *Object Number of Service Charge that needs to be applied on guest check*
string SvcChgAmountOrPercent *Amount or percentage to be applied as Service Charge. API expects value for this property in case of "Open Service Charge". However, in case of "Closed Service Charge", the amount or percent will be taken from POS database with the help of Service Charge Object Number.*
string SvcChgReference *Reference text to be added to given Service Charge item*

SimphonyPosAPI_EPayment

This structure is to encapsulate the details of electronic payment on a guest check.

Table 8-9 Electronic Payment Public Attributes

EPaymentDirective **PaymentCommand**

Enumeration on payment method (e.g. credit authorization only, credit authorization and pay, debit authorization only, debit authorization and pay, SVC authorization, SVC redeem etc.). Possible values are:

- NO_E_PAYMENT
- AUTHORIZE_AND_PAY
- DEBIT_AUTHORIZE_AND_PAY

EAccountDataSource **AccountDataSource**

Enumeration on source of payment details (e.g., magnetic stripe, RFID card, manually keyed etc.). Possible values are:

- SOURCE_UNDEFINED
- RFID_TRACK_DATA_RULES
- RFID_M_CHIP_RULES
- MANUALLY_KEYED_TRACK_1_CAPABLE
- MANUALLY_KEYED_TRACK_2_CAPABLE
- MANUALLY_KEYED_NO_CARD_READER

EAccountType **AccountType**

Type of account (e.g., Checking and Savings). Possible values are:

- ACCOUNT_TYPE_UNDEFINED
- CHECKING
- SAVINGS

string **AcctNumber**

Account Number of payment card.

string **AuthorizationCode**

Authorization code of payment card

DateTime **StartDate**

Start Date as mentioned in payment card

short **IssueNumber**

Issue Number as mentioned in payment card

string **Track1Data**

Magnetic stripe data for Track 1

string **Track2Data**

Magnetic stripe data for Track 2

string **Track3Data**

Magnetic stripe data for Track 3

string **BaseAmount**

Base Amount to be debited. This doesn't include tip or cash back amount.

string	**TipAmount** *Amount to be debited for Tip*
string	**CashBackAmount** *Cash Back amount*
string	**KeySerialNum** *Debit Key Serial Number for given transaction. Maximum length is 20 characters.*
string	**DeviceId** *Device Identifier*
DateTime	**ExpirationDate** *Expiration date as mentioned in payment card.*
string	**PinBlock** *Pin Number of payment card in encrypted format. This is used only with debit card payment.*
string	**CVVNumber** *CVV (i.e. Card Verification Value) Number of payment card*
string	**AddressVerification** *Address for verification*
string	**InterfaceName** *Interface name of Stored Value Card*
string	**SvcResponse** *Stored Value Card response message. This contains descriptive error message in case of payment failure.*
string	**SvcAccountType** *Stored Value Account. Maximum 32 characters.*

SimphonyPosAPI_TmedDetailItemEx

This structure is to encapsulate the details of tender media for payment operation

Table 8-10 Tender Media Public Attributes

int	**TmedObjectNum** *Object number of tender media chosen for payment*
string	**TmedPartialPayment** *This indicates the amount tendered by the customer in cash for payment. This amount does not include tips. Leave this field empty in case of paid-in-full. This field is applicable for only cash payment.*
string	**TmedReference** *Tender Media reference information*
SimphonyPosAPI_EPayment	**TmedEPayment** *Electronic Payment details*

SimphonyPosAPI_TotalsResponse

This structure is to encapsulate the details on totals of a transaction.

Table 8-11 Totals Response Public Attributes

string TotalsSubTotal *Subtotal amount of current transaction*
string TotalsTaxTotals *Total tax applied on current transaction*
string TotalsOtherTotals *Service Charge applied on current transaction*
string TotalsAutoSvcChgTotals *Automatic Service Charge applied on current transaction*
string TotalsTotalDue *Total amount due*
SimphonyPosAPI_OperationalResult OperationalResult *A structure that indicates whether current operation has succeeded or not. In case of failure, this will have appropriate error code and message*

SimphonyPosAPI_ConfigInfoResponse

This structure is to encapsulate the details of configured details for menu, price, currency, discounts, employees, order type, revenue center, tender media, service charge etc.

Table 8-12 Configuration Details Public Attributes

ARRAY(EConfigurationInfoType) ConfigInfoType *List of type of configuration information for which details need to be fetched from Simphony POS database (e.g. menu item definition, service charge definition, discount definition etc). Possible enumeration values are:* • UNDEFINED = 0 • MENUITEMDEFINITIONS = 1 • MENUITEMPRICE = 2 • MENUITEMCLASS =3 • SERVICECHARGE = 4 • DISCOUNTDEFINITIONS = 5 • TENDERMEDIA = 6 • ORDERTYPE = 7 • FAMILYGROUP = 8 • MAJORGROUP = 9 • REVENUECENTERPARAMETER = 10 • REVENUECENTERS = 11 • INTERFACES = 12 • MENUITEMMASTERS = 13 • SERVINGPERIODS = 14 • CURRENCY = 15

- VERSION = 16
- EMPLOYEES = 17
- TABLES = 18
- LANGUAGEINFORMATION = 19
- MENULEVEL = 20

string **LanguageInformation**

Details of languages (e.g. English, Spanish) that are configured via EMC

string **ServingPeriod**

Details of serving period (e.g. Breakfast 4am to 11am)

string **ServiceCharge**

Details of service charges (e.g. service charge amount/percent, tips etc.)

string **Version**

Current version of the Transaction Services web service (e.g. 2.700.0.77)

string **MenuLevel**

Details of menu levels (e.g. Main, Sub and Custom levels)

string **TenderMedia**

Details of tender media configured in EMC for payment (e.g. Cash, and Credit Cards)

string **Tables**

Details of dining tables configured in EMC for given property

string **RevenueCenters**

Details of revenue centers configured in EMC for given property

string **RevenueCenterParameter**

Details of revenue center parameters that are configured in EMC (e.g. secondary print language, minimum and maximum check number, database update frequency, option bits etc.)

string **OrderType**

Details of order types (e.g. dine-in, dine out etc.) configured in EMC for given property

string **MajorGroup**

Details of all major groups configured in EMC for menu items (e.g. Food, Beverages)

string **MenuItemClass**

Details of all menu item classes (e.g. tax class, sales, discount and service charge itemizers, pricing calculation etc) as configured in EMC

string **MenuItemPrice**

Details of all menu item price records (e.g. menu item definition id, price, preparation cost etc.) configured in EMC

string **Employees**

Details of all employees configured in EMC at enterprise level

string **Discounts**

Details of all discounts configured in EMC at enterprise level

string **Currency**

Details of all currencies (e.g. US Dollar, Peso etc) configured in EMC at enterprise level

string **MenuItemDefinitions**

Details of all menu item definitions configured in EMC for given revenue center

string **FamilyGroups**

Details of all family groups (i.e. category of menu items) configured in EMC

string **Interfaces**

Details of all interfaces configured in EMC at enterprise level

string **MenuItemMasters**

Details of all menu item master records (i.e. property level menu item record)

SimphonyPosApi_OperationResult **OperationalResult**

A structure that indicates whether current operation has succeeded or not. In case of failure, this will have appropriate error code and message

SimphonyPosAPI_CheckPrintResponse

This structure is to encapsulate the details of response on printing a guest check

Table 8-13 Check Print Response Public Attributes

ARRAY(string) **CheckPrintLines**

Printed lines of a guest check

SimphonyPosAPI_OperationalResult **OperationalResult**

A structure that indicates whether current operation has succeeded or not. In case of failure, this will have appropriate error code and message

SimphonyPosAPI_PrintJobStatus

This structure is to encapsulate the details of response on retrieving status of a print job

Table 8-14 Print Job Status Public Attributes

SimphonyPrintApi_Status **Status**

An enumerator that indicates current status of a specified print job. Possible values are:

- JobPending = 0
- JobComplete = 1
- JobAborted = 2
- JobSentToBackup = 3
- JobFailed = 4
- JobNotFound = 5

string **StatusMsg**

Current status of a specified print job in string format

string **SystemStatusMsg**

This is for future use. Currently, this holds the status of a specified print job in string format like StatusMsg field

ARRAY(int) PrintJobList

List of print jobs on the POS system

SimphonyPosAPI_OperationalResult OperationalResult

A structure that indicates whether current operation has succeeded or not. In case of failure, this will have appropriate error code and message

SimphonyPosAPI_OperationalResult

This structure is to encapsulate the result of an operation

Table 8-15 Operational Result Public Attributes

bool Success

Indicates whether or not the operation has succeeded. This will be "true" if there is no exception or errors; otherwise false

TransactionServices_ErrorCode ErrorCode

Error code that represents the reason for failure. Possible values are:

- AmountNotEntered,
- AppInitInProgress,
- CCAuthDeclined,
- CCAuthDeclinedWithMessage,
- CCServerDown,
- CheckEmployeeNumberMismatch,
- CheckNotFound,
- CheckListNotFound,
- CheckOpenedOnSystem,
- CheckTableNumberMismatch,
- ComboMealNotFound,
- ConnectionDown,
- DataOutOfRange,
- DetailDoesNotSupportTriggeredEvents,
- DiscountNotFound,
- DiscountAmountRequired,
- DiscountAmountTooLarge,
- DiscountAmountZero,
- DiscountItemNotAllowed,
- DiscountNotAllowedFilterActive,
- DiscountOnParentCombo,
- DuplicateLineNumber,
- EGatewayClientStartError,
- EGatewayClientStopError,
- EGatewayConnectionError,
- EGatewayConnectionNotInPool,
- EGatewayWaitConnectionTimeout,
- EmployeeClockIOStatusMismatch,

- EmployeeIDMismatch,
- EmployeeNotFound,
- EmployeeRVCMismatch,
- ErrorCreatingGuestcheck,
- ErrorInvalidWorkstation,
- ErrorReadingCheck,
- ErrorPickupCheck,
- FailedDataStoreInitialization,
- FailedDbSettingLoad,
- FailedErrorTranslationInitial,
- FailedPostCARequest,
- FailedInitialization,
- FailedLoggerInitialization,
- FailedSecurityAPIInitialization,
- FailedSubmitPrintJob,
- InternalCommunicationError,
- InternalProcessingError,
- InvalidArguments,
- InvaiIdAuthCode,
- InvalidCheckNumber,
- InvalidCreditCardExpirationDate,
- InvalidCreditCardHost,
- InvalidCreditCardNumber,
- InvalidClientName,
- InvalidClosedDays,
- InvalidConfigInfoRequestType,
- InvalidConfigInfoType,
- InvalidCustomerInfo,
- InvalidDetailLine,
- InvalidDetailLineType,
- InvalidEmployeeNumber,
- InvalidGuestCount,
- InvalidLineNumber,
- InvalidMenuItemPrice,
- InvalidOrderTypeNumber,
- InvalidPropertyNum,
- InvalidRvcNum,
- InvalidServingPeriod,
- InvalidTableNumber,
- InvalidTranslationSpecifier,
- ItemDiscountNeedsParentItem,
- LicensingFailed,
- MenuItemOutOfOrder,
- MissingDetailLinesElement,
- MissingTransactionElement,
- MissingTransactionHeaderElement,

- NoRequestHeader,
- NoSalesForDiscount,
- NotImplemented,
- NoSalesToApplyServiceCharge,
- NullInput,
- PaidPartially,
- PaymentAborted,
- PriceMenuItemWithZeroAmount,
- SecurityInitFailed,
- ServiceChargeTaxClassNotFound,
- Success,
- TenderTypeNotFound,
- TransactionEmployeeNotFound,
- TranslationFileNotAvailable,
- UnhandledException,
- UnknownCreditCardType,
- UnknownExceptionCode
- TransactionLocked

string ErrorMessage

Texts that further explains the exception and reason for failure

9 Example and Code Snippets

Calculate Totals of a Transaction

Given below is a scenario in which user wants to find out total amount of a transaction for items that are being ordered by customer.

1. Ring in two menu items
    a. Ring in two condiments to first menu item, and one condiment to second menu item
    b. Override price of first menu item
    c. For first Menu Item, instead of default definition, pick up a specific MI definition based on main and sub menu levels
    d. Apply an “Open” discount to first menu item
    e. Add a reference text to first menu item
2. Ring in a combo meal
3. Apply an “Open” service charge
    a. Add a reference text on service charge
4. Apply a “Closed” discount on subtotal (i.e., check level)
5. Order type is Dine-in

Sample data for the scenario mentioned above:

S. No	Type of Data	Parameter Name	Sample Data
1	Vendor code	*vendorCode*	***yzsroioq***
2	Menu Items and Condiments along with item level Discount	*ppMenuItems*	• Object number of two menu items are **110003** and **110004** • Object numbers of two condiments of first menu item are **41103** and **44502**. Object number of condiment of second menu item is **41103** • Overridden price for first menu item is **$10** • Menu levels to pick up first menu item is ***Main Menu Level - 2*** and ***Sub Menu Level - 3*** • “Open” discount percent for first menu item is **7%** • Reference text for first menu item is *“**Chef's favorite**”*
3	Combo Meal	*ppComboMeals*	• Combo Meal details Object number of combo meal is **10** o Object number of main item is **110003** o Object numbers of side items are **41103** and **44502** o Object number of drink is **110004**
4	Service Charge	*pServiceChg*	“Open” service charge is **6%**

S. No	Type of Data	Parameter Name	Sample Data
			Reference text is: "***6% service charge including tips***"
5	Subtotal Discount	*pSubTotalDiscount*	"Open" subtotal discount amount is **$5**
6	Revenue Center object number	*revenueCenterObjectNum*	**3016**
7	Order Type	*orderType*	**1** (i.e. Dine-in)
8	Employee Number	*employeeObjectNum*	**90001**
9	Total Response	*pTotalsResponse*	N/A (this parameter is to hold output)

The method *CalculateTransactionTotals* can be used in this situation. Given below is the signature of the method for quick reference.

Table 9-1 Calculate Transaction Totals Method Signature

```
void CalculateTransactionTotals
(
    string                                  vendorCode,
    ref ARRAY(SimphonyPosAPI_MenuItem)      ppMenuItems,
    ref ARRAY(SimphonyPosAPI_ComboMeal)     ppComboMeals,
    ref SimphonyPosAPI_SvcCharge            pServiceChg,
    ref SimphonyPosAPI_Discount             pSubTotalDiscount,
    int                                     revenueCenterObjectN
um,
    short                                   orderType,
    int                                     employeeObjectNum,
    ref SimphonyPosAPI_TotalsResponse       pTotalsResponse
)
```

The following code snippet demonstrates how data for input parameters of the *CalculateTransactionTotals* method can be constructed and used to invoke the method.

```
SimphonyPosAPIWebSoapClient mTSApi = new SimphonyPosAPIWebSoapClient();

string vendorCode = "lzsroioq";
int revenueCenterObjectNum = 3016;
int employeeObjectNum = 90001;
short orderType = 1; // e.g. Dine-in

public void InvokeCalculateTransactionTotalMethod()
{
  SimphonyPosApi_MenuItem[] ppMenuItems = GetMenuItemList();
  SimphonyPosApi_ComboMeal[] ppComboMeals = GetComboMealList();
  SimphonyPosApi_SvcCharge pSvcCharge = GetServiceCharge(true);
  SimphonyPosApi_Discount pSubtotalDiscount = GetSubtotalDiscount(true);
  SimphonyPosApi_TotalsResponse pTotalsResponse = new SimphonyPosApi_TotalsResponse();

  mTSApi.CalculateTransactionTotals(vendorCode, ref ppMenuItems, ref ppComboMeals,
    ref pSvcCharge,ref pSubtotalDiscount, revenueCenterObjectNum, orderType,
    employeeObjectNum, ref pTotalsResponse);

  if (pTotalsResponse.OperationalResult.Success)
  {
    Console.WriteLine("Calculate Transaction Total succeeded...");

    Console.WriteLine("Total Due: " + pTotalsResponse.TotalsTotalDue);
    Console.WriteLine("Subtotal: " + pTotalsResponse.TotalsSubTotal);
    Console.WriteLine("Total Auto Service Charge: " +
      pTotalsResponse.TotalsAutoSvcChgTotals);
    Console.WriteLine("Total Service Charge (Manual): " +
      pTotalsResponse.TotalsOtherTotals);
    Console.WriteLine("Total Tax: " + pTotalsResponse.TotalsTaxTotals);
  }
  else
  {
    Console.WriteLine(String.Format(
      "Calculate Transaction Total failed. Error Code: {0}, Error Message: {1}",
      pTotalsResponse.OperationalResult.ErrorCode,
      pTotalsResponse.OperationalResult.ErrorMessage));
  }
}
```

The following sections explain constructing data for each input parameter with the sample data provided above.

Vendor Code

The Vendor Code or Vendor Activation Code is a string value that uniquely identifies a vendor for Transaction Services. This was introduced to validate the license of TS API. The Vendor Activation Code should be configured in the EMC along the following location for TS API to work properly:

1. Click the **Enterprise** level.
2. Click the **Setup** tab, under the **Parameters** section, click **Enterprise Parameters**, and then click the **Licensing** tab.

This requirement is removed from Simphony from version 2.7 MR3 and later. However, this parameter is still out there even in the latest TS API for backward compatibility. The client application that integrates with TS API of version 2.7 MR3 or later can pass an empty value to this parameter while prior versions should pass a valid Vendor Code that was distributed to them.

See Figure 5-2 Transaction Services Vendor License Configuration for the dialog to configure vendor codes for the TS API.

Parameter Signature

```
String      vendorCode
```

e.g.,

```
string vendorCode = "yzsroioq";
```

Menu Items and Condiments

This parameter represents the list of menu items with required condiments for those menu items. Each menu item and condiment is identified by an Object Number. Menu items and condiments are configured at enterprise or property or revenue center level in the EMC within the following module:

1. Click the **Enterprise** level.
2. Click the **Configuration** tab, under the **Menu Items** section, and then click **Menu Item Maintenance**.

Parameter Signature

```
ref                    ARRAY(SimphonyPosAPI_MenuItem)
    ppMenuItems
```

Table 9-2 SimphonyPosAPI_MenuItem Signature

```
public class SimphonyPosApi_MenuItem
{
    public SimphonyPosApi_MenuItemDefinition[] Condiments;
    public SimphonyPosApi_MenuItemDefinition MenuItem;
}
```

Menu Items are the core foundation of all POS transactions. Everything "ordered" or "rung in" for the systems is a Menu Item. In restaurant revenue centers, it is obvious that drinks and entrees, etc., are Menu Items. Perhaps less obvious, in retail revenue centers, shirts and pants (etc.) are also considered Menu Items. Therefore, in Simphony, it can be said that any item being sold is a Menu Item.

The code snippet given below demonstrates how to construct input data for menu item list parameter. This code adds two menu items and respective condiments to the list as required. Also, this applies a discount (i.e., item level) to the first menu item.

```
private SimphonyPosApi_MenuItem[] GetMenuItemList()
{
   List<SimphonyPosApi_MenuItem> menuItemList = new List<SimphonyPosApi_MenuItem>();

   SimphonyPosApi_MenuItem firstMenuItem = new SimphonyPosApi_MenuItem();
   firstMenuItem.MenuItem = GetFirstMenuItem();
   firstMenuItem.MenuItem.ItemDiscount = GetItemDiscount(true);
   firstMenuItem.Condiments = new SimphonyPosApi_MenuItemDefinition[2];
   firstMenuItem.Condiments[0] = GetFirstCondimentItem();
   firstMenuItem.Condiments[1] = GetSecondCondimentItem();
   menuItemList.Add(firstMenuItem);

   SimphonyPosApi_MenuItem secondMenuItem = new SimphonyPosApi_MenuItem();
   secondMenuItem.MenuItem = GetSecondMenuItem();
   secondMenuItem.Condiments = new SimphonyPosApi_MenuItemDefinition[1];
   secondMenuItem.Condiments[0] = GetFirstCondimentItem();
   menuItemList.Add(secondMenuItem);

   return menuItemList.ToArray();
}
```

In addition, the following code demonstrates how to construct two menu items with given input data. Each menu item and condiment is identified by a unique identifier called Menu Item Object Number. 110003 and 110004 are the Object Numbers of first and second menu items respectively. The price of first menu item in this example is overridden by $7. It's possible that any menu item or condiment is configured to have more than one definition with different price record for each definition. When no menu levels are specified it picks up the first definition by default.

Here in this example, both main and sub menu levels are specified for the first menu item in order to pick up a particular definition instead of the default. A reference text is also added to the first menu item for reference purposes.

```
private SimphonyPosApi_MenuItemDefinition GetFirstMenuItem()
{
   SimphonyPosApi_MenuItemDefinition menuItemDefn = new
      SimphonyPosApi_MenuItemDefinition();
   menuItemDefn.MiObjectNum = 110003;
   menuItemDefn.MiOverridePrice = "7";
   menuItemDefn.MiMenuLevel = 2;
   menuItemDefn.MiSubLevel = 3;
   menuItemDefn.MiReference = "Chef's favorite";
   return menuItemDefn;
}

private SimphonyPosApi_MenuItemDefinition GetSecondMenuItem()
{
   int menuItemObjectNum = 110004;
   SimphonyPosApi_MenuItemDefinition menuItemDefn = new
     SimphonyPosApi_MenuItemDefinition();
   menuItemDefn.MiObjectNum = menuItemObjectNum;
   return menuItemDefn;
}
```

The following code demonstrates how to construct an object for two condiment menu items with the provided input data.

```
private SimphonyPosApi_MenuItemDefinition GetFirstCondimentItem()
{
   int menuItemObjectNum = 41103;
   SimphonyPosApi_MenuItemDefinition menuItemDefn = new
      SimphonyPosApi_MenuItemDefinition();
   menuItemDefn.MiObjectNum = menuItemObjectNum;
   return menuItemDefn;
}

private SimphonyPosApi_MenuItemDefinition GetSecondCondimentItem()
{
   int menuItemObjectNum = 44502;
   SimphonyPosApi_MenuItemDefinition menuItemDefn = new
     SimphonyPosApi_MenuItemDefinition();
   menuItemDefn.MiObjectNum = menuItemObjectNum;
   return menuItemDefn;
}
```

The following code demonstrates how to construct a discount object for given input data. Each discount configured in EMC is identified by a unique identifier called Discount Object Number. In case of “preset” discount, amount or percentage of discount will be taken from value that’s configured in EMC. However, in case of an “open” discount, amount or percentage of discount should be supplied by the caller. The property *DiscAmountOrPercent* could be an amount or percent based on how given discount is configured using EMC. This example demonstrates that 10 is the Discount Object Number of an open discount and the caller is applying a 7% discount to a menu item. All manual discounts should be added to applicable menu item explicitly in this way while API takes care of applying automatic discount implicitly by itself.

```
private SimphonyPosApi_Discount GetItemDiscount(bool isOpenDiscount)
{
   SimphonyPosApi_Discount discount = new SimphonyPosApi_Discount();
   discount.DiscObjectNum = 10;

   // percentage or amount based on how it's configured in EMC
   if (isOpenDiscount)
      discount.DiscAmountOrPercent = "7";

   discount.DiscReference = "Mother’s day discount";
   return discount;
}
```

Combo Meal

Combo Meal is a combination meal such as a burger with fries and a drink or pancake with ham and coffee which are usually offered together at a lower price than they would cost individually. Combo Meal should be configured in the EMC before it can be rung up via TS API.

Combo Meals can be found in EMC under **Property** level | **Configuration** tab | **Sales** | **Combo Meals** module.

To configure Combo Meal:

1. Create a Menu Item
2. Create a Combo Meal Menu Item class
3. Add menu item to Combo Meal class
4. Create a Combo Meal Group
5. Add Main, Drink and Side Item

Parameter Signature

ref ARRAY(SimphonyPosAPI_ComboMeal)	*ppComboMeals*

SimphonyPosApi_ComboMeal Signature

```
public class SimphonyPosApi_ComboMeal
{
        public SimphonyPosApi_MenuItem ComboMealMainItem;
        public SimphonyPosApi_MenuItem ComboMealMenuItem;
        public int ComboMealObjectNum;
        public SimphonyPosApi_MenuItem[] SideItems;
}
```

The code snippet provided below, demonstrates how to construct combo meal object for given input data. This example, adds a main menu item, two side items and a drink to form a combo meal. Each combo meal is identified by a unique identifier called Combo Meal Object Number. Login into the EMC to get object numbers of combo meal and related items.

```
private SimphonyPosApi_ComboMeal[] GetComboMealList()
{

SimphonyPosApi_ComboMeal[] comboMeal = new SimphonyPosApi_ComboMeal
[1];

   SimphonyPosApi_ComboMeal comboMeal1 = new SimphonyPosApi_ComboM
eal();
   comboMeal1.ComboMealObjectNum = 10;

   // Add a Main item
   SimphonyPosApi_MenuItem mainItem = new SimphonyPosApi_MenuItem();
   mainItem.MenuItem = new SimphonyPosApi_MenuItemDefinition();
   mainItem.MenuItem.MiObjectNum = 110003;
   comboMeal1.ComboMealMainItem = mainItem;

   // Add 2 Side items
   SimphonyPosApi_MenuItem[] sideItemList = new SimphonyPosApi_MenuItem[2
];
   SimphonyPosApi_MenuItem firstSideItem = new SimphonyPosApi_MenuItem();
   firstSideItem.MenuItem = new SimphonyPosApi_MenuItemDefinition();
   firstSideItem.MenuItem.MiObjectNum = 41103;
   sideItemList[0] = firstSideItem;
   SimphonyPosApi_MenuItem secondSideItem = new SimphonyPosApi_MenuIte
m();
   secondSideItem.MenuItem = new SimphonyPosApi_MenuItemDefinition();
   secondSideItem.MenuItem.MiObjectNum = 44502;
   sideItemList[1] = secondSideItem;
   comboMeal1.SideItems = sideItemList;

   // Add a Drink
   SimphonyPosApi_MenuItem menuItem = new SimphonyPosApi_MenuItem();
   menuItem.MenuItem = new SimphonyPosApi_MenuItemDefinition();
   menuItem.MenuItem.MiObjectNum = 110004;
   comboMeal1.ComboMealMenuItem = menuItem;

   comboMeal[0] = comboMeal1;
   return comboMeal;
}
```

Service Charge

A Service Charge is an amount that is added to a sales transaction for a service rendered. There are two ways to add service charge to the transaction, which are

- Automatic Service Charge
- Manual Service Charge

An Automatic Service Charge is a service charge that applies to all the items in Menu Item Class with the 'Add to Automatic Service Charge Itemizer' option bit enabled, without being entered by operator intervention.

However, a manual service charge that needs applied, should be added to input parameter of TS API.

Service Charge can be configured in the EMC here:

1. Click either the **Enterprise** or **Property** level,
2. Click the **Configuration** tab, under **Sales**, and then click **Service Charges**.

Parameter Signature

```
ref SimphonyPosAPI_SvcCharge      pServiceChg
```

SimphonyPosApi_SvcCharge Signature

```
public class SimphonyPosApi_SvcCharge
{
        public string SvcChgAmountOrPercent;
        public int SvcChgObjectNum;
        public string SvcChgReference;
}
```

The following code demonstrates how to construct a Service Charge object for given input data. Each service charge that's configured in EMC is identified by a unique identifier called Service Charge Object Number. In case of "preset" service charge, amount or percentage of service charge will be taken from value that's configured in EMC. However, in case of an "open" service charge, amount or percentage of service charge should be supplied by the caller. The field *SvcChgAmountOrPercent* could be an amount or percent based on how it's is configured in EMC. This example demonstrates that 12 is the Service Charge Object Number of an "open" service charge and the caller is applying a 6% service charge on the guest check. Any manual service charge should be added to guest check explicitly in this way while API takes care of applying automatic service charge implicitly by itself.

```
private SimphonyPosApi_SvcCharge GetServiceCharge(bool isOpenServiceCharg
e)
{

SimphonyPosApi_SvcCharge serviceCharge = new SimphonyPosApi_SvcCharge()
;
   serviceCharge.SvcChgObjectNum = 12;

   if (isOpenServiceCharge)
     serviceCharge.SvcChgAmountOrPercent = "6"

   serviceCharge.SvcChgReference = "6% service charge including tips";

   return serviceCharge;
}
```

Subtotal Discount

A discount reduces the price of an item or items on a check. Discounts are generally used for promotional purposes (a coupon for a free dessert) or for customer satisfaction. Discounts can be programmed as "Subtotal Discounts" or "Item Discounts". An "Item Discount" is used to discount a single item, whereas "Subtotal Discounts" apply to one or more items on the check based on the configuration of the discount in EMC.

By default, all discounts are Subtotal Discounts, which means that the discount applies to all items on a check that belong to a Menu Item Group or Itemizer Group affected by the discount. A discount is a subtotal discount when the **This is an Item Discount** option bit is disabled in the EMC.

There are three different types of "activation" for discounts:

1. **Manual**

   A manual discount is applied by the user to a check. This type of discount is a "traditional" discount.

2. **Automatic**

   An automatic discount is applied by the discount engine when certain criteria of the transaction are met. As a user rings items, the workstation continually looks for items that will ‘trigger’ a automatic discount, and then the ‘award’ amount is applied to the check if necessary.

3. **Coupon**

   An automatic coupon discount is an automatic discount with one difference: the user must first apply the discount to the check, letting the discount engine know that the discount is available for the check.

Parameter Signature

```
ref SimphonyPosAPI_Discount        pSubTotalDiscount
```

SimphonyPosApi_Discount Signature

```
public class SimphonyPosApi_Discount
{
        public string DiscAmountOrPercent;
        public int DiscObjectNum;
        public string DiscReference;
}
```

The code snippet given below demonstrates how to construct a Sub Total discount object with given input data. Each discount configured in EMC is identified by a unique identifier called Discount Object Number. In case of “preset” discount, amount or percentage of discount will be taken from value that’s configured in EMC itself. However, in case of an “open” discount, amount or percentage of discount should be supplied by the caller. The property *DiscAmountOrPercent* could be an amount or percent based on how given discount is configured in EMC. This example demonstrates that 11 is the Discount Object Number of an open Sub Total discount and the caller is applying a 5% discount to guest check for all triggered menu item group. All manual Sub Total discounts should be added to guest check explicitly in this way while API takes care of applying automatic Sub Total discounts implicitly by itself.

```
private SimphonyPosApi_Discount GetSubtotalDiscount(bool isOpenDiscount)
{

SimphonyPosApi_Discount subTotalDiscount = new SimphonyPosApi_Discount();
    subTotalDiscount.DiscObjectNum = 11;

    // percentage or amount based on how it's configured in EMC
    if (isOpenDiscount)
        subTotalDiscount.DiscAmountOrPercent = "5";

    subTotalDiscount.DiscReference = "Weekend discount";
    return subTotalDiscount;
}
```

Revenue Center Object Number

Revenue Centers are distinctly identifiable department, division, or unit of a firm that generates revenue through sale of goods and/or services. For example, rooms department and food-and-beverages department of a hotel are its revenue centers. Each revenue center of a property is identified by a unique identifier called Revenue Center Object Number.

Revenue Center information can be found in the EMC.

1. Click the **Property** and click the **Setup** tab.
2. Under the **Property Configuration** header click the **RVC Configuration** module. 3016 is the Object Number of the revenue center in this example.

Parameter Signature

```
int   revenueCenterObjectNum
e.g.,
int revenueCenterObjectNum = 3016;
```

Order Type ID

An Order Type is a configurable menu item sales category; Order Types can be used to control tax rates that are active during a transaction. ‘Dine-out’ and "Dine in" are commonly-programmed Order Types.

Parameter Signature

short *orderType*

e.g., short orderType = 1; // e.g. Dine-in

Employee Object Number

Each employee in a property is identified by a unique identifier called Employee Object Number. This object number of an employee should be passed to the API for this operation in order to associate an employee for given transaction.

Employee details including their object number can be found in the EMC here:

1. Click the **Property,** and then click the **Configuration** tab
2. Under the **Personal** heading, click **Employee Maintenance**.

In this example, **90001** is the object number of an employee called David and is assigned revenue center 3016.

Parameter Signature

int *employeeObjectNum*

e.g., int employeeObjectNum = 90001;

API Response

The response of the method call can be found in *pTotalsResponse* parameter. The value of *OperationalResult* property of pTotalsResponse object indicates whether or not the operation has succeeded. In case operation succeeded then data for subtotal, total due, tax amount, auto & manual service charge amounts can be found in respective properties of ***pTotalsResponse.*** In case of failure, *OperationalResult.ErrorCode* property will hold the error code while detailed error message can be found in *OperationalResult.ErrorMessage* property.

Parameter Signature

```
ref SimphonyPosAPI_TotalsResponse        pTotalsResponse
```

SimphonyPosApi_TotalsResponse Signature

```
public class SimphonyPosApi_TotalsResponse
{
   public SimphonyPosApi_OperationalResult OperationalResult;
   public string TotalsTotalDue
   public string TotalsSubTotal
   public string TotalsTaxTotals
   public string TotalsAutoSvcChgTotals
   public string TotalsOtherTotals
}
```

Create a Guest Check

Once total amount of a transaction is calculated and reviewed by the user, he may want to post that transaction and create a Guest Check in Simphony POS database by providing tender/payment details. The tender can be of any type like cash, credit/debit/SVC card. The method *PosTransactionEx* can be used for such purpose.

TS API supports auto-fire feature on guest check. Auto-fire means that the guest check will be immediately created in the system however it'll fire only when the time that's mentioned to fire at the time of guest check creation is attained. This example demonstrates auto-fire feature by firing the guest check only after 12 hours from when guest check is posted. This auto-fire feature will be of use in hotels where a guest wants to order food for his or her dinner in the morning itself. In such cases, the guest check will get created in the system as soon as the transaction is posted but it'll fire only at the specified time in the evening so that chef can prepare ordered food for dinner for that specific customer.

The input data for menu items, combo meals, discount, service charge etc. given in the previous section for calculate totals example is taken for this method as well. However, this method needs data for following additional parameter too.

- Tender/payment details

The Post Transaction method posts current transaction to Simphony POS database to create a guest check. For example, *CalculateTransactionTotals*. This method calculates the transaction totals too. If payment/tender media is of type "Service Total" then the system will create the guest check and keep it in the open state. However, when a tender media with appropriate payment details (cash, credit/debit, SVC) are passed for full payment then the check will be created and taken to "closed" state at the end of the call. Any tender with partial payment will still have the created check in open state only. Another tender with payment for balance amount can be added later to that check using a method called *AddToExistingCheckEx* to close that check.

Post Transaction Method Signature

```
void PostTransactionEx
(
    String                                  vendorCode,
    ref SimphonyPosAPI_GuestCheck           pGuestCheck,
    ref ARRAY(SimphonyPosAPI_MenuItem)      ppMenuItems,
    ref ARRAY(SimphonyPosAPI_ComboMeal)     ppComboMeals,
    ref SimphonyPosAPI_SvcCharge            ServiceChg,
    ref SimphonyPosAPI_Discount             pSubTotalDiscount,
    ref SimphonyPosAPI_TmedDetailItemEx     pTmedDetailEx,
    ref SimphonyPosAPI_TotalsResponse       pTotalsResponse,
    ref ARRAY(string)
    ppCheckPrintLines,
    ref ARRAY(string)
    ppVoucherOutput
)
```

The following code snippet demonstrates how data for input parameters of *PostTransactionEx* method can be constructed and used to invoke the method. This example demonstrates creating a guest check with the same input data that are mentioned in the previous section for calculated totals.

```
SimphonyPosAPIWebSoapClient mTSApi = new SimphonyPosAPIWebSoapClient();

string vendorCode = "lzsroioq";
int revenueCenterObjectNum = 3016;
int employeeObjectNum = 90001;
short orderType = 1; // e.g. Dine-in

public void InvokePostTransactionEx()
{
   bool isAutoFireCheck = true;

   SimphonyPosApi_GuestCheck guestCheck = new SimphonyPosApi_GuestCheck();
   guestCheck.CheckOrderType = orderType;
   guestCheck.CheckEmployeeObjectNum = employeeObjectNum;
   guestCheck.CheckRevenueCenterID = revenueCenterObjectNum;

   // Optional parameters
   guestCheck.CheckGuestCount = 2; // Number of guests
   guestCheck.CheckTableObjectNum = 5; // Dining table number

   if (isAutoFireCheck)
   {
      // 0x10 is the status bit for auto fire (a.k.a. future order)
      // Note: 0x1 - Rush Order, 0x2 - VIP Order, 0x10 - Auto fire Order
      guestCheck.CheckStatusBits |= 0x10;
      // Check fires after 12 hours
      guestCheck.CheckDateToFire = DateTime.Now.AddHours(1);
   }

   string[] ppCheckPrintLines = new string[] { "" }; // Output parameter
```

```
    string[] ppVoucherOutput = new string[] { "" }; // Output parameter

    SimphonyPosApi_MenuItem[] ppMenuItems = GetMenuItemList();
    SimphonyPosApi_ComboMeal[] ppComboMeals = GetComboMealList();
    SimphonyPosApi_SvcCharge pSvcCharge = GetServiceCharge(true);
    SimphonyPosApi_Discount pSubtotalDiscount = GetSubtotalDiscount(true);
    SimphonyPosApi_TmedDetailItemEx tenderMedia = GetTenderMedia(
       TenderMediaType.CreditCard);
    SimphonyPosApi_TotalsResponse pTotalsResponse = new
       SimphonyPosApi_TotalsResponse();

    mTSApi.PostTransactionEx(vendorCode, ref guestCheck, ref ppMenuItems,
       ref ppComboMeals, ref pSvcCharge, ref pSubtotalDiscount, ref tenderMedia,
       ref pTotalsResponse, ref ppCheckPrintLines, ref ppVoucherOutput);

    if (guestCheck.OperationalResult != null && guestCheck.OperationalResult.Success)
    {
      Console.WriteLine("Post Transaction operation has succeeded...");

      Console.WriteLine("Guest Check ID: " + guestCheck.CheckID);
      Console.WriteLine("Guest Check Number: " + guestCheck.CheckNum);
      Console.WriteLine("Guest Check Sequence Number: " + guestCheck.CheckSeq);

      Console.WriteLine("Total Due: " + pTotalsResponse.TotalsTotalDue);
      Console.WriteLine("Subtotal: " + pTotalsResponse.TotalsSubTotal);
      Console.WriteLine("Total Auto Service Charge: " +
          pTotalsResponse.TotalsAutoSvcChgTotals);
      Console.WriteLine("Total Service Charge (Manual): " +
          pTotalsResponse.TotalsOtherTotals);
      Console.WriteLine("Total Tax: " + pTotalsResponse.TotalsTaxTotals);
    }
    else
    {
       Console.WriteLine(String.Format("Post Transaction operation has failed.
          Error Code: {0}, Error Message: {1}",
          pTotalsResponse.OperationalResult.ErrorCode,
          pTotalsResponse.OperationalResult.ErrorMessage));
    }
 }
```

The following sections explain parameters that are not explained in Calculate Totals Transaction method in the previous section. Refer to the previous section for all other parameters, as they are all already explained there.

Guest Check

For post transaction operations, the caller of the method is expected to pass data for the following mandatory properties of the *guestCheck* parameter.

- CheckEmployeeObjectNum
- CheckRevenueCenterID
- CheckOrderType

The following properties are optional

- CheckGuestCount
- CheckTableObjectNum
- CheckStatusBits
- CheckDateToFire

The following properties are populated by the API when an operation succeeds. In case of failure, only the *OperationalResult* populates to hold data for an error code and error message.

- CheckID
- CheckNum
- CheckSeq
- OperationalResult
- PCheckInfoLines
- PPrintJobIds

Check Number and Check Sequence Number are used to identify the created guest check uniquely, and they can be used in the future for updating that specific guest check. For example, this method can create a guest check with a partial payment and then another method called *AddToExistingCheckEx* can be invoked later to add another tender for the balance due amount to the same check by specifying Check Number and Check Sequence Number of that check. An example of the *AddToExistingCheckEx* method is provided in the next section.

Tender / Payment

A Tender Media is a form of payment or a service total used on a Guest Check. Each tender media is identified by a unique identifier called Tender Media Object Number. Tender media should have been configured in EMC before it can be used in TS API.

Code snippet given below demonstrates how tender media with sample data can be constructed based on tender media type. This code indicates the data being expected by payment driver while creating guest check.

```
private SimphonyPosApi_TmedDetailItemEx GetTenderMedia(TenderMediaType tmType)
{
  SimphonyPosApi_TmedDetailItemEx tenderMedia = new SimphonyPosApi_TmedDetailItemEx();
  SimphonyPosApi_EPayment ePayment = new SimphonyPosApi_EPayment();
  switch (tmType)
  {
    case TenderMediaType.Cash:
    {
        tenderMedia.TmedObjectNum = 12;
        tenderMedia.TmedPartialPayment = "30"; // tendered amount excluding tip

        // indicates cash payment
        ePayment.PaymentCommand = EPaymentDirective.NO_E_PAYMENT;
        ePayment.TipAmount = "5"; // tendered amount for tip

        tenderMedia.TmedEPayment = ePayment;
        tenderMedia.TmedReference = "Total  amount tendered (including tip) is $35";
        break;
    }
    case TenderMediaType.CreditCard:
    {
        tenderMedia.TmedObjectNum = 30;
        ePayment.PaymentCommand = EPaymentDirective.CREDIT_AUTHORIZE_AND_PAY;

        // Track2 has most of the data required by the payment driver
        ePayment.Track2Data = "7777666655554444=00010002000370783149";
        ePayment.BaseAmount = "30"; // Base amount excluding tip
        ePayment.TipAmount = "5"; // Amount for tip
        ePayment.CashBackAmount = "15";

        tenderMedia.TmedReference =
           "Total amount to be deducted from CREDIT CARD is $50";
        break;
    }
    case TenderMediaType.DebitCard:
    {
        tenderMedia.TmedObjectNum = 31;
        ePayment.PaymentCommand = EPaymentDirective.DEBIT_AUTHORIZE_AND_PAY;

        // Track2 has most of the data required by the payment driver
        ePayment.Track2Data = "8888777766665555=00020003000470783249";
        ePayment.BaseAmount = "30"; // Base amount excluding tip
        ePayment.TipAmount = "5"; // Amount for tip
        ePayment.CashBackAmount = "15";

        tenderMedia.TmedReference =
           "Total amount to be deducted from DEBIT CARD is $50";
```

```
            break;
        }
        case TenderMediaType.StoredValueCard:
        {
            tenderMedia.TmedObjectNum = 35;
            ePayment.PaymentCommand = EPaymentDirective.STORED_VALUE_CARD_REDEEM;

            // Track2 has most of the data required by the payment driver
            ePayment.Track2Data = "9999888877776666=10020003000470783249";
            ePayment.BaseAmount = "30"; // Base amount excluding tip
            ePayment.TipAmount = "5"; // Amount for tip
            ePayment.CashBackAmount = "15";

            tenderMedia.TmedReference =
                "Total amount to be deducted from SVC CARD is $50";
        }
        case TenderMediaType.ServiceTotal:
        {
            tenderMedia.TmedObjectNum = 49;
            tenderMedia.TmedReference = "Payment is not done yet";
            break;
        }
      }
    return tenderMedia;
}
```

See the Calculate Totals of a Transaction section of this document for more details on other input parameters of this method.

API Response

If post transaction operation has succeeded then the output details like Check ID, Check Number and Check Sequence Number of the created check is populated in the appropriate fields of *pGuestCheck* parameter. Also, it populates print receipt of guest check in *ppCheckPrintLines* property of *pGuestCheck* while it populates credit voucher of current transaction in *ppVoucherOutput* property of same parameter. If this method encounters any problem such as a payment failure, then it will not create a guest check and it throws an appropriate exception to the caller. Also, the *OperationalResult* property of *pTotalsResponse* parameter holds the appropriate error code and message in that case.

Add an Item to an Open Guest Check

Once user has posted a transaction/check to Simphony POS database, sometimes they may needs to update it. For example, where a transaction was posted to create a guest check with Service Total as the tender media. That is, the payment has not been applied yet for that transaction. In that case, they may want to add a tender to that guest check later, in order to make payment for due amount. In such cases, *AddToExistingCheckEx* method can be used to achieve it.

The following code snippet demonstrates adding a tender media to an existing open Guest Check to make payment so that the check can be closed. Like a tender media, any other items like menu item, combo meal, service charge or discount can also be added to an open guest check.

```
SimphonyPosAPIWebSoapClient mTSApi = new SimphonyPosAPIWebSoapClient();
string vendorCode = "lzsroioq"; // This can be empty from 2.7MR3 onwards

public void AddTenderToExistingGuestCheck()
{
   SimphonyPosApi_GuestCheck guestCheck = new SimphonyPosApi_GuestCheck();
   guestCheck.CheckEmployeeObjectNum = 90001;
   guestCheck.CheckRevenueCenterID = 3016;
   guestCheck.CheckNum = 1043;// Check Number of Guest Check to which tender needs to applied
   guestCheck.CheckSeq = 534418293; // Sequence Number of Guest Check

   string[] ppCheckPrintLines = new string[] { "" };
   string[] ppVoucherOutput = new string[] { "" };

   SimphonyPosApi_MenuItem[] ppMenuItems = new SimphonyPosApi_MenuItem[0];
   SimphonyPosApi_ComboMeal[] ppComboMeals = new SimphonyPosApi_ComboMeal[0];
   SimphonyPosApi_SvcCharge pSvcCharge = new SimphonyPosApi_SvcCharge();
   SimphonyPosApi_Discount pSubtotalDiscount = new SimphonyPosApi_Discount();
   SimphonyPosApi_TmedDetailItemEx tenderMedia =
         GetTenderMedia(TenderMediaType.DebitCard);
   SimphonyPosApi_TotalsResponse pTotalsResponse = new SimphonyPosApi_TotalsResponse();

   mTSApi.AddToExistingCheckEx(vendorCode, ref guestCheck, ref ppMenuItems,
      ref ppComboMeals, ref pSvcCharge, ref pSubtotalDiscount, ref tenderMedia,
      ref pTotalsResponse, ref ppCheckPrintLines, ref ppVoucherOutput);

   if (pTotalsResponse.OperationalResult.Success)
   {
      Console.WriteLine("Add To Existing Check succeeded...");

      Console.WriteLine("Total Due: " + pTotalsResponse.TotalsTotalDue);
      Console.WriteLine("Subtotal: " + pTotalsResponse.TotalsSubTotal);
      Console.WriteLine("Total Auto Service Charge: " +
        pTotalsResponse.TotalsAutoSvcChgTotals);
      Console.WriteLine("Total Service Charge (Manual): " +
        pTotalsResponse.TotalsOtherTotals);
      Console.WriteLine("Total Tax: " + pTotalsResponse.TotalsTaxTotals);
   }
   else
   {
       Console.WriteLine(String.Format("Add To Existing Check failed.
          Error Code: {0}, Error Message: {1}",
          pTotalsResponse.OperationalResult.ErrorCode,
```

```
        pTotalsResponse.OperationalResult.ErrorMessage));
    }
}
```

Like tender media provided in this example, one or more items like menu, combo meal, service charge and subtotal discount can be added to an existing open guest check using this method. However, this method can operate only on open Guest Checks and it'll thrown exception if caller tries to add any item a closed check.

For details on all input parameters of this method, see the Calculate Totals of a Transaction and Create a Guest Check sections of this document.

API Response

Adding any item to an existing open Guest Check will close the original check and will create a new check with all items internally. When this method is invoked, pGuestCheck parameter will be interrogated and fields like Check ID, Check Number and Check Sequence Number will be updated with details of new guest check.

Void All Items of an Open Guest Check

Sometimes, user may needs to cancel an already posted transaction/check due to incorrect order entry or other reasons. In such cases, VoidTransaction method can be used to achieve it. This method can operate on only open Guest Check. It voids each and every item in the check and keeps the check still open. In order to identify the check, the caller is expected to pass both Check Number and Check Sequence Number to this method.

VoidTransaction Method Signature

```
void  VoidTransaction
(
        string                                  vendorCode,
        ref SimphonyPosAPI_GuestCheck
                pGuestCheck
)
```

The following code snippet demonstrates invoking the *VoidTransaction* method with sample input data.

```
SimphonyPosAPIWebSoapClient mTSApi = new SimphonyPosAPIWebSoapClient();
string vendorCode = "lzsroioq"; // This can be empty from 2.7MR3 onwards

public void InvokeVoidTransaction()
{
  SimphonyPosApi_GuestCheck guestCheck = new SimphonyPosApi_GuestCheck();
  guestCheck.CheckNum = 1008;  // Check Number of the Guest Check
  guestCheck.CheckSeq = 315015863; // Sequence Number of the Guest Check

  mTSApi.VoidTransaction(vendorCode, ref guestCheck);

  if (guestCheck.OperationalResult.Success)
  {
     Console.WriteLine("Void Transaction succeeded...");
  }
  else
  {
     Console.WriteLine(String.Format("Void Transaction failed. Error Code: {0},
       Error Message: {1}",guestCheck.OperationalResult.ErrorCode,
       guestCheck.OperationalResult.ErrorMessage));
  }
}
```

API Response

If operation has succeeded then *OperationalResult.Success* field will hold “true”. It’ll hold “false” otherwise. Also, *OperationalResult.ErrorCode* will hold error code while *OperationalResult.ErrorMessage* holds reason for failure.

Get Status of a Print Job

When a transaction is posted to POS database using *PostTransactionEx* method, at the end of posting, it creates a print job to print the Guest Check. The ID of that print job is stored and returned to the caller via *PPrintJobIds* field of *SimphonyPosApi_GuestCheck* parameter. The job IDs are accumulated in *PPrintJobIds* parameter. So, the last job id present in *PPrintJobIds* array indicates the print job id of last Guest Check that was posted to Simphony POS database.

In case Guest Check did not print due to any reason then the status of corresponding print job can be retrieved using *CheckPrintJobStatus* method of TS API. This method accepts the ID of print job as one of the parameters and returns the status of that job.

The following code snippet demonstrates how to retrieve the status of a print job.

```
SimphonyPosAPIWebSoapClient mTSApi = new SimphonyPosAPIWebSoapClient();
string vendorCode = "lzsroioq"; // This can be empty from 2.7MR3 onwards

public void InvokeCheckPrintJobStatus()
{
   // Print Job Id corresponding to a Guest Check that didn't print
   int printJobId = 1052;
   SimphonyPrintApi_PrintJobStatus printJobStatus =
       new SimphonyPrintApi_PrintJobStatus();

   mTSApi.CheckPrintJobStatus(vendorCode, printJobId, ref printJobStatus);

   if (printJobStatus.OperationalResult.Success)
   {
      Console.WriteLine("Checking status of print job succeeded...");
      onsole.WriteLine("Print job status:" + printJobStatus.Status.ToString());
   }
   else
   {
      Console.WriteLine("Checking status of print job failed. Error Code: {0},
         Error Message: {1}", printJobStatus.OperationalResult.ErrorCode,
         printJobStatus.OperationalResult.ErrorMessage);
   }
}
```

API Response

If operation succeeds then *OperationalResult.Success* field will hold true. Also, *printJobStatus* will hold the status of queries print job. The enumerator given below has potential status of any print job.

```
public enum SimphonyPrintApi_Status
   {
      JobPending = 0,
      JobComplete = 1,
      JobAborted = 2,
      JobSentToBackup = 3,
      JobFailed = 4,
      JobNotFound = 5,
   }
```

Get Summary of All Open Guest Checks

At times, you may want to get a summary of all of the open Guest Checks from all of the revenue centers of the property.

GetOpenChecks Method Signature

```
void   GetOpenChecks
(
        string                              vendorCode,
        int                                 EmployeeObjectNum,
        ref SimphonyPosAPI_OpenChecks       openChecks
)
```

Code snippet provided below demonstrates retrieving a summary of all of the open Guest Checks created by a specific employee whose Employee Object Number is 90001. The caller has to pass 0 (i.e., zero) to Employee Object Number parameter if he/she wishes to fetch all open checks irrespective of the owner who created the check.

```
SimphonyPosAPIWebSoapClient mTSApi = new SimphonyPosAPIWebSoapClient();
string vendorCode = "lzsroioq"; // This can be empty from 2.7MR3 onwards
int employeeObjectNum = 90001; // Pass 0 to fetch all open checks irrespective of Check Owner

public void InvokeGetOpenChecks()
{
   SimphonyPosApi_OpenChecks openChecks = new SimphonyPosApi_OpenChecks();

   mTSApi.GetOpenChecks(vendorCode, employeeObjectNum, ref openChecks);

   if (openChecks.OperationalResult.Success)
   {
      Console.WriteLine("Get Open Check succeeded...");

      foreach (SimphonyPosApi_CheckSummary check in openChecks.CheckSummary)
      {
         Console.WriteLine("Check Number:" + check.CheckNum);
         Console.WriteLine("Check Sequence Number:" + check.CheckSeq);
         Console.WriteLine("Check Total Due:" + check.CheckTotalDue);
         // The field CheckRevenueCenterObjectNum returns RVC ID (not Object Number)
         Console.WriteLine("RVC ID:" + check.CheckRevenueCenterObjectNum);
      }
   }
   else
   {
      Console.WriteLine("Get Open Check failed. Error Code: {0},
         Error Message: {1}",
         openChecks.OperationalResult.ErrorCode,
         openChecks.OperationalResult.ErrorMessage);
   }
}
```

API Response

If operation succeeds then *OperationalResult.Success* field will hold “true” and the summary of open Guest Checks is populated in the *openChecks* parameter. Also, please

note that the field *CheckRevenueCenterObjectNum* of *SimphonyPosApi_CheckSummary* structure is mislabeled and it will return Revenue Center ID and not Revenue Center Object Number as the name suggests.

In case of failure, *OperationalResult.Success* field will hold "false" while the *OperationalResult.ErrorCode* holds error code and *OperationalResult.ErrorMessage* holds reason for failure.

Get Open Guest Checks with RVC Object Number

If the caller expects the field *CheckRevenueCenterObjectNum* of *SimphonyPosApi_CheckSummary* to hold the RVC Object Number (instead of RVC ID) then *GetOpenChecksEx* method can be used instead of *GetOpenChecks* (that's explained in the previous section). Because, *GetOpenChecks* populates *CheckRevenueCenterObjectNum* with the ID of the revenue center while *GetOpenChecksEx* populates the same field with the Object Number.

GetOpenChecksEx Method Signature

```
void   GetOpenChecksEx
(
       string                          vendorCode,
       int                             employeeObjectNum,
       ref SimphonyPosAPI_OpenChecks   openChecks
)
```

Code snippet given below demonstrates retrieving summary of all open Guest Checks created by a specific employee whose Employee Object Number is 90001. The caller has to pass 0 (i.e., zero) to Employee Object Number parameter if you wish to fetch all open checks irrespective of the owner who created the check. The property *CheckRevenueCenterObjectNum* of response object returns the object number of RVC instead of the ID. The Object number is different from the ID.

```
SimphonyPosAPIWebSoapClient mTSApi = new SimphonyPosAPIWebSoapClient();
string vendorCode = "lzsroioq"; // This can be empty from 2.7MR3 onwards
int employeeObjectNum = 90001; // Pass 0 to fetch all open checks irrespective of Check Owner

public void InvokeGetOpenChecks()
{
    SimphonyPosApi_OpenChecks openChecks = new SimphonyPosApi_OpenChecks();
    mTSApi.GetOpenChecksEx(vendorCode, employeeObjectNum, ref openChecks);
    if (openChecks.OperationalResult.Success)
    {
       Console.WriteLine("Get Open Check succeeded...");
       foreach (SimphonyPosApi_CheckSummary check in openChecks.CheckSummary)
       {
          Console.WriteLine("Check Number:" + check.CheckNum);
          Console.WriteLine("Check Sequence Number:" + check.CheckSeq);
          Console.WriteLine("Check Total Due:" + check.CheckTotalDue);
          Console.WriteLine("RVC Object Number:" + check.CheckRevenueCenterObjectNum);
      }
    }
    else
    {
       Console.WriteLine("Get Open Check failed. Error Code: {0}, Error Message: {1}",
          openChecks.OperationalResult.ErrorCode,
```

```
        openChecks.OperationalResult.ErrorMessage);
    }
}
```

API Response

If the operation succeeds then *OperationalResult.Success* field will hold “true” and summary of open Guest Checks will be populated in openChecks parameter. Also, please note that the field *CheckRevenueCenterObjectNum* of *SimphonyPosApi_CheckSummary* structure will return Revenue Center Object Number as the name implies.

In case of failure, *OperationalResult.Success* field will hold “false” while *OperationalResult.ErrorCode* holds error code and *OperationalResult.ErrorMessage* holds reason for failure.

Get Open Guest Checks from a specific RVC

At times, you may want to see a summary of all open Guest Checks from a specific revenue center of the property. In that case, the following method can be used.

GetOpenChecksByRVC Method Signature

```
void   GetOpenChecksByRVC
(
        string                                  vendorCode,
        int                                     EmployeeObjectNum,
        int                                     revenueCenterObjectNum,
        ref SimphonyPosAPI_OpenChecks           openChecks
)
```

The code snippet provided below, demonstrates retrieving a summary of all of the open Guest Checks from RVC #3016 that are created by a specific employee whose Employee Object Number is 90001. The caller has to pass 0 (i.e., zero) to Employee Object Number parameter if they wish to fetch all of the open checks irrespective of the owner (who created the check).

```
SimphonyPosAPIWebSoapClient mTSApi = new SimphonyPosAPIWebSoapClient();
string vendorCode = "lzsroioq"; // This can be empty from 2.7MR3 onwards
int employeeObjectNum = 90001; // Pass 0 to fetch all open checks irrespective of Check Owner
int revenueCenterObjectNum = 3016;

public void InvokeGetOpenChecksByRVC()
{
   SimphonyPosApi_OpenChecks openChecks = new SimphonyPosApi_OpenChecks();

   mTSApi.GetOpenChecksByRVC(vendorCode, employeeObjectNum, revenueCenterObjectNum,
       ref openChecks);

   if (openChecks.OperationalResult.Success)
   {
      Console.WriteLine("Get Open Check succeeded...");

     foreach (SimphonyPosApi_CheckSummary check in openChecks.CheckSummary)
     {
        Console.WriteLine("Check Number:" + check.CheckNum);
        Console.WriteLine("Check Sequence Number:" + check.CheckSeq);
        Console.WriteLine("Check Total Due:" + check.CheckTotalDue);
        // The field CheckRevenueCenterObjectNum returns RVC ID (not Object Number)
        Console.WriteLine("RVC ID:" + check.CheckRevenueCenterObjectNum);
     }
   }
   else
   {
      Console.WriteLine("Get Open Check failed. Error Code: {0}, Error Message: {1}",
         openChecks.OperationalResult.ErrorCode,
         openChecks.OperationalResult.ErrorMessage);
   }
}
```

API Response

If operation succeeds then *OperationalResult.Success* field will hold “true” and the summary of open Guest Checks is populated in the *openChecks* parameter. In case of failure, *OperationalResult.Success* field holds “false” while *OperationalResult.ErrorCode* holds error code and *OperationalResult.ErrorMessage* holds the reason for failure.

Get Printed Texts of a Guest Check

The print lines of an open Guest Check can be retrieved by specifying Check Number and few other required details. This is often required where an external printer is used for printing an open guest check. The method *GetPrintedCheck* can be used for this purpose. This method will just retrieve the print lines in the output parameter without actually printing the guest check on a printer. Note that this method works only on open Guest Checks.

GetPrintedCheck Method Signature

```
void GetPrintedCheck
(
        string                                  vendorCode,
        int                                     CheckSeq,
        int                                     EmplObjectnum,
        int                                     TmedObjectNum,
        ref SimphonyPosApi_CheckPrintResponse   ppCheckPrintLines
)
```

Code snippet given below demonstrates retrieving print lines of a Guest Check.

```
SimphonyPosAPIWebSoapClient mTSApi = new SimphonyPosAPIWebSoapClient();
string vendorCode = "lzsroioq"; // This can be empty from 2.7MR3 onwards
int employeeObjectNum = 90001; // for authentication

public void InvokeGetPrintedCheck()
{
   int checkNumber = 1052; // Check Number for which print lines required
   int tenderMediaObjNum = 49; // Tender Media Object Number of Service Total
   SimphonyPosApi_CheckPrintResponse checkPrintLines =
       new SimphonyPosApi_CheckPrintResponse();

   mTSApi.GetPrintedCheck(vendorCode, checkNumber, employeeObjectNum,
      tenderMediaObjNum, ref checkPrintLines);

   if (checkPrintLines.OperationalResult.Success)
   {
       Console.WriteLine("Get Printed Check succeeded...");

       Console.WriteLine("Printed check lines:");
       foreach (string printLine in checkPrintLines.CheckPrintLines)
       {
          Console.WriteLine(printLine);
       }
   }
   else
   {
       Console.WriteLine("Get Printed Check failed. Error Code: {0},
         Error Message: {1}", checkPrintLines.OperationalResult.ErrorCode,
         checkPrintLines.OperationalResult.ErrorMessage);
   }
}
```

API Response

If operation succeeds then *OperationalResult.Success* field will hold “true” and print lines of Guest Check will populate in *checkPrintLines* parameter. In case of failure, *OperationalResult.Success* field will hold “false” while *OperationalResult.ErrorCode* holds error code and *OperationalResult.ErrorMessage* holds reason for failure.

Get Configured Information

Some of the data that are configured in EMC can be retrieved from POS database via TS API.

Users can retrieve the configuration data such as menu item definition, family group, interfaces, menu item master, menu item price, major group, revenue center parameter, tender media, currency, employees, menu item class, serving periods, service charge, discounts, dining tables, order types, revenue centers, menu levels, language information and application version from Simphony POS database using GetConfigurationInfo method.

GetConfigurationInfo Method Signature

```
void  GetConfigurationInfo
(
        String                                  vendorCode,
        Int                                     employeeObjectNum,
        ARRAY(int)                              configurationInfoType,
        Int                                     revenueCenter,
        ref SimphonyPosAPI_ConfigInfoResponse   configInfoResponse
)
```

The following code snippet demonstrates retrieving configured data for menu item definition, menu item price, tender media, currency and service charge.

```
SimphonyPosAPIWebSoapClient mTSApi = new SimphonyPosAPIWebSoapClient();
string vendorCode = "lzsroioq"; // This can be empty from 2.7MR3 onwards
int employeeObjectNum = 90001; // for authentication
int revenueCenterObjectNum = 3016;

public void InvokeGetConfigurationInfo()
{
   int[] configurationInfoType = new int[] {
           (int)EConfigurationInfoType.MENUITEMDEFINITIONS,
           (int)EConfigurationInfoType.MENUITEMPRICE,
           (int)EConfigurationInfoType.TENDERMEDIA,
           (int)EConfigurationInfoType.CURRENCY,
           (int)EConfigurationInfoType.SERVICECHARGE };

   SimphonyPosApi_ConfigInfoResponse configInfoResponse = new
       SimphonyPosApi_ConfigInfoResponse();

   mTSApi.GetConfigurationInfo(vendorCode, employeeObjectNum, configurationInfoType,
       revenueCenterObjectNum, ref configInfoResponse);

   if (configInfoResponse.OperationalResult.Success)
   {
      Console.WriteLine("Get Configuration Info succeeded...");
      Console.WriteLine("Menu item definitions: " +
          configInfoResponse.MenuItemDefinitions);
      Console.WriteLine("Menu item price: " + configInfoResponse.MenuItemPrice);
      Console.WriteLine("Tender media: " + configInfoResponse.TenderMedia);
      Console.WriteLine("Currency: " + configInfoResponse.Currency);
      Console.WriteLine("Service Charge: " + configInfoResponse.ServiceCharge);
   }
   else
   {
      Console.WriteLine("Get Configuration Info failed. Error Code: {0},
         Error Message: {1}", configInfoResponse.OperationalResult.ErrorCode,
```

```
        configInfoResponse.OperationalResult.ErrorMessage);
    }
}
```

API Response

If the operation succeeds then *OperationalResult.Success* field will hold “true” and configured data for queried type can be found in appropriate fields of configInfoResponse object. In case of failure*, OperationalResult.Success* will hold “false” while *OperationalResult.ErrorCode* holds error code and *OperationalResult.ErrorMessage* holds reason for failure.

10 Simphony Platform Requirements

Simphony Software Version

This release of the Simphony POS API requires Simphony v2.7 or later.

Off-line Transaction Support

The Simphony Transaction Services can never be in an offline state. It does not have an offline feature. As it is hosted by either a ServiceHost or IIS, the lazy playback mechanism posts the checks to the Check and Posting Server (CAPS). If CAPS is offline, then checks won't be posted to the CAPS machine unless it is restarted again.

Printing Services

The API supports printing to Remote and Local Order Devices. When a check is opened thru the TS API and posted to the Simphony database, the Menu Items will print on the remote and local devices based on the default Workstation definition and the assigned Menu Item Print Class. There should be no difference between how an API check prints versus a check opened directly by the user on the POS devices. Local Guest Check printing is not supported at this time thru the API. The Print Controller service must be running for printing to work.

Calling Conventions

There are two types of parameters passed to the API: ref and non-ref parameters. All parameters are mandatory. However, if you do not wish to use one of the parameters, simply create the structure and set all of its members to zero.

For example, if a check does not contain a Subtotal Discount, then you would pass the address of this structure to the API - everything will be zero. To add a Discount, fill in the appropriate members of the Discount object.

11 Demo Client For TS API

Overview

The Demo Client is a Microsoft Windows application that was developed to demonstrate or test the features of the Transaction Services API. This application is distributed with the Simphony installation media. This application builds data for input parameters based on values provided by the user, and sends request to TS API and displays the response in the UI.

Application Path

The demo client application can be found in following folder of install media.

<*InstallMediaFolder*>\Install\Simphony2\Tools**PosAPIDemoClient**

Prerequisites

A workstation is configured using the EMC and is running the Service Host application to host the Transaction Services web service. Navigate to the URL of TS web service using a web browser to see if TS web service is up and running.

Initial Setup

Follow the steps provided below to configure and run the TS demo client application.

1. Copy the *PosAPIDemoClient* folder from the installation media to a local folder.
2. Open the configuration file named **POSAPI_WebClient.exe.config** with a text editor such as Microsoft Notepad.
3. Modify the "value" key with correct URL of the TS web service, such as:

   <*Value*>http
   ://<*IPAddress*>:8080/EGateway/SimphonyPosApiweb.asmx</*Value*>

   Replace the placeholder <*IPAddress*> above with the IP Address of the workstation that hosts the TS API.
4. **Save** the changes.
5. Launch the **POSAPI_WebClient.exe**.

Demonstration

Calculate Totals of a Transaction

The Demo client has a button called Calculate Totals to invoke the *CalculateTransactionTotals* method of TS API. Follow the steps provided below to pass input data through UI and invoke the method.

1. Select the **MenuItem** from the **Type** dropdown field.
   a. Enter the Menu Item's Object Number **110003** in the **Number** field, and then click the **Add Item** button.
   b. Obtain the Menu Item Object Number from the EMC, click the **Property**, and then click the **Configuration** tab.
   c. Under the Menu Items heading, click **Menu Item Maintenance**.
2. Enter **3016** to the **RVC #** field.
   a. Obtain the RVC number from the EMC, click the **Property**, and then click the **Setup** tab.

b. Under the Property Configuration heading, click **Revenue Center Configuration**.

3. Enter **90001** in the **Employee #** field.
    a. Obtain the Employee Number from the EMC and click the **Property Configuration** tab
    b. Under the Personnel heading, click **Employee Maintenance** module.
4. Enter a valid vendor code to the License Activation Code field. This field can be empty for version 2.7 MR3 or later.
5. Click **Calculate Totals** to send the request to the TS API.

The status of operation shows at the bottom of the UI while the result appears within the block highlighted in green below (bottom right hand corner).

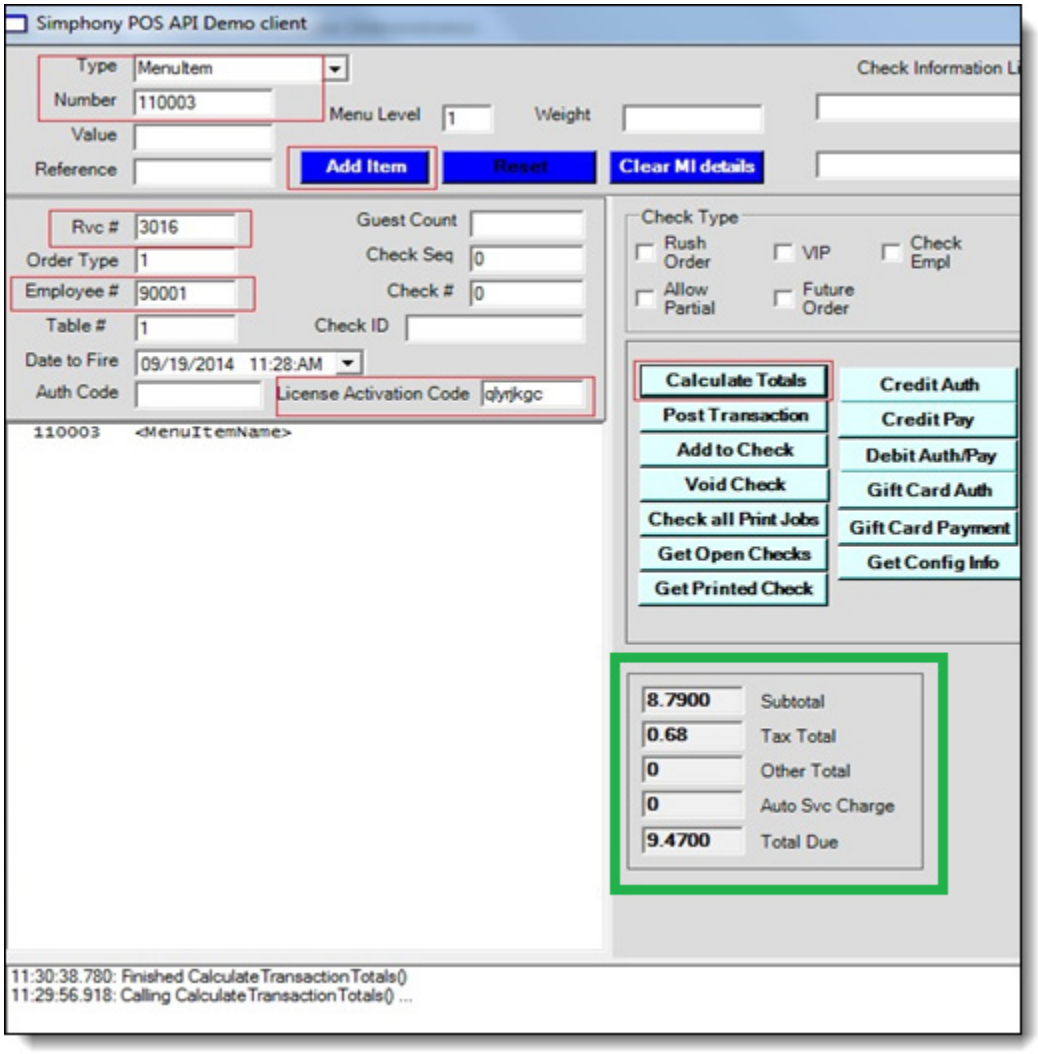

Figure 11-1 Simphony Transaction Services POS API Demo Client

Create a Guest Check

Post Transaction button can used to send a request to create a new Guest Check in the Simphony database.

1. Select the **MenuItem** option from the **Type** dropdown menu.

2. Enter the Menu Item's Object Number, **110003**, in the **Number** field and click the **Add Item** button.
3. Select **RequiredCondiment** from the **Type** dropdown menu.
4. Enter the Object Number, **41103,** in the **Number** field and click the **Add Item** button.
5. For payment, select **Tender** from the **Type** dropdown menu.
    a. For cash payment, enter the Tender Media Object Number (e.g., 2) for Cash to the **Number** field, and then enter the amount (e.g., 10.00) to the **Value** field.
    b. For a Debit/Credit payment, enter the Tender Media Object Number of the Credit/Debit tender, and then click the **Credit Auth** button to provide payment card details in the popup.
6. Click the **Add Item** button to add Tender details.
7. Enter values to the **RVC #**, **Employee #** and **License Activation Code** fields.
8. Click **Post Transaction** to send the request message to the TS API.

The result is populated in the fields highlighted in green below:

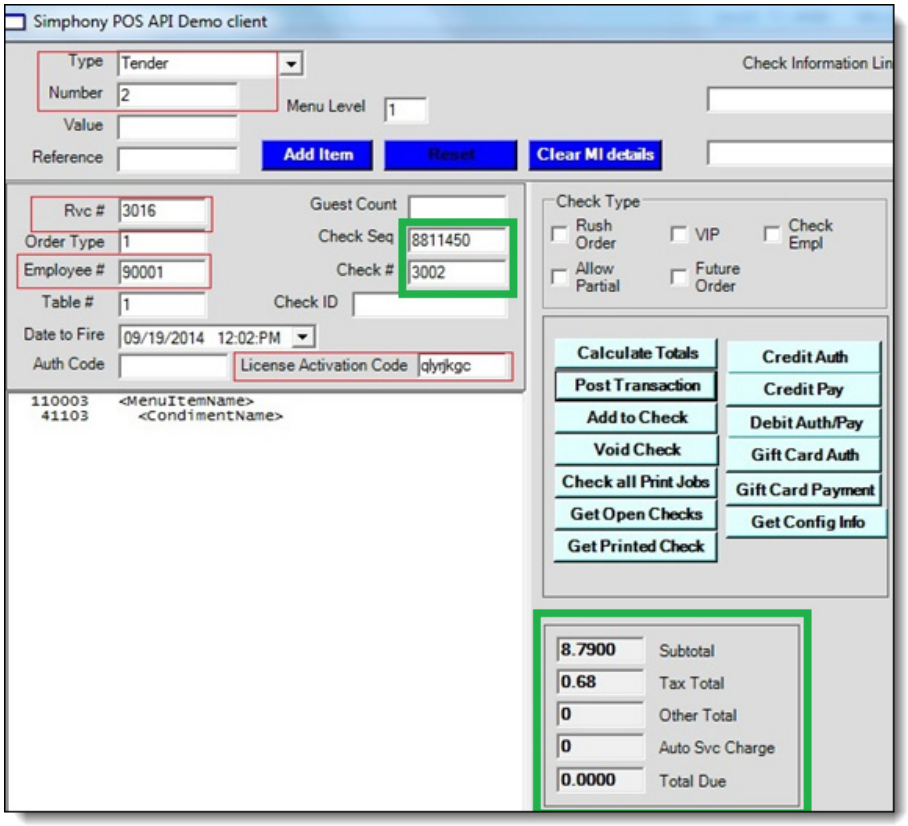

Figure 11-2 Creating a Check in the TS Demo Client

Add an item to an Open Guest Check

The Add to Check button can be used to add one or more items to an existing guest check.

This example, creates a guest check using **Post Transaction** first, and then adds one menu item to that check using the **Add To Check** button.

1. Select **MenuItem** from the **Type** dropdown menu.

2. Enter the Menu Item's Object Number **110003** in the **Number** field, and then click the **Add Item** button.
3. Select **RequiredCondiment** from the **Type** dropdown menu.
4. Enter Object Number **44502** in the **Number** field, and then click the **Add Item** button.
5. Select **Tender** from the **Type** dropdown.
6. Enter the Tender Media Object Number of the **Service Total** in the **Number** field, and then click the **Add Item** button.
7. Enter data for **RVC #**, **Employee #** and **License Activation Code** fields.
8. Press the **Post Transaction** button to create the guest check.

As the tender type is a Service Total, the check that's created is in an open state. Now, one or more items can be added to this open check by following the steps provided below.

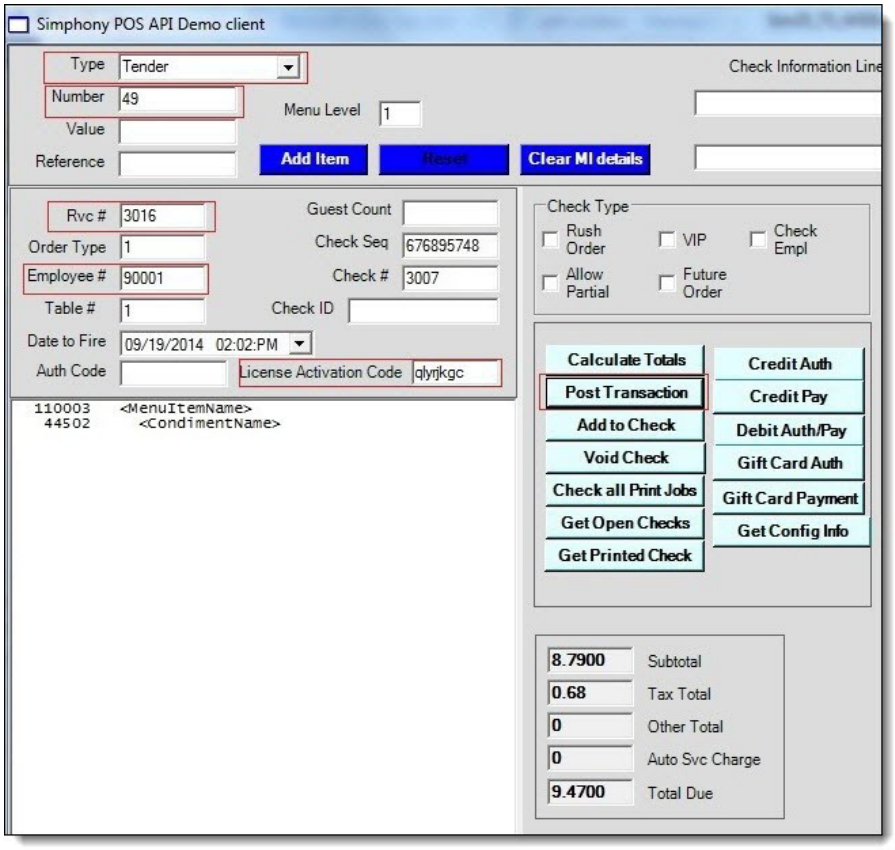

Figure 11-3 Adding Items to an Open Guest Check

9. Press the **Clear MI details** button to clear current details.

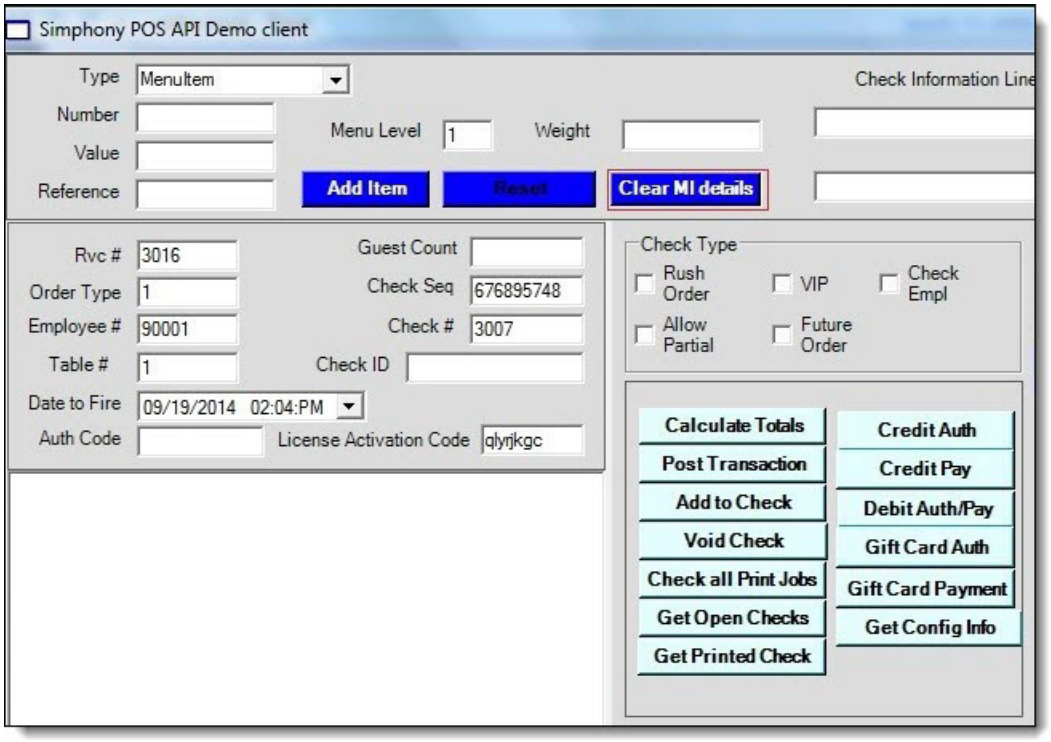

Figure 11-4 Adding Items to an Open Guest Check – continued

10. Select **MenuItem** from the **Type** dropdown menu to add a menu item to existing guest check.
11. Enter the Menu Item's Object Number **110004** in the **Number** field, and then click the **Add Item** button.
12. Select **RequiredCondiment** from the **Type** dropdown menu.
13. Enter Object Number **41103** in the **Number** field, and then click the **Add Item** button.
14. Ensure that the value of the **Check Sequence Number** and the **Check Number** of the original check is still there in **Check Seq** and **Check #** fields.
15. Click the **Add to Check** button.

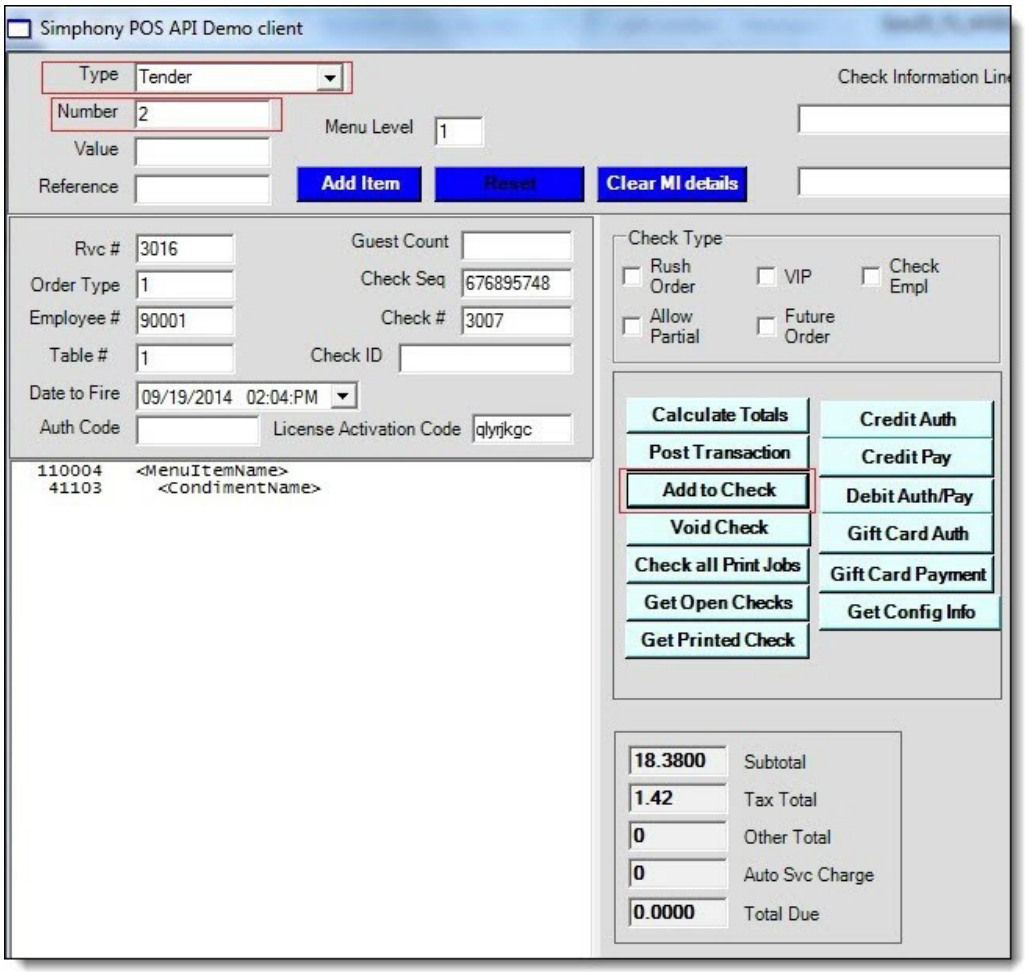

Figure 11-5 Adding Items to an Open Guest Check – continued

Combo Meal Ordering

Following steps explains how to add a combo meal to the check. Make sure that a Combo Meal is already configured in the EMC so that it can be rung up on the POS API.

1. Select **ComboMeal** from the **Type** dropdown menu.
2. Enter the Combo Meal Object Number in the **Number** field, enter the Object Number in the **Combo Menu Item Number** field, and then click the **Add Item** button.

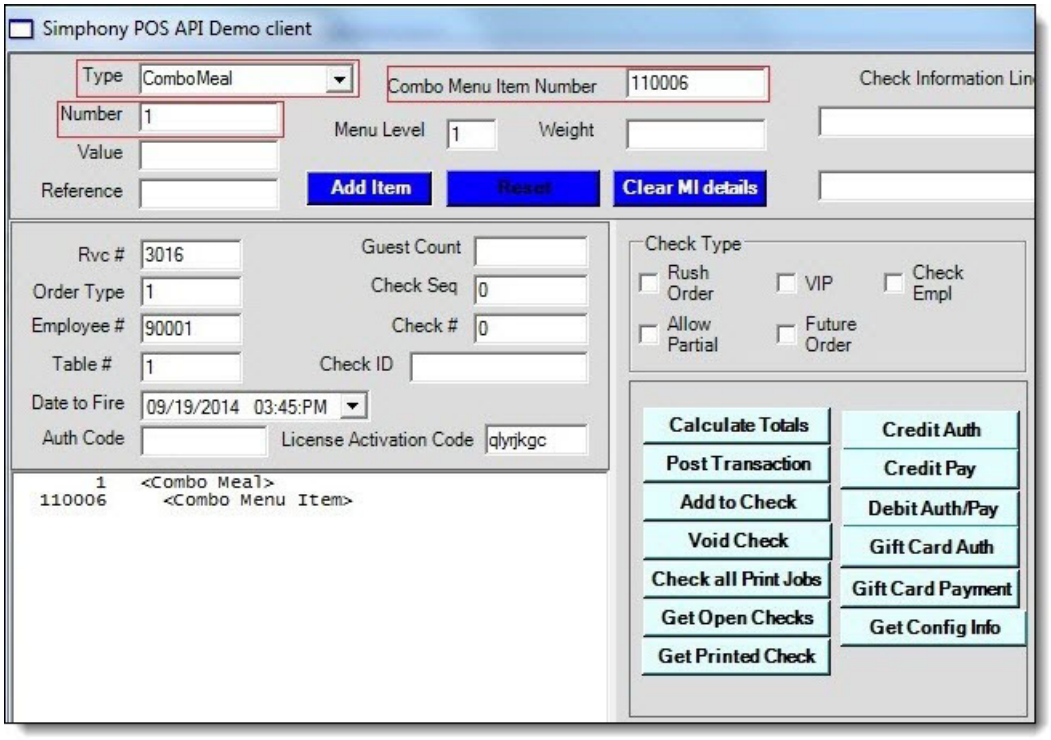

Figure 11-6 Combo Meal Ordering

3. Select **ComboMain** from **Type** dropdown menu to add a main item.
4. Enter the Combo Meal's Object Number in the **Number** field and then click the **Add Item** button.
5. Select **ComboSide** from the **Type** dropdown menu to add side items.
6. Enter the Side Item's Object Number in the **Number** field, and then click the **Add Item** button.
7. Click the **Calculate Totals** button to calculate the price of the combo meal, or add a tender and then click the **Post Transaction** button to create a guest check.

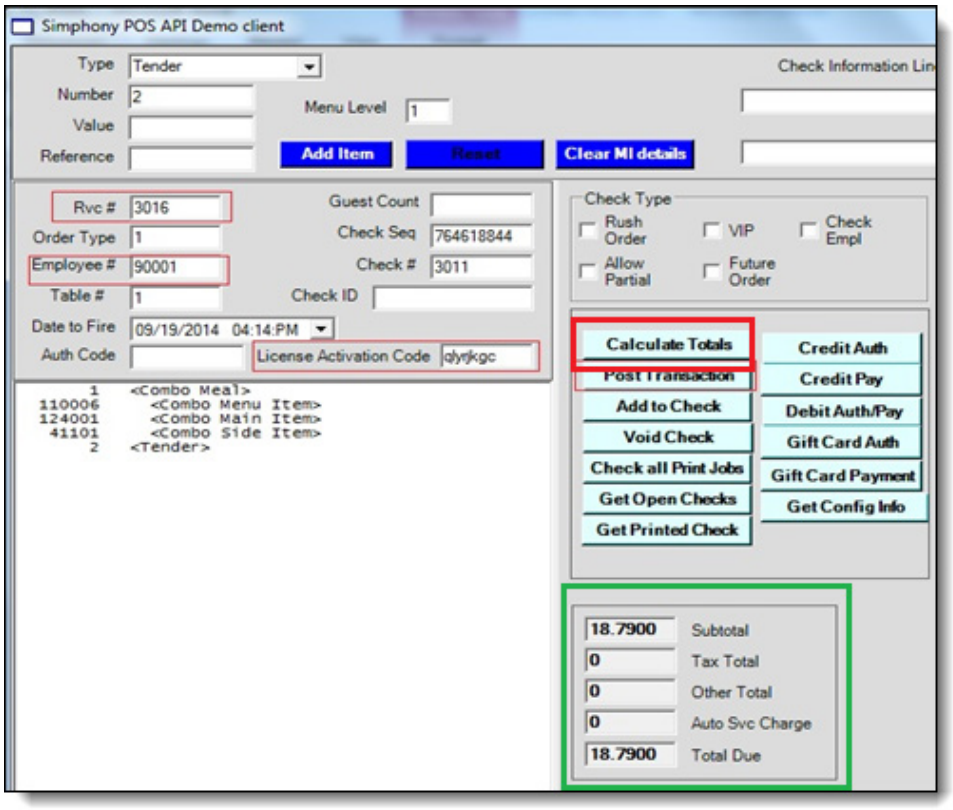

Figure 11-7 Adding Items to Combo Meals

Void All Items of an Open Guest Check

The **Void Check** button can be used to send a request to void all items of a guest check.

1. Enter the **Check Sequence Number** and the **Check Number** of the guest check in the relevant fields (highlighted in red) to void all items of that guest check. No other input is required to perform this operation.
2. Press the **Void Check** button to send void request to the TS API.

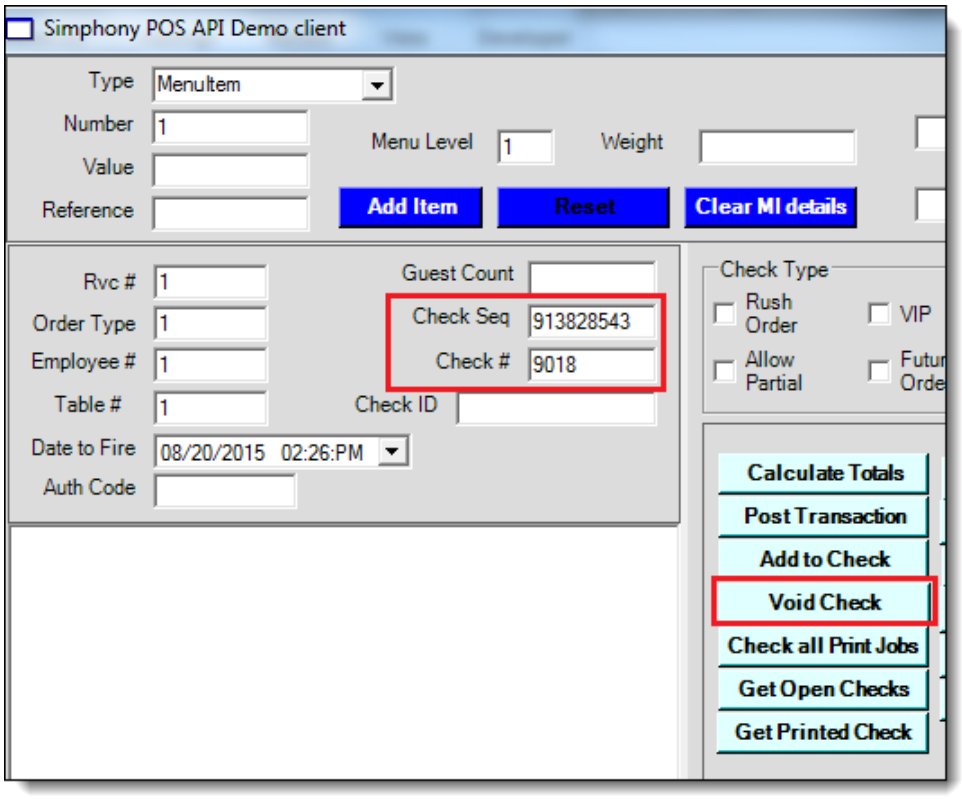

Figure 11-8 Void All Items on an Open Check

Get Summary of All Open Guest Checks

Get Open Checks button can be used to send a request to TS API to retrieve summary of all open guest checks from all or a specific revenue center of the property from Simphony POS database. This button calls different method *(*i.e., *GetOpenChecks, GetOpenChecksEx, GetOpenChecksByRVC)* of the TS API based on input given to the **Revenue Center** field of *FormGuestCheckParams* dialog, as explained below.

1. Enter the value for the **License Activation Code** field.
2. Click the **Get Open Checks** button.

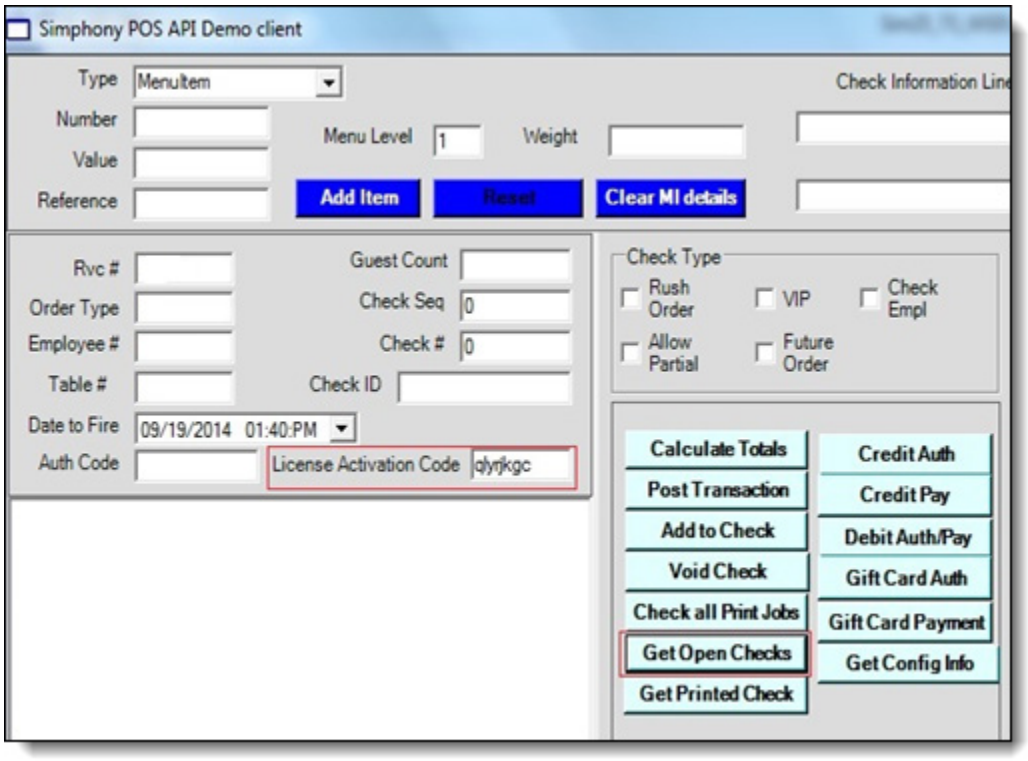

Figure 11-9 Get Summary of All Open Checks

3. The following dialog shows:

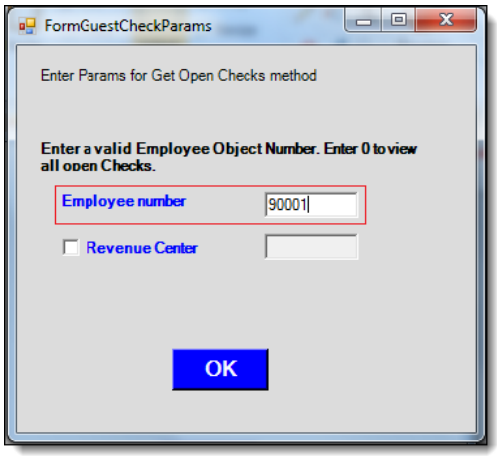

Figure 11-10 Get Open Checks Prompt

4. Provide **Employee Object Number** in the **Employee number** field to filter checks based on employee (or provide “0” to get all open checks irrespective of who created them) who created it.
5. Provide an appropriate value for the **Revenue Center** field and enable the check box too:
    a. Provide **-1**, if you want to retrieve open checks from all revenue centers of the property and display **ID** (instead of Object Number) of Revenue Center for each check in the output window
    b. Provide **-2**, if you want to retrieve open checks from all revenue centers of the property and display **Object Number** (instead of ID) of Revenue Center for each check in the output window
    c. Provide **Object Number of any RVC**, if you want to retrieve all open checks from that specific Revenue Center and display **ID** (instead of Object Number) of Revenue Center for each check in the output window

Provided below is the output window that shows a summary of all open checks.

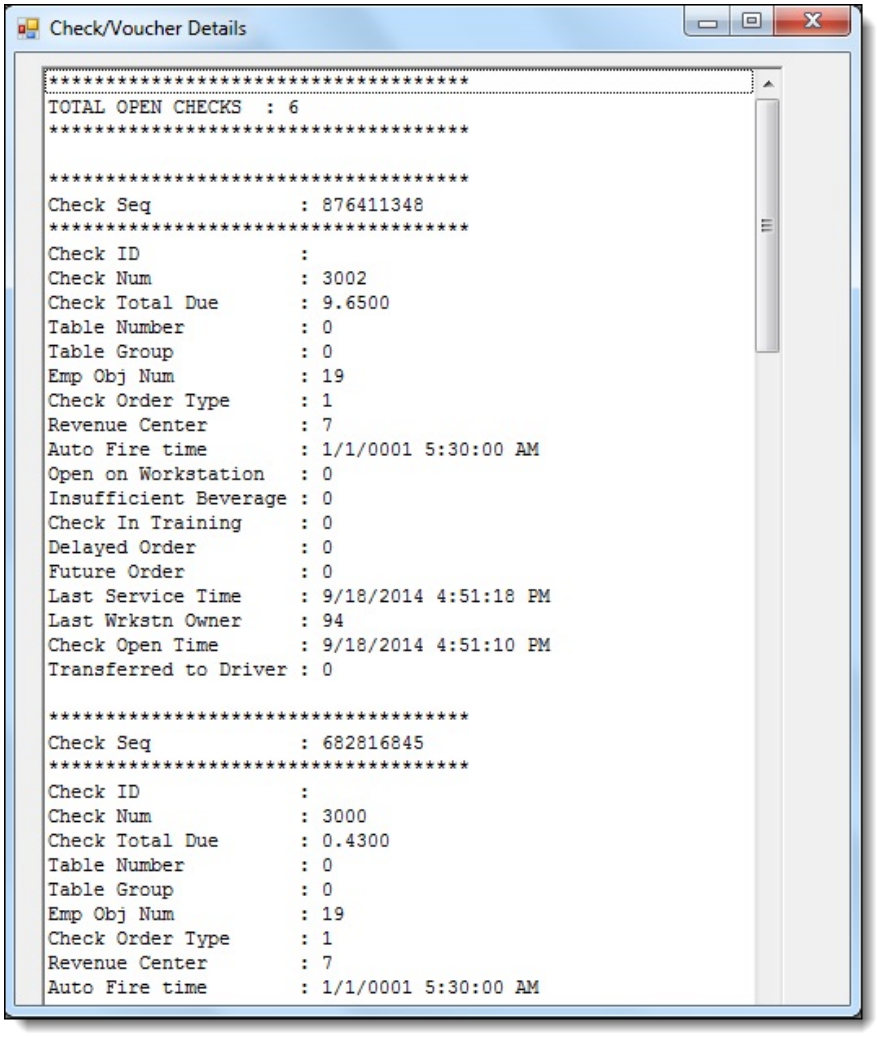

Figure 11-11 Check/Voucher Details

Get Printed Texts of a Guest Check

Get Printed Check is used to open posted checks by supplying the Check number and the Tender number.

1. Enter the **License Activation Code**.
2. Select the **Get Printed Check** button.

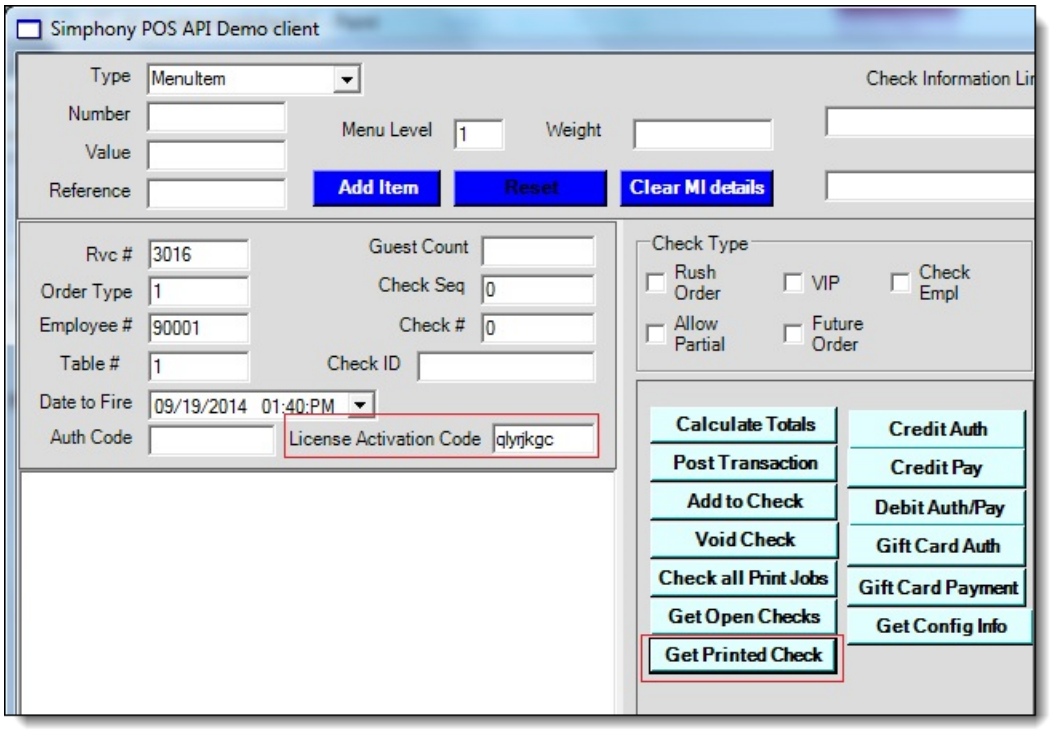

Figure 11-12 Get Printed Check

3. Enter an **Employee number**, **Check number** (or **Check Sequence**) and a **Tender Media Object Number** and click **OK**.

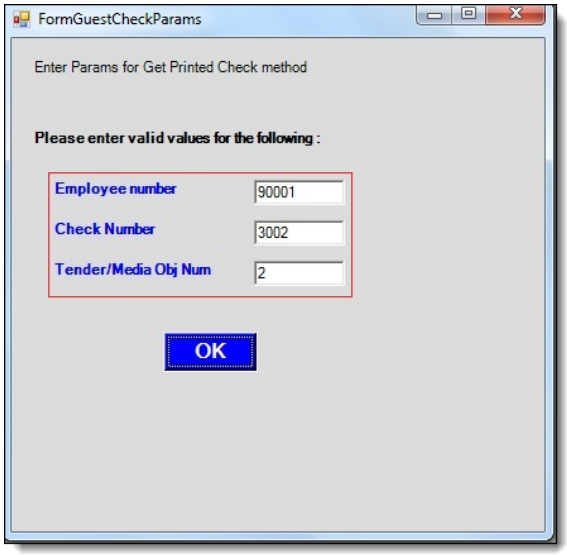

Figure 11-13 Get Printed Check Parameters

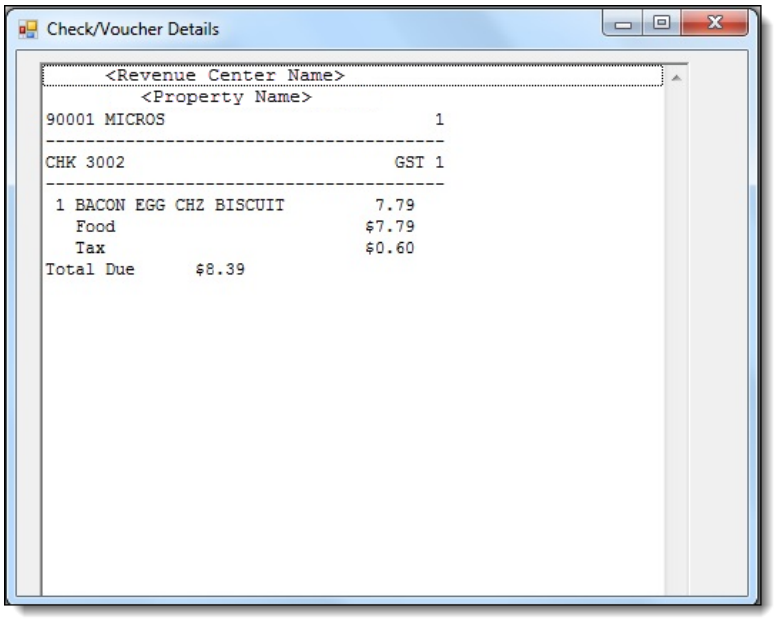

Figure 11-14 Get Printed Check/Voucher Details

Get Configured Information

1. Enter the **License Activation Code**.
2. Click the **GetConfigInfo** button on the client.

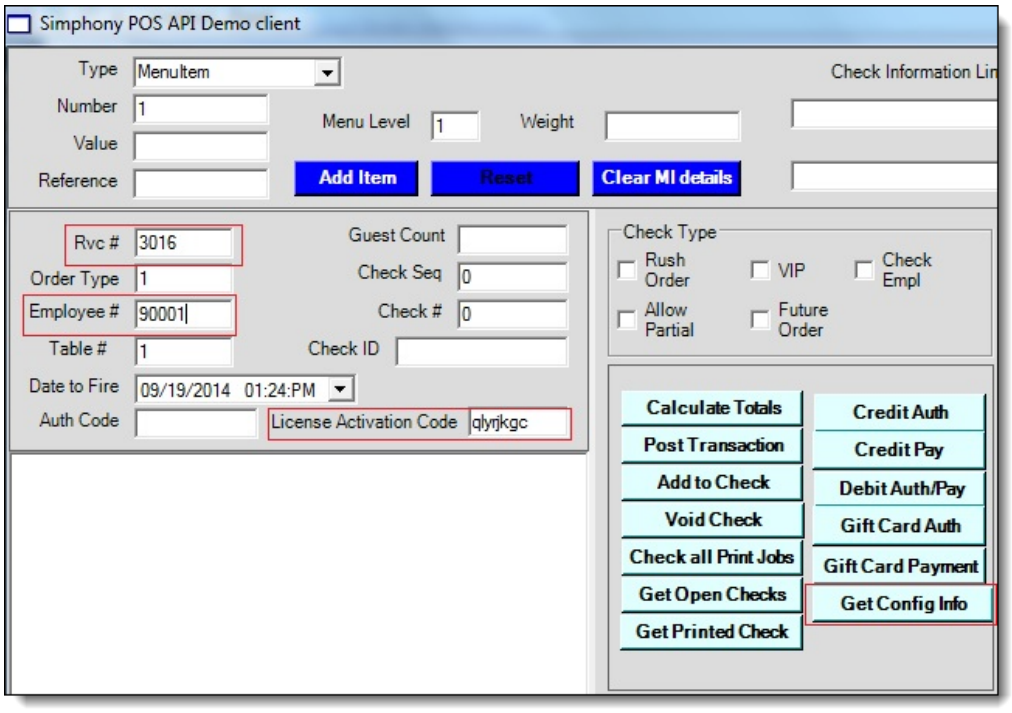

Figure 11-15 Get Config Info

3. Enter an **Employee Object Number** in the **Employee number** field and enter the applicable **Configuration Number** values listed below, separated by comma(s) and select **OK**.

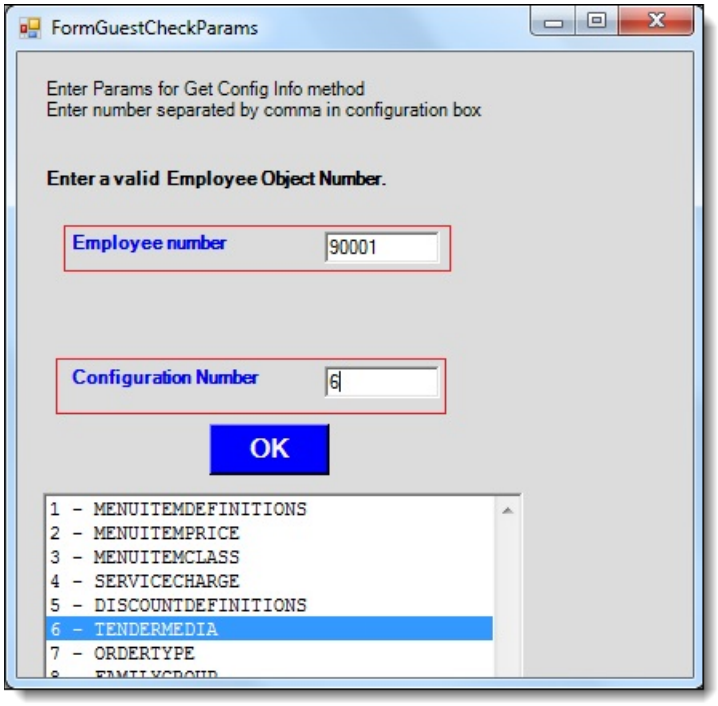

Figure 11-16 Get Configuration Parameters

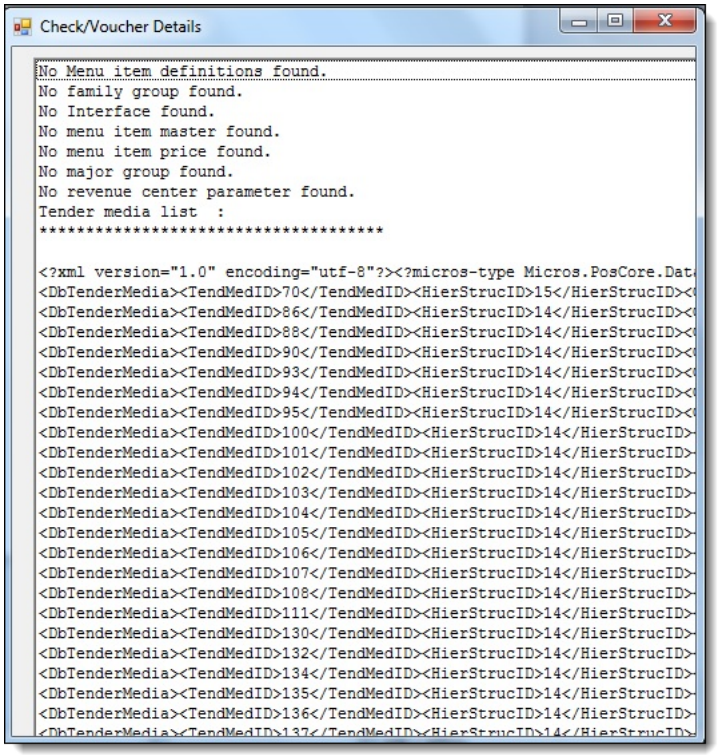

Figure 11-17 Get Config Check/Voucher Details