Oracle® Hospitality Simphony

Transaction Services API Document
Release 2.9
E92507-03

July 2018

ORACLE

Contents

Preface

This document is intended to be used by software engineers developing applications that interface with 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

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

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	
December 2012	
August 2015	
September 2016	
December 2017	Edited the following sections: • Simphony Architecture • License Requirement • Supported Operations • Structure Reference • Example and Code Snippets • Demo Client for Transaction Services API
March 2018	• Updated the Get Summary of All Open Guest Checks section (page 31) and the Check Voucher Details image (page 110)
July 2018	• Removed the License Requirement section that applied to version 2.7 MR3. This section is not valid for version 2.9.

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.

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, multiple tax rates, & 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.

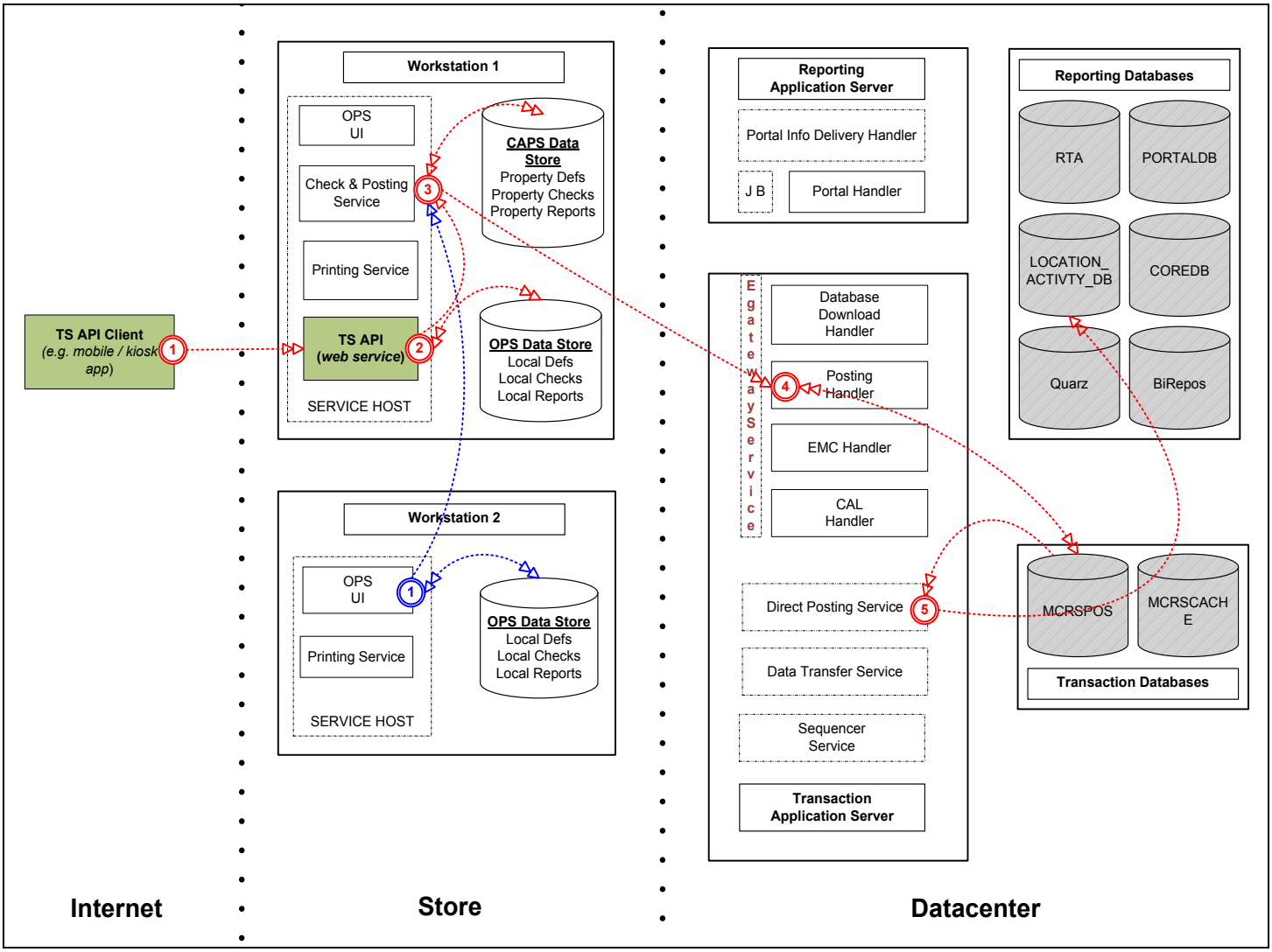
TS web service uses the same underlying business libraries that are being used by OPS for all POS operation. So, technically TS web service is a headless (i.e. UI less) version of OPS.

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.

The picture given below shows the architecture of Simphony system that includes Transaction Services API/web service as well.

The picture depicts how the data flows from a TS client application till reporting database.



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.

Hosting Method

The 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.

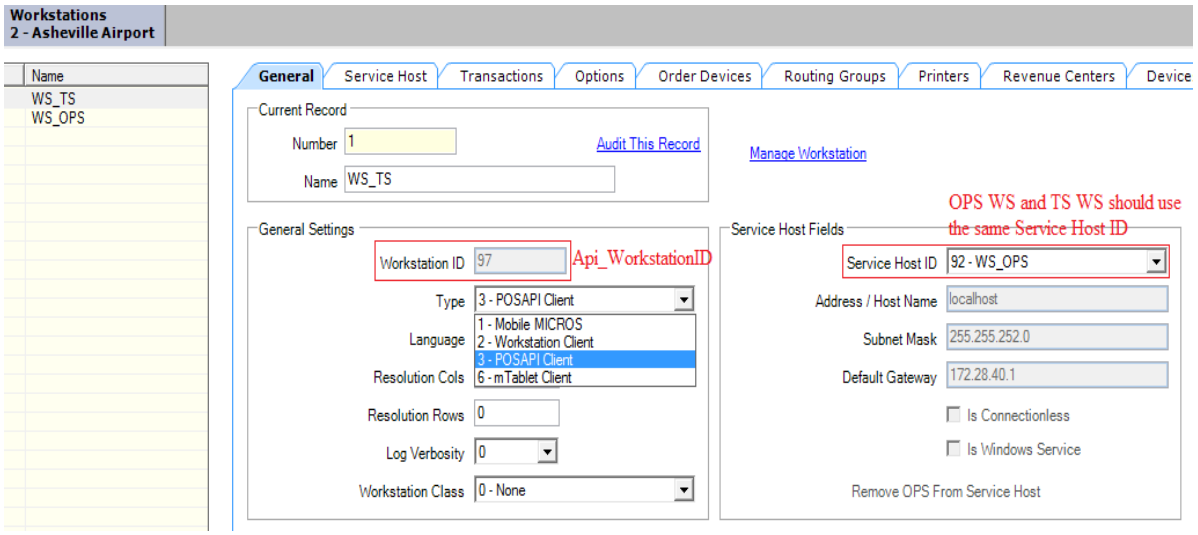
The Service Host that's configured via EMC to run on a workstation hosts the Transaction Services API as a web service by default at port number 8080. The format of web service URL is

`http://<<WorkstationIPAddress>>:8080/EGateway/SimphonyPosApiWeb.asmx`

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

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

1. Logon to EMC application.
2. Select a property that needs to configure.
3. Navigate to **Setup tab**.
4. Click **Workstation** link under Hardware/Interfaces.
5. Insert a new workstation as a POS API client (i.e. TS API).
6. Double-click the record to open the created workstation in Form view.
7. Select **2 – POSAPI Client** from the dropdown for Type under the **General Settings** section of **General tab**.
8. If OPS user interface is not needed to be hosted by s 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 the corresponding OPS from the Service Host ID dropdown field of Service Host Fields section.
10. **Save** all changes



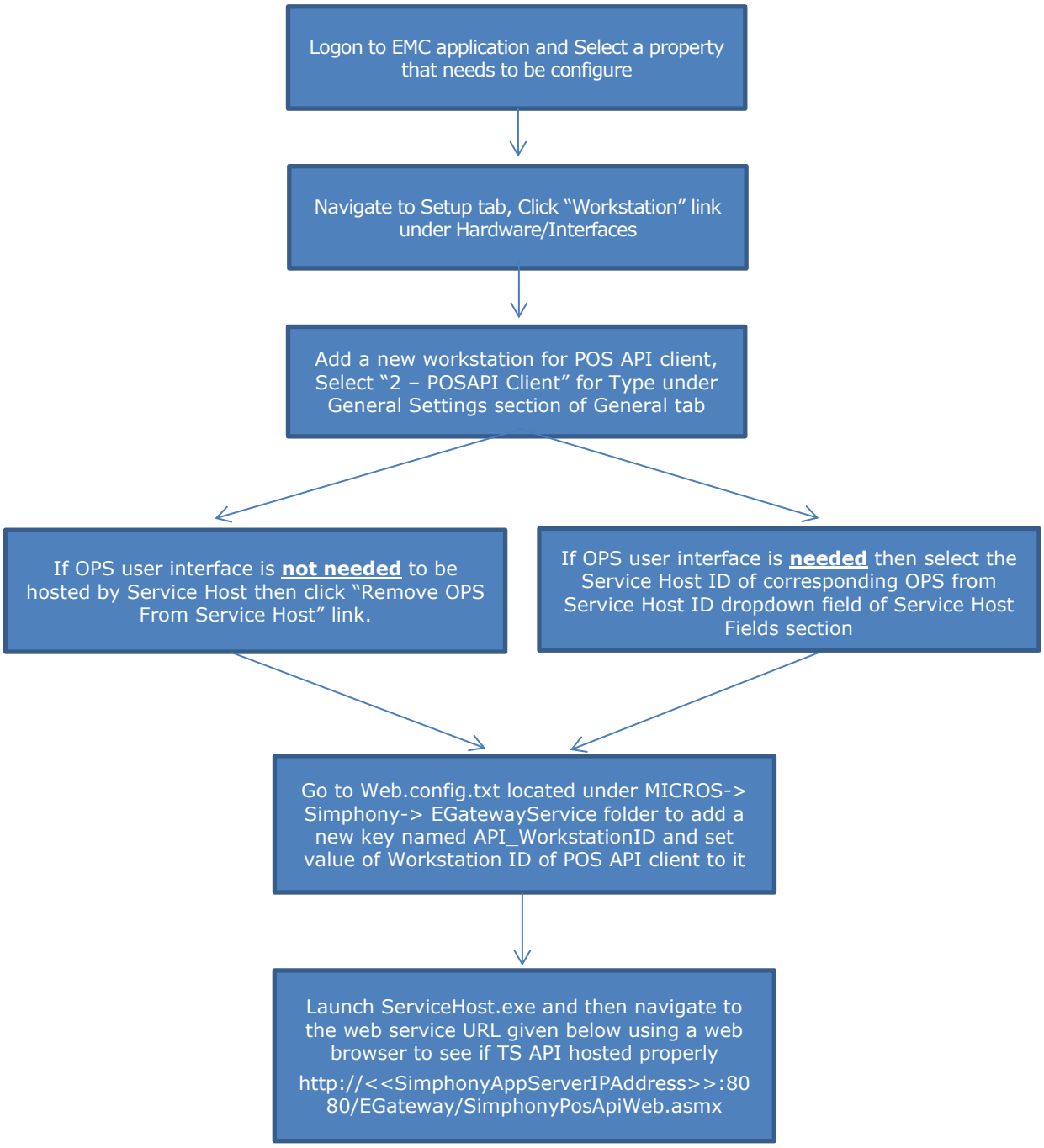
To verify whether or not TS web service is hosted correctly, first launch ServiceHost.exe on the POS workstation and then navigate to the web service URL given above using any web browser. If WSDL details are displayed in the web browser then it means web service is hosted properly.

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.

Quick Installation Roadmap



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

TS API Class Hierarchy

Here is a complete list of Interfaces and Structure provided by Transaction Services API for integration:

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
IGetCheckInfo Interface	Interface that provides support for fetching all open/closed guest checks, retrieving printed lines of guest check and configuration related information from the POS database
SimphonyPosAPI_CheckSummary Structure	Structure that defines summary of a guest check
SimphonyPosApi_CheckSummaryEx Structure	This structure derives from **SimphonyPosAPI_CheckSummary** and holds couple of
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_CheckRequest Structure	Structure that defines input parameters required to call GetChecks web method
SimphonyPosApi_CheckResponse Structure	Structure that holds the response of GetChecks web method
SimphonyPosApi_CheckDetailReques t Structure	Structure that defines input parameters required to call GetCheckDetail web method
SimphonyPosApi_CheckDetailRespon se Structure	Structure that holds the response of GetCheckDetail web method
SimphonyPosAPI_MenuItem Structure	Structure that holds the definition of Menu Item and it's Condiments
SimphonyPosAPI_MenuItemDefinitio n 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_ConfigInfoRequest Structure	Structure that defines input parameters required to call GetConfigurationInfoEx web method
SimphonyPosApi_ConfigInfo Structure	Structure that holds filter criteria to be used to retrieve configuration data using GetConfigurationInfoEx method
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_CheckPrintRespons e 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.

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.

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

```
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
)
```

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.

The default quantity of any menu item added to the transaction is just 1. When the quantity of any menu item should be set to more than one then the caller of this method is expected to pass the actual quantity via MiReference field of Menu Item Definition object in the following XML format.

"<ExtraData><MiQuantity>3</MiQuantity></ExtraData>"

In the example given above, 3 is the quantity. The regular reference text can be specified before or after the above XML. For example,

"<ExtraData><MiQuantity>3</MiQuantity></ExtraData>Make is spicy"

In the example above, the text "Make it spicy" will be treated as the reference text for the given menu item and the XML that defines quantity will not appear anywhere in the screen. Please refer to the code snippet section of this document for more details.

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.

Parameters

vendorCode	Vendor code for license validation (pass an empty value for Simphony version 2.7 MR3 or later)
ppMenuItems	List of menu items with required condiments and item discount. Quantity of *any* menu item can be specified in the MiReference field of Menu Item
ppComboMeal	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, Carry-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 *pTotalsResponse* reference parameter mentioned above.

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. Such guest check can be assigned to a particular event that's defined on corresponding property, by passing Object Number of Event Definition to EventObjectNum field of pGuestCheck parameter. This method 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. The default quantity of any menu item would be just 1 and if the caller wants to specify a different quantity then please refer to CalculateTransactionTotals method above for more details on how to pass a quantity for any menu item. 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 an empty value for Simphony 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, event object number, 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 get populated as results by this method.
ppMenuItem	List of menu items (with condiments) to be added to a given check. The quantity of any menu item can be specified in the MiReference field of the Menu Item Definition object in a predefined .xml format.
ppComboMeals	A list of combo meals (main and side items) to be added to given check.
pServiceChg	The 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 text of guest checks (output parameter).
ppVoucherOutput	Raw credit voucher (output parameter).

Return value

Void. The *OperationalResult* property of *pGuestCheck* parameter holds operation results. Also, *ppCheckPrintLines* holds the printed lines of guest checks while *ppVoucherOutput* holds printed lines of credit vouchers.

Add Items to Existing Guest 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. Also, the default quantity of any menu item would be just 1 and if the caller wants to specify a different quantity then please refer to CalculateTransactionTotals method above for more details on how to pass a quantity for any menu item

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.

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

1. CheckID
2. CheckTableObjectNum (when supported)
3. CheckOrderType

4. CheckEmployeeObjectNum
5. CheckDateToFire
6. pCheckInfoLines

The following fields will not be modified by this method.

1. CheckNum
2. CheckSeq
3. CheckRevenueCenterObjectNum

Parameter

vendorCode	Vendor code for license validation (pass an empty value for Simphony 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, event object number, 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 get populated as results by this method.
ppMenuItem	List of menu items (with condiments) to be added to a given check. The quantity of any menu item can be specified in the MiReference field of the Menu Item Definition object in a predefined .xml format.
ppComboMeals	A list of combo meals (main and side items) to be added to given check.
pServiceChg	The 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 text of guest checks (output parameter).
ppVoucherOutput	Raw credit voucher (output parameter).

Return value

Void. The *OperationalResult* property of *pGuestCheck* parameter holds operation results. Also, *ppCheckPrintLines* holds printed lines of guest checks while *ppVoucherOutput* holds printed lines of credit vouchers.

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 an empty value for Simphony version 2.7 MR3 or later).
pGuestCheck	Details of the guest checks to be voided.

Return value

Void. The *OperationalResult* property of *pGuestCheck* parameter will hold operation results.

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 an empty value for Simphony version 2.7 MR3 or later).
ppJobId	ID of a print job for which status needs to be retrieved.
ppJobStatus	Status of the print job (output parameter).

Return value

Void. Result is encapsulated in 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 those web methods are explained in the table below.

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 **GetChecks** (SimphonyPosApi_CheckRequest ppCheckFilter, ref SimphonyPosApi_CheckResponse ppChecksResponse) *Gets summary of both open and closed guest checks after applying given filter condition*
void **GetCheckDetail** (SimphonyPosApi_CheckDetailRequest ppCheckDetailFilter, ref SimphonyPosApi_CheckDetailResponse ppCheckDetailResponse) *Gets completes details of a guest check in xml format*
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 configuration data for one or more types from POS database.* ***Note:** This method returns all the records of specified configuration data type and it may throw "timeout" error when the POS database has a huge volume of configuration data for one or more types. The integrator can use the new method named GetConfigurationInfoEx (introduced in 2.9) in such cases to retrieve configuration data batch by batch by specifying ranges in the input parameter.*
void **GetConfigurationInfoEx** (SimphonyPosApi_ConfigInfoRequest configInfoRequest, ref SimphonyPosApi_ConfigInfoResponse configInfoResponse)

A new version of GetConfigurationInfo method to retrieve configuration data from POS database batch by batch by specifying ranges. This new method can be used instead of GetConfigurationInfo if the volume of configuration data is huge. Because, the other method named GetConfigurationInfo may throw "timeout" error when it tries to pull huge volume of records in one request.

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

Get Summary of All Open Guest Checks

```
void GetOpenChecks
(
    string                              vendorCode,
    int                                 employeeId,
    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 *employeeId* parameter. However, when 0 is passed to *employeeId*, 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 an empty value for Simphony version 2.7 MR3 or later).
employeeId	Employee Id of employees to filter open Guest Checks based on who created it. Pass specific employee Id 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 *openChecks* reference parameter.

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 2.7MR5 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* will hold Revenue Center Object Number instead of 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 an empty value for Simphony version 2.7 MR3 or later).
employeeObjectNum	Object number of employees to filter open Guest Checks based on who created it. Pass specific employee object numbers 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 *openChecks* reference parameter.

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 summary of open Guest Checks from a specific revenue center.

Method description

This method was introduced later in 2.7MR4 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 will return summary of all open checks from all revenue centers of the property.

Parameter

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

Return value

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

Get Summary and KDS order status of Open and/or Closed Guest Checks

```
Void   GetChecks
(
      SimphonyPosApi_CheckRequest          ppCheckFilter,
      ref SimphonyPosApi_CheckResponse     ppChecksResponse
)
```

Business purpose

User wants to see summary of both open and closed Guest Checks that satisfy one or more filter conditions. Also, this method can be used when the user wants to know KDS order status of one or more guest checks.

Method description

This method was introduced in v3.0 to get summary of both open and closed guest checks that satisfy one or more filter conditions. If no filter is passed then this method will return all "open" guest checks that are created on current business date in the default revenue center assigned to the POS API workstation. If closed checks need to be returned too, then the field *IncludeClosedCheck* of ppCheckFilter should be set to "true". This method applies filters based on data passed to following fields of ppCheckFilter.

- CheckNumbers

  Pass one or more check numbers to filter checks based on check numbers
- EmloyeeObjectNum

  Pass Object Number of an employee to get only checks created by that particular employee
- RvcObjectNum

  Pass Object Number of an RVC that's currently assigned to POS API workstation to get only checks created on that RVC
- OrderTypeID

  Pass Object Number of an Order Type to get guest checks created for that specific order type
- KdsOrderStatus

  Pass one or more KDS Order Status ID to retrieve checks that holds those status ID as their current KDS order status. Possible KDS order status IDs are given below.
  - 30 means DS_SENT (i.e. Order has been sent to the Kitchen with at least 1 menu item)
  - 50 means DS_PREP_DONE (i.e. Order has been prepared by at least one station)
  - 60 means DS_READY (i.e. Order has been expo done and is ready to be distributed to the customer)
  - 100 means DS_CANCELLED (i.e. Order cancelled)
- LookupStartDate

  Pass a date to retrieve only checks that were created after that given date
- IncludeClosedCheck

Pass “true” to retrieve closed checks too. When it’s “false” or value not specified then this method will return only “open” guest checks.

None of these fields mandates input data and the caller of this method can pass input data to one or more of these fields to filter the guest checks as needed.

The field VendorCode of ppCheckFilter is reserved for future use and is not currently used for any purpose.

Parameter

ppCheckFilter	Criteria based on which guest checks need to be filtered.
ppChecksResponse	Holds status of operation along with a summary of filtered guest checks. The Summary includes KDS order status as well.

Return value

Void. Result is encapsulated in *ppChecksResponse* reference parameter.

Get Check Detail

```
Void   GetCheckDetail
(
      SimphonyPosApi_CheckDetailRequest        ppCheckDetailFilter,
      ref SimphonyPosApi_CheckDetailResponse   ppCheckDetailResponse
)
```

Business purpose

User wants to see complete details of the guest check (including but not limited to Menu Items, Condiments, Discounts, Tax, Service Charge, Tender and Order Status via Extensibility Data).

Method description

This method was introduced in v3.0 to provide complete details of the guest check in xml format. The details include menu items, condiments, discounts, service charge, tender, extensibility data etc. Caller of this method is expected to pass Check Number and corresponding Check Sequence Number to retrieve details on that check. Also, the caller is expected to parse the output check xml to extract required data.

To take advantage of OIS (i.e. Order Information Services) feature that's introduced in v3.0, the caller is expected to parse the extensibility data section (detail type is DtlExtensibilityDataType) of this XML to retrieve the order status. However, please note that this feature requires few configurations to be done from Simphony server end with the help of EMC application before the order status can be retrieved using this method of Transaction Services. The value of ExtensibilityAppName node in the check XML can be used to identify the extensibility data that are specific to OIS (this app name is configured in Simphony using EMC application). The StringData node (sibling of ExtensibilityAppName) would contain timestamp (apart from order status) in order to identify the most recent status of the order. Please refer to the sample check XML (Filename is SampleGuestCheckXML.txt) that's attached in this section to see how the extensibility data (ExtensibilityAppName is OIS) with OIS details looks like. Also, please refer to the code snippet section of this document to see a sample code that reads and prints the most recent status of the order from check XML.

Here's the format/template of check xml for quick reference.

```
<?xml version=”1.0” encoding="utf-16"?>
<Check>
  <Summary>
    <![CDATA[This section will contain a number of nodes on check summary]]>
      ...
      ...
      ...
  </Summary>
  <DetailLines>
    <![CDATA[This section will contain a collection of DetailLine nodes]]>
    <DetailLine>
      <DetailType>DtlType</DetailType>
      <DetailLink>LinkID</DetailLink>
      <ParentDetailLink>ParentLineID</ParentDetailLink>
      ...
      ...
      ...
    </DetailLine>
```

```
    ...
    ...
    ...
    <DetailLine>
      <DetailType>DtlType</DetailType>
      <DetailLink>LinkID</DetailLink>
      <ParentDetailLink>ParentLineID</ParentDetailLink>
      <ServiceCharge>
      </ServiceCharge>
    </DetailLine>
  </DetailLines>
</Check>
```

As shown above, check xml will have following two sections:

1. Summary
   - This section of xml holds a number of child nodes to contain summary data like Check Number, Check Open Time, Check Close Time, Sub Total, Tax, Amount Due etc.
2. DetailLines
   - This section will contain a collection of DetailLine nodes. Each DetailLine node indicates an item added to the guest check. It could be a menu item, service charge, discount or tax etc. Each DetailLine node will contain following 3 child nodes apart from other nodes.
     - DetailType

       This node describes the type of the item. Following table provides the complete list of detail types.

ID	Type
DtlMiType	Menu item or condiment
DtlDscType	Discount
DtlSvcType	Service Charge
DtlTmedType	Tender Media
DtlRefType	Reference Text
DtlExtensibilityDataType	Extensibility data

     - DetailLink
       - This node holds a number to uniquely identify an item in the guest check xml.
     - ParentDetailLink
       - This node holds the ID of its parent DetailLine node. For example, a condiment item will hold the DetailLink ID of its main menu item as ParentDetailLink.

Apart from these 3 nodes, each DetailLine node will contain other child nodes based on the type of DetailLine node.

Herewith attach a sample check xml for reference.

The guest check corresponding to this xml had following items in it.

- Two menu items named Pizza and Nachos
- 6% add-on tax
- 10% discount
- 10% service charge
- Cash payment of $40
- Order Status (Assigned & Delivered) via Extensibility Data

Parameter

ppCheckDetailFilter	Input parameter to pass the Check Number and Check Sequence Number.
ppCheckDetailResponse	Output parameter that holds check detail in xml format.

Return value

Void. Result is encapsulated in *ppChecksResponse* reference parameter.

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 an empty value for Simphony 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.
ppCheckPrintLines	This holds an array of printed lines of the check along with response code.

Return value

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

Get Configured Information *(method 1)*

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

Business purpose

User wants to fetch configuration data from Simphony POS database.

Method description

This method returns configuration data like menu item definition, menu item price found in the POS database. If the volume of configuration data is found to be huge then this method may throw timeout error. So, in such cases, the client application can call a new version of this method named GetConfigurationInfoEx that is explained in the next section. The new method will return configuration data for only a specified range of records. The caller will have to call this new method multiple times to retrieve complete list of records. Please go through the next section for more details on the new version of this method. Given below are the list of configuration data that can be retrieved using GetConfigurationInfo method.

1. Menu Item Definitions
2. Menu Item Prices
3. Menu Item Classes
4. Service Charges
5. Discounts
6. Tender Media
7. Order Types
8. Family Groups
9. Major Groups
10. Revenue Center Parameters
11. Revenue Center Configurations
12. Interfaces
13. Menu Item Masters
14. Serving Periods
15. Currencies
16. Product version
17. Employees
18. Dining Tables
19. Languages
20. Menu Level Set
21. Menu Item SLU
22. Main Menu Levels
23. Sub Menu Levels

24. Events Definitions
25. TAX

Parameter

vendorCode	Vendor code for license validation (pass an empty value for Simphony 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.
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 *configInfoResponse* reference parameter.

Get Configured Information *(method 2)*

```
void GetConfigurationInfoEx
(
    SimphonyPosAPI_ConfigInfoRequest      configInfoRequest,
    ref SimphonyPosAPI_ConfigInfoResponse configInfoResponse
)
```

Business purpose

User wants to fetch configuration data for a specified range of records from Simphony POS database.

Method description

As mentioned in the previous section, this is a new version of GetConfigurationInfo method. The main difference between these two methods is that this new method would return configuration data for only a specified range of records while GetConfigurationInfo returns all records by default. This new method can be used if the given system has huge volume of configuration data for one or more types. For example, menu item definition has more than 30,000 records.

Parameter

configInfoRequest	Request parameter that holds filter conditions to be applied while retrieving configuration data. This includes configuration data type and ranges (i.e. StartIndex & MaxRecordCount).
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 *configInfoResponse* reference parameter.

Structure Reference

SimphonyPosAPI_CheckSummary

This structure is to encapsulate summary details of a check.

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_CheckSummaryEx

This structure is an extended version of SimphonyPosAPI_CheckSummary structure. This extended version is created to hold couple of additional fields on KDS Order status. This structure inherits all the fields from SimphonyPosAPI_CheckSummary.

Public Attributes

SimphonyPosApi_KdsOrderStatus	**LastKnownKdsOrderStatus** *Status reported by KDS device last time on an order (e.g. DS_SENT, DS_PREP_DONE, DS_CANCELLED)*
List<SimphonyPosApi_KdsOrderStatus>	**KdsOrderStatusHistory** *History of order status reported by KDS on each menu item of an order*

SimphonyPosAPI_OpenChecks

This structure is to encapsulate details of all 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.

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*
int EventObjectNum *Object Number of an event definition that needs to be associated to given guest check*
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_CheckRequest

This structure is to encapsulate the input parameters of GetChecks method.

Public Attributes

string VendorCode *Vendor code for license validation (pass an empty value for Simphony version 2.7 MR3 or later).*
Int[] CheckNumbers *Check Number(s)*
int EmployeeObjectNum *Object Number of an employee*
int RvcObjectNum *Object Number of a Revenue Center*
int OrderTypeID *Object Number of an Order Type*
Int[] KdsOrderStatus *Status ID of KDS Order*
DateTime LookUpStartDate *Lookup start date. All checks that were created on/after this date would be returned in the result.*
bool IncludeClosedCheck *A boolean flag to specific whether or not closed checks need to be included in the result*

SimphonyPosAPI_CheckResponse

This structure is to encapsulate the response of GetChecks method.

Public Attributes

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_CheckSummaryEx Checks *Extended summary of guest checks that includes KDS order status*

SimphonyPosAPI_CheckDetailRequest

This structure is to encapsulate the input fields of GetCheckDetail method.

Public Attributes

string VendorCode *Vendor code for license validation (pass an empty value for Simphony version 2.7 MR3 or later).*
int CheckNumber *Guest Check Number*
int CheckSeqNumber *Sequence Number of Guest Check*

SimphonyPosAPI_CheckDetailResponse

This structure is to encapsulate the response of GetCheckDetail method.

Public Attributes

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*
string CheckDetail *Check Detail in XML format*

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).

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

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.

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 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

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

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

Note: Transaction Services ONLY supports **MCreditDebit Payment** driver for credit/debit card payment.

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

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

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_ConfigInfoRequest

This structure is to encapsulate the input parameters like Vendor Code, Employee Object Number, RVC Object Number and Configuration Info Types with range conditions.

Public Attributes

string **VendorCode** *The Vendor Code for license validation (pass empty for Simphony version 2.7 MR3 or later*
int **EmployeeObjectNumber** *Employee Object Number for validation purpose only*
int **RVCObjectNumber** *Object Number of the Revenue Center for which configuration data is needed*
ARRAY(SimphonyPosApi_ConfigInfo) **ConfigurationInfo** *This holds the IDs of configuration data type along with start index and maximum records to be returned for each configuration data type*

SimphonyPosAPI_ConfigInfo

This structure is to encapsulate the input parameters like Configuration Info Type and ranges of records to be retrieved.

Public Attributes

EConfigurationInfoType **ConfigurationInfoTypeID** *The type of configuration data that need to be fetched from Simphony POS database (e.g. menu item definition, service charge definition, discount definition etc.).*
int **StartIndex** *Index of first record to be fetched from the POS database*
int **MaxRecordCount** *Maximum Number of Records to be fetched*

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.

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
- MENUITEMSLU = 21
- MAINMENULEVEL = 22
- SUBMENULEVEL = 23
- EVENTDEFINITIONS = 24
- TAX = 25

string MenuItemDefinitions

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

string MenuItemPrice

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

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 ServiceCharge

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

string Discounts

Details of all discounts configured in EMC at enterprise level

string TenderMedia

Details of tender media configured in EMC for payment (e.g. Cash, and Credit Cards) **Note**: Transaction Services ONLY supports the **MCreditDebit Payment** driver for credit/debit card payment.
string **OrderType** *Details of order types (e.g. dine-in, dine out etc.) configured in EMC for given property*
string **FamilyGroups** *Details of all family groups (i.e. category of menu items) configured in EMC*
string **MajorGroup** *Details of all major groups configured in EMC for menu items (e.g. Food, Beverages)*
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 **RevenueCenters** *Details of revenue centers configured in EMC for given property*
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)*
string **ServingPeriod** *Details of serving period (e.g. Breakfast 4am to 11am)*
string **Currency** *Details of all currencies (e.g. US Dollar, Peso etc) configured in EMC at enterprise level*
string **Version** *Current version of the Transaction Services web service (e.g. 2.700.0.77)*
string **Employees** *Details of all employees configured in EMC at enterprise level*
string **Tables** *Details of dining tables configured in EMC for given property*
string **LanguageInformation** *Details of languages (e.g. English, Spanish) that are configured via EMC*
string **MenuLevel** *Details of menu level sets configured (e.g. Main, Sub and Custom levels)*
string **MenuItemSlu** *Details of menu item SLU names(user can configure a maximum of 127 SLU names)*
string **MainMenuLevel** *Details of main menu levels (user can configure a maximum of 8 main levels)*
string **SubMenuLevel** *Details of sub menu levels (user can configure a maximum of 8 sub levels)*
string **EventDefinitions** *Details of event definitions created at property levels*

string **TAX** ***Details of tax rates configured in EMC at Enterprise and Property levels (maximum of 64 tax rates)***
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

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

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

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,
- InvaildAuthCode,
- 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*

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.

- Ring in two menu items
  - Ring in two condiments to first menu item, and one condiment to second menu item
  - Quantity of 1st menu item is 3 and 2nd menu item is just 1
  - Override price of first menu item
  - For first Menu Item, instead of default definition, pick up a specific MI definition based on main and sub menu levels
  - Apply an "Open" discount to first menu item
  - Add a reference text to first menu item
- Ring in a combo meal
- Apply an "Open" service charge
  - Add a reference text on service charge
- Apply a "Closed" discount on subtotal (i.e. check level)

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 o ***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** - Object number of main item is **110003** - Object numbers of side items are **41103** and **44502** - Object number of drink is **110004**
4	Service Charge	*pServiceChg*	• "Open" service charge is **6%** 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)

Order type is Dine-in

Provided below is sample data for the scenario mentioned above.

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

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                                     revenueCenterObjectNum,
      short                                   orderType,
      int                                     employeeObjectNum,
      ref SimphonyPosAPI_TotalsResponse       pTotalsResponse
 )
```

Following code snippet demonstrates how data for input parameters of ***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));
    }
}
```

Following sections explain constructing data for each input parameter with sample data given above.

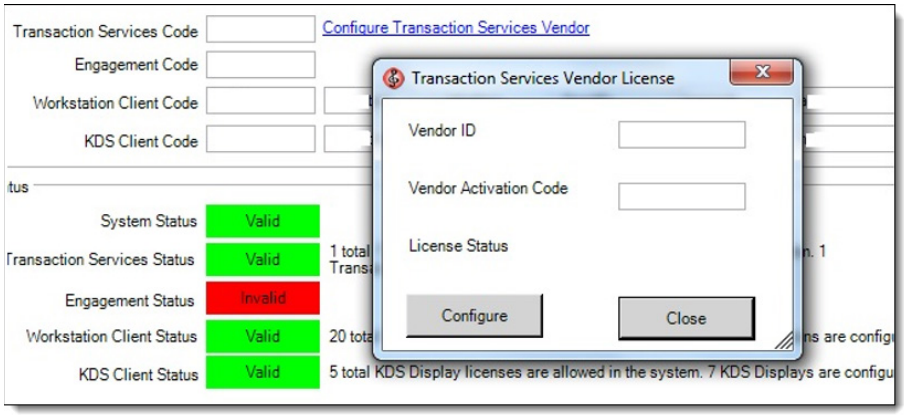
Vendor Code

Vendor Code is no longer supported beyond Simphony version 2.7 MR3. This should be an empty value from clients. 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 have been configured in EMC under following location for TS API to work properly.

Enterprise level, **Setup** tab, **Parameters** section, **Enterprise Parameters** module, **Licensing** tab.

This requirement has been removed from Simphony version 2.7 MR3 and later. However, this parameter is still out there even in the latest Transaction Services API for backward compatibility. Client applications that integrate with Transaction Services API with Simphony 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.

Below is the EMC that that shows the dialog to configure the Vendor Activation Code for the Transaction Services 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 using following module.

Enterprise level, **Configuration tab**, **Menu Items** section, **Menu Item Maintenance** module.

Parameter Signature

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

SimphonyPosAPI_MenuItem signature

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

Menu Item is the core foundation of all POS transactions. Everything "ordered" or "rung in" for the systems is a Menu Item. In restaurant kind of revenue centers, it is obvious that drinks and entrees, etc., are Menu Items. Perhaps less obviously, 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 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, 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 first menu item in order to pick up a particular definition instead of default. A reference text also added to first menu item for reference purpose.

```
private SimphonyPosApi_MenuItemDefinition GetFirstMenuItem()
{
    SimphonyPosApi_MenuItemDefinition menuItemDefn = new
        SimphonyPosApi_MenuItemDefinition();
    menuItemDefn.MiObjectNum = 110003;
    menuItemDefn.MiOverridePrice = "7";
    menuItemDefn.MiMenuLevel = 2;
    menuItemDefn.MiSubLevel = 3;

    // Specify 3 as quantity and 'Make it spicy' as reference text
    menuItemDefn.MiReference = "<ExtraData><MiQuantity>3</MiQuantity></ExtraData>Make
it spicy";
    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 object for two condiment menu items with given 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 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 a 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;
}
```

Code snippet given 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. Please login to EMC to get object numbers of combo meal and related items.

```
private SimphonyPosApi_ComboMeal[] GetComboMealList()
{

SimphonyPosApi_ComboMeal[] comboMeal = new SimphonyPosApi_ComboMea
l[1];

    SimphonyPosApi_ComboMeal comboMeal1 = new SimphonyPosApi_Combo
Meal();
    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_Me
nuItem[2];
    SimphonyPosApi_MenuItem firstSideItem = new SimphonyPosApi_Men
uItem();
    firstSideItem.MenuItem = new SimphonyPosApi_MenuItemDefinition
();
    firstSideItem.MenuItem.MiObjectNum = 41103;
    sideItemList[0] = firstSideItem;
    SimphonyPosApi_MenuItem secondSideItem = new SimphonyPosApi_Me
nuItem();
    secondSideItem.MenuItem = new SimphonyPosApi_MenuItemDefinitio
n();
    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.

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

Service Charge can be configured in EMC under

Enterprise or **Property** level, **Configuration** tab, **Sales**, **Service Charges** module.

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 isOpenServic
eCharge)
{

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

    if (isOpenServiceCharge)
       serviceCharge.SvcChgAmountOrPercent = "6"

    serviceCharge.SvcChgReference = "6% service charge including ti
ps";

    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 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 isOpenDisc
ount)
{

SimphonyPosApi_Discount subTotalDiscount = new SimphonyPosApi_Disco
unt();
     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 can be found in **EMC**, **Property** level, **Setup** tab, **Property Configuration**, **RVC Configuration** module. **3016** is the Object Number of a 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 EMC under following module:-

Property level, **Configuration** tab, **Personal**, **Employee Maintenance** module.

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.

Note: Transaction Services ONLY supports the **MCreditDebit Payment** driver for credit/debit card payment, there is no Stored Value Card (SVC) support for Transaction Services.

The method PosTransactionEx can be used for this 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

Post Transaction method posts current transaction to Simphony POS database to create a guest check. Like 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 tender media with appropriate payment details (cash, credit/debit) 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 calculate 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 the 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 operation, caller of the method is expected to pass data for following mandatory properties of guestCheck parameter.

CheckEmployeeObjectNum

CheckRevenueCenterID

CheckOrderType

Following properties are optional

CheckGuestCount

CheckTableObjectNum

CheckStatusBits

CheckDateToFire

Following properties will be populated by the API when operation succeeds. In case of failure, only OperationalResult will populate to hold data for 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 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 balance amount to the same check by specifying Check Number and Check Sequence

Number of that check. Example of AddToExistingCheckEx method is given in the next section.

Tender/payment

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=000100020003707831 49";
           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=000200030004707832 49";
           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;
}
```

For details on other input parameters of this method, please refer to “Calculate Totals of a Transaction” section of this document.

API Response

If post transaction operation has succeeded then the output details like Check ID, Check Number and Check Sequence Number of created check will be populated in 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 like payment failure then it will not create guest check and it will throw an appropriate exception to the caller. Also, the OperationalResult property of pTotalsResponse parameter will hold 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, he/she may needs to update it. For example, we can take a case where a transaction was posted to create a guest check with Service Total as tender media. That is, payment has not been made yet for that transaction. In that case, user 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.

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 be added an open guest check too.

```
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 given 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, please refer to "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 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, user may want to get summary of all open Guest Checks from all revenue centers of the property.

GetOpenChecks method signature

```
void   GetOpenChecks
(
       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 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 summary of open Guest Checks will be populated in 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 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 RVC Object Number (instead of RVC ID) then GetOpenChecks**Ex** method can be used instead of GetOpenChecks that's explained in the previous section. Because, GetOpenChecks populates CheckRevenueCenterObjectNum with ID of revenue center while GetOpenChecks**Ex** populates the same field with 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 he/she wishes to fetch all open checks irrespective of the owner who created the check. The property CheckRevenueCenterObjectNum of response object returns object number of RVC instead of ID. Object number is different from 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 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, user may want to see summary of all open Guest Checks from a specific revenue center of the property. In that case, following method can be used.

GetOpenChecksByRVC method signature

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

Code snippet given below demonstrates retrieving summary of all 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 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
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 summary of open Guest Checks will be populated in *openChecks* 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 Summary and KDS order status of Open and Closed Guest Checks

If user wants to see summary of Guest Checks that satisfies few filter conditions then following method can be used.

GetChecks method signature

```
void  GetChecks
(
      SimphonyPosApi_CheckRequest  ppCheckFilter,
      SimphonyPosApi_CheckResponse  ppChecksResponse
)
```

Code snippet given below demonstrates how to retrieve both open and closed guest checks that were created in the last 5 days by an employee whose object number is 90001.

```
SimphonyPosAPIWebSoapClient mTSApi = new SimphonyPosAPIWebSoapClient();

public void InvokeGetChecks()
{
    const int NUMBER_OF_DAYS = 5;

    // Construct request object
    SimphonyPosApi_CheckRequest request = new SimphonyPosApi_CheckRequest();
    request.LookUpStartDate = DateTime.Today.AddDays(-1 * NUMBER_OF_DAYS);
    request.IncludeClosedCheck = true; // get closed checks too
    request. EmployeeObjectNum = 90001;

    // Construct response object
    SimphonyPosApi_CheckResponse response = new SimphonyPosApi_CheckResponse();

    // Call web service method
    mTSApi.GetChecks(request, ref response);
    if (response.OperationalResult.Success)
    {
        foreach (SimphonyPosApi_CheckSummaryEx check in response.Checks)
        {
            Console.WriteLine("Check Number:" + check.CheckNum);
            Console.WriteLine("Check Sequence Number:" + check.CheckSeq);

            Console.WriteLine("KDS Order Status Code:" + check.LastKnownKdsOrderStatus);
        }
    }
    else
    {
        Console.WriteLine("GetChecks failed. Error Code: {0}, Error Message: {1}",
             response.OperationalResult.ErrorCode,
             response.OperationalResult.ErrorMessage);
    }
}
```

API Response

If operation succeeds then OperationalResult.Success field will hold “true” and summary of filtered Guest Checks will be populated in `response.Checks` 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 Check Detail

If user wants to see complete details of a guest check or most recent status of the order then following method can be used.

GetCheckDetail method signature

```
void   GetCheckDetail
(
        SimphonyPosApi_CheckDetailRequest  ppCheckDetailFilter,
        SimphonyPosApi_CheckDetailResponse  ppChecksDetailResponse
)
```

Code snippet given below demonstrates retrieving complete details of a guest check in XML format and print the most recent status of the order.

```
SimphonyPosAPIWebSoapClient mTSApi = new SimphonyPosAPIWebSoapClient();

public void InvokeGetCheckDetail()
{
    SimphonyPosApi_CheckDetailRequest request = new SimphonyPosApi_CheckDetailRequest();
    request.CheckNumber = 104;
    request.CheckSeqNumber = 123456789;

    SimphonyPosApi_CheckDetailResponse response = new SimphonyPosApi_CheckDetailResponse();

    mTSApi.GetCheckDetail(request, ref response);
    if (response.OperationalResult.Success)
    {
        Console.WriteLine("Check XML:" + response.CheckDetail);

        XmlDocument checkXML = new XmlDocument();
        checkXML.LoadXml(response.CheckDetail);

        XmlNode checkNumber = checkXML.SelectSingleNode("CheckNumber");
        Console.WriteLine("Check Number - " + checkNumber.InnerText);

        XmlNode dueAmount = checkXML.SelectSingleNode("Due");
        Console.WriteLine("Check Due - " + dueAmount.InnerText);

        // Read the most recent status of the order from extensibility data
        string extensibilityAppName = "OIS"; // this name is configured in EMC
        XmlNodeList xnl = checkXML.SelectNodes(string.Format(
              "//extensibility_data[ExtensibilityAppName = \"{0}\"]", extensibilityAppName));

        if (xnl != null && xnl.Count > 0)
        {
            string mostRecentOrderStatus = string.Empty;
            DateTime mostRecentTimeStamp = DateTime.MinValue;

            foreach (XmlNode node in xnl)
            {
                XmlNode stringData = node.SelectSingleNode("StringData");
                // Read the value of 'Timestamp' property from stringData
                DateTime timeStamp = GetOrderTimeStamp(stringData.InnerText);

                if (timeStamp > mostRecentTimeStamp)
```

```
                {
                    mostRecentOrderStatus = node.SelectSingleNode("DisplayName").InnerText;
                    mostRecentTimeStamp = timeStamp;
                }
            }

            Console.WriteLine("Most Recent Order Status - " + mostRecentOrderStatus);
            Console.WriteLine("Timestamp - " + mostRecentTimeStamp);
        }
    }
    else
    {
        Console.WriteLine("GetCheckDetail failed. Error Code: {0}, Error Message: {1}",
            response.OperationalResult.ErrorCode,
            response.OperationalResult.ErrorMessage);
    }
}
```

API Response

If operation succeeds then OperationalResult.Success field will hold “true” and check xml will be populated in `response.CheckDetail` 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 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 (method 1 - GetConfigurationInfo)

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

The client application 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, application version, menu levels, menu item SLU, main menu levels, sub menu levels, event types, event sub types and event definition from Simphony POS database using GetConfigurationInfo method. This method returns all the records of specified configuration data type. The integrator can use another version of this method (explained in the next section) named GetConfigurationInfoEx to retrieve records batch by batch when the volume of configuration data is huge.

GetConfigurationInfo method signature

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

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 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.

Get Configured Information (method 2 - GetConfigurationInfoEx)

GetConfigurationInfoEx is a new version of GetConfigurationInfo method to retrieve configuration data batch by batch or in full. This method is useful when the POS database has a huge volume of configuration data for one or more types like Menu Item Definition, Menu Item Master etc. This method is designed to return all the records for a configuration data type when no ranges are specified in the input parameters. In other words, this method will behave exactly like the old method (GetConfigurationInfo) when no ranges (i.e. start index & maximum records count) are specified.

GetConfigurationInfoEx method signature

```
void GetConfigurationInfoEx
(
        SimphonyPosApi_ConfigInfoRequest        configInfoRequest,
        ref SimphonyPosAPI_ConfigInfoResponse   configInfoResponse
)
```

Following code snippet demonstrates retrieving configured data for menu item definition and menu item price.

```
SimphonyPosAPIWebSoapClient mTSApi = new SimphonyPosAPIWebSoapClient();

string vendorCode = "lzsroioq"; // This can be empty from 2.7MR3 onwards
int employeeObjectNum = 90001; // for authentication
int revenueCenterObjectNum = 3016;

private const int MAX_RECORD_COUNT= 5000; // Indicates the maximum number of records to be retrieved

private static string[] CONFIG_DATA_NODE_NAMES = new string[] { "DbMenuItemMaster",
     "DbMenuItemPrice" };

public void InvokeGetConfigurationInfo()
{
    SimphonyPosApi_ConfigInfoRequest request = new SimphonyPosApi_ConfigInfoRequest();
    request.EmployeeObjectNumber = employeeObjectNum;
    request.RVCObjectNumber = revenueCenterObjectNum;

    List<SimphonyPosApi_ConfigInfo> configTypeList = new List<SimphonyPosApi_ConfigInfo>();

    // Menu Item Master
    SimphonyPosApi_ConfigInfo miMasterType = new SimphonyPosApi_ConfigInfo();
    miMasterType.ConfigurationInfoTypeID = EConfigurationInfoType.MENUITEMMASTERS;
    miMasterType.MaxRecordCount = MAX_RECORD_COUNT;
    miMasterType.StartIndex = 1; // 1 is just an initial value; this will be incremented each
                                 // time GetConfigurationInfoEx method is called.
    configTypeList.Add(miMasterType);

    // Menu Item Price
    SimphonyPosApi_ConfigInfo miPriceType = new SimphonyPosApi_ConfigInfo();
    miPriceType.ConfigurationInfoTypeID = EConfigurationInfoType.MENUITEMPRICE;
    miPriceType.MaxRecordCount = MAX_RECORD_COUNT;
    miPriceType.StartIndex = 1; // 1 is just an initial value; this will be incremented each
                                // time GetConfigurationInfoEx method is called.
```

```
    configTypeList.Add(miPriceType);

    request.ConfigurationInfo = configTypeList.ToArray();
    int miMasterRecordsCount = 0;
    int miPricecRecordsCount = 0;


    while (true) // call GetConfigurationInfoEx method in a loop until it return all
                 // the records of requested configuration data type
    {
       SimphonyPosApi_ConfigInfoResponse response = new SimphonyPosApi_ConfigInfoResponse();

       // Call the TS method and check the response
       mTSApi.GetConfigurationInfoEx(request, ref response);
       if (response.OperationalResult.Success)
       {
          bool bFoundData = false;

          // Menu Item Master
          if (!string.IsNullOrWhiteSpace(response.MenuItemMasters))
          {
             // add it to the output list
             XmlDocument tempXmlDoc = new XmlDocument();
             tempXmlDoc.LoadXml(response.MenuItemMasters);
             XmlNodeList miMasterList =
                      tempXmlDoc.GetElementsByTagName(CONFIG_DATA_NODE_NAMES[0]);

             if (miMasterList != null && miMasterList.Count > 0)
             {
                bFoundData = true;
                miMasterRecordsCount += miMasterList.Count;

                // TODO: Save these records to a file or memory for consolidation
             }
          }

          // Menu Item Price
          if (!string.IsNullOrWhiteSpace(response.MenuItemPrice))
          {
              // add it to the output list
              XmlDocument tempXmlDoc = new XmlDocument();
              tempXmlDoc.LoadXml(response.MenuItemPrice);
              XmlNodeList miPriceList =
                     tempXmlDoc.GetElementsByTagName(CONFIG_DATA_NODE_NAMES[1]);

              if (miPriceList != null && miPriceList.Count > 0)
              {
                  bFoundData = true;
                  miPricecRecordsCount += miPriceList.Count;

                  // TODO: Save these records to a file or memory for consolidation
              }
            }

            if (bFoundData == false)
```

```
            {
               break; // all records are already returned by TS, so come out of the loop now.
            }

            // increment the start index so that GetConfigurationInfoEx can be
            // called again to get next set of records.
            IncrementStartIndex(configTypeList.ToArray(), MAX_RECORD_COUNT);
         }
      }

      System.Console.WriteLine("Operation completed successfully.");
      System.Console.WriteLine("MI Master count - " + miMasterRecordsCount);
      System.Console.WriteLine("MI Price count - " + miPricecRecordsCount);

      System.Console.ReadLine();
}

private static void IncrementStartIndex(SimphonyPosApi_ConfigInfo[] configTypeList,
       int maxRecordCount)
{
    foreach(SimphonyPosApi_ConfigInfo configInfo in configTypeList)
    {
        configInfo.StartIndex += maxRecordCount;
    }
}
```

API Response

If 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.

Simphony Platform Requirements

Simphony Software Version

This release of the Simphony POS API requires Simphony version 2.7 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.

Demo Client for Transaction Services API

Overview

The Demo client is a Windows application that's developed to demonstrate or test the features of TS API. This application is distributed with Simphony install 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 EMC and is running Service Host application to host 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 steps given below to configure and run demo client application:

1. Copy *PosAPIDemoClient* folder from install media to a local folder.
2. Open the config file named POSAPI_WebClient.exe.config with a text editor like Notepad.
3. Modify the "value" key with correct URL of TS web service.
   a. e.g.,
   ```
   <value>http://<<IpAddress>>:8080/EGateway/SimphonyPosApiweb.asmx</value>
   ```
   b. Replace the placeholder <<IPAddress>> above by the IP address of workstation that hosts TS API
4. Save the changes.
5. Launch POSAPI_WebClient.exe.

Demonstration

Calculate Totals of a Transaction

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

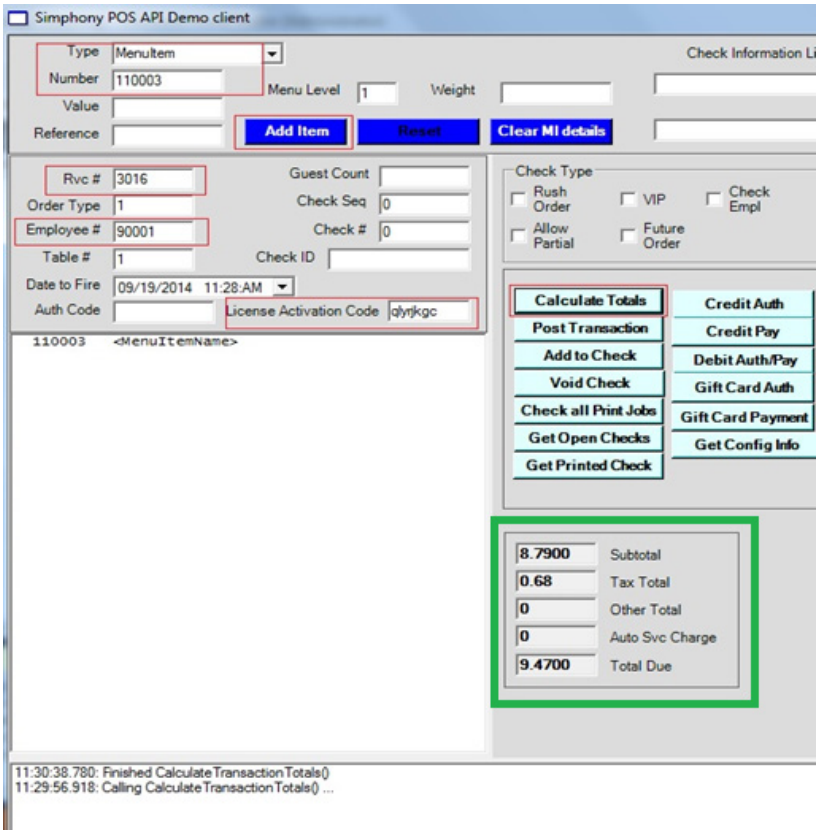
1. Select the **MenuItem** option from the **Type** dropdown field

   Then enter the Menu Item's Object Number '**110003**' in the **Number** field and click **Add Item** button. You can obtain the Menu Item Object Number from EMC, Property level, Configuration tab, Menu Items, Menu Item Maintenance module.
2. Enter 3016 to **RVC #** textbox.

   You can obtain the RVC number from EMC, Property level, Setup tab, Property Configuration, RVC Configuration module.
3. Enter 90001 to **Employee #** textbox

   You can obtain the Employee Number from EMC, Property level, Configuration tab, Personnel, Employee Maintenance module
4. Enter a valid vendor code to the License Activation Code field. This can be left blank for Simphony version 2.7 MR3 or later.
5. Press **Calculate Totals** button to send request to TS API

   The status of operation will display at the bottom of UI while result appears at the block highlighted in green below.



Create a Guest Check

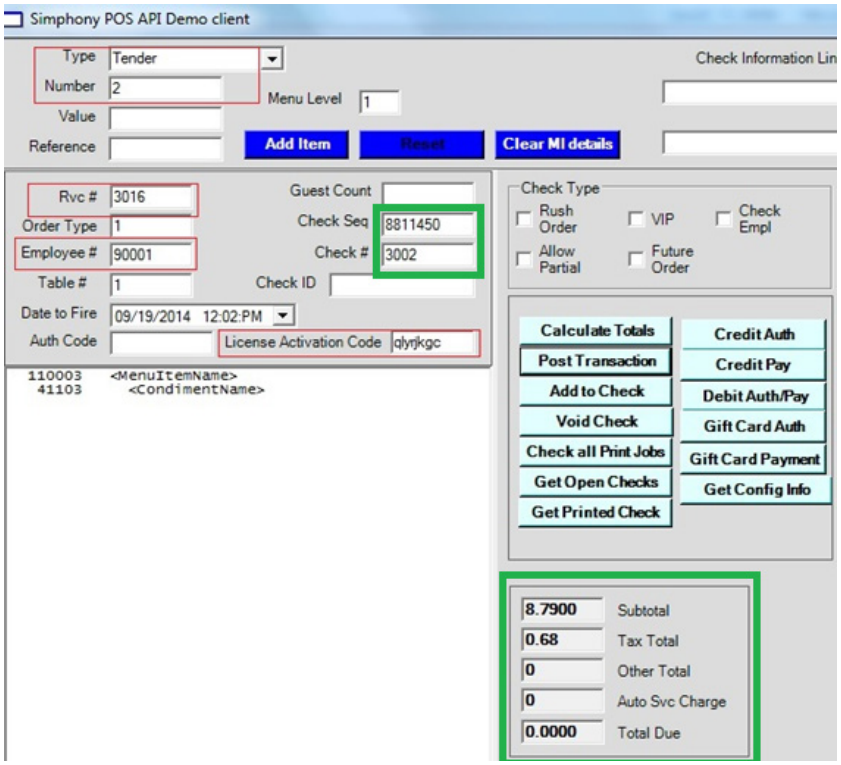
Post Transaction button can used to send a request to create a new Guest Check in the Simphony database.

1. Select **MenuItem** option from the **Type** dropdown field.
2. Enter the Menu Item's Object Number '**110003**' in the **Number** field and click **Add Item** button.
3. Select the **RequiredCondiment** option from **Type** dropdown field.
4. Enter the Object Number '**41103**' in the **Number** textbox and select **Add Item** button.
5. For payment, select **Tender** from **Type** dropdown
    a. For cash payment, type Tender Media Object Number (e.g., 2) of Cash to **Number** field. Then type amount (e.g. 10) to **Value** textbox.
    b. For Debit/Credit payment, type Tender Media Object Number of Credit/Debit and then click **Credit Auth** button to provide payment card details in the popup.

Note: Transaction Services ONLY supports the **MCreditDebit Payment** driver for credit/debit card payment.

---

6. Click **Add Item** button to add Tender details
7. Enter values to the **RVC #**, **Employee #** and **License Activation** Code (pre-Simphony version 2.7 MR3) fields.
8. Click **Post Transaction** button to send request to TS API
9. The result is populated in the fields highlighted in green.

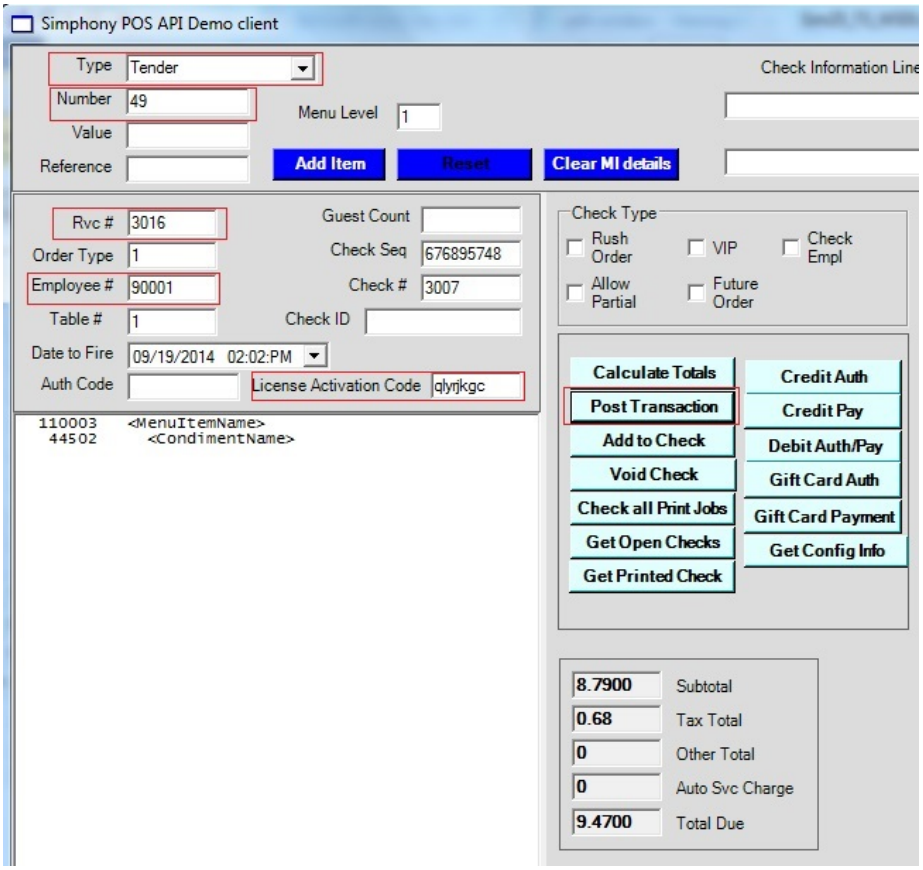


Add an item to an Open Guest Check

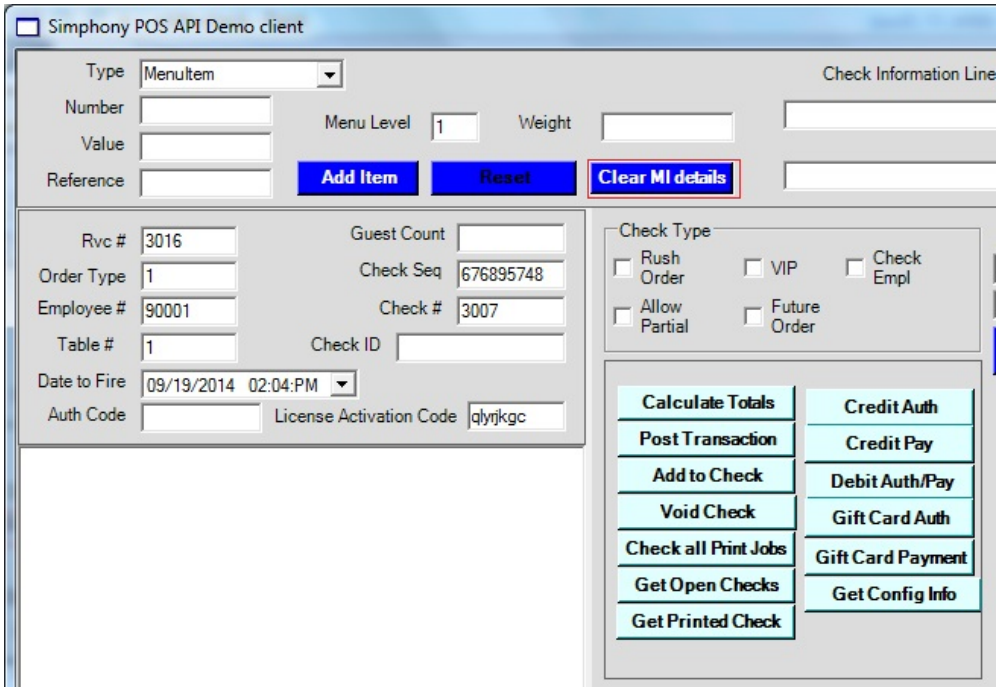
Add to Check button can be used to add one or more items to an existing guest check.

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

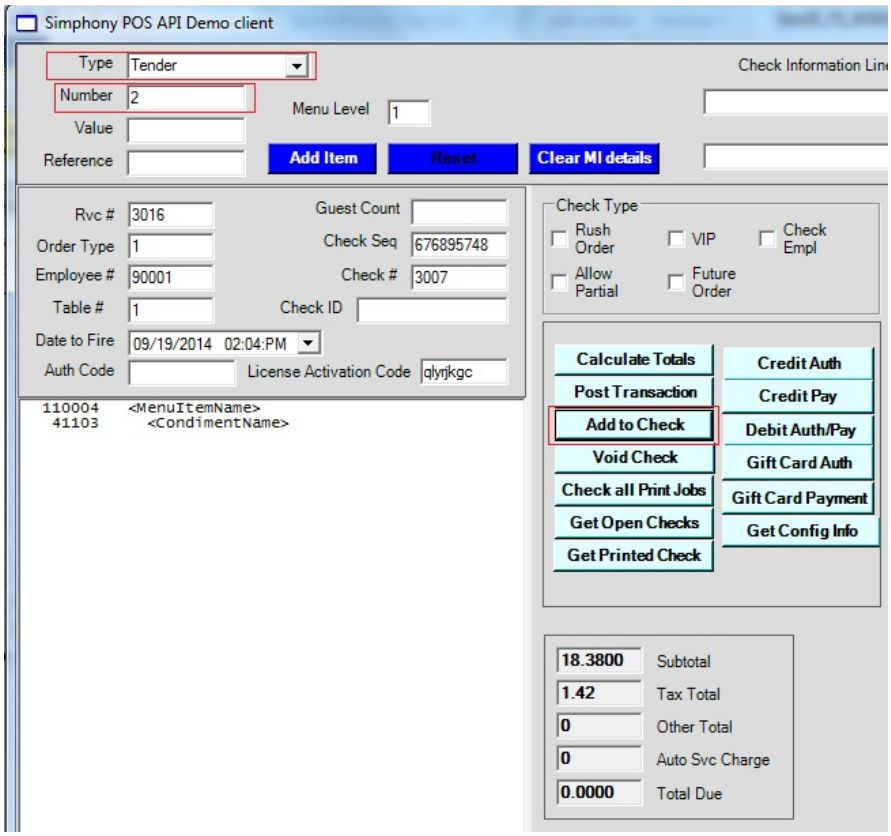
1. Select the **MenuItem** option from the **Type** dropdown field.
2. Enter the Menu Item's Object Number '**110003**' in the **Number** field and click **Add Item** button.
3. Select the **RequiredCondiment** option from the **Type** dropdown field.
4. Enter the Object Number '44502' in the **Number** field and click **Add Item** button.
5. Select the value **Tender** from **Type** dropdown.
6. Enter the Tender Media Object Number of **Service Total** in the **Number** field and then click **Add Item** button.
7. Enter data for **RVC #**, **Employee #** and **License Activation** Code (pre-Simphony version 2.7 MR3)
8. Press the **Post Transaction** button to create the guest check. As the tender type is **Service Total**, the check that's created now will be in open state. Now, one or more items can be added to this open check by following further steps given below.



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



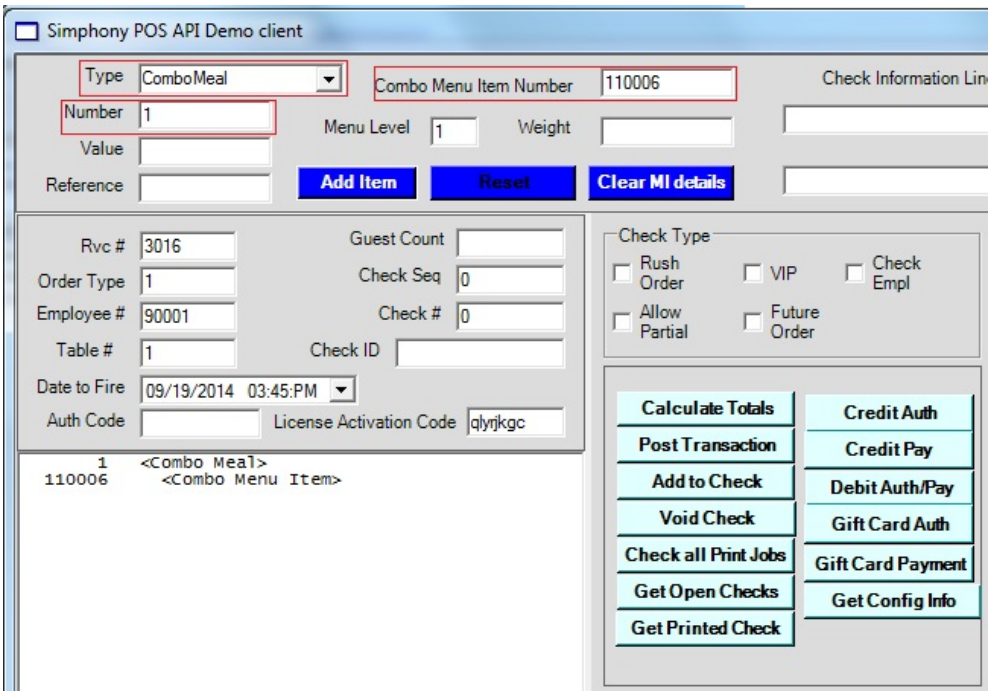
10. Select the **MenuItem** option from the **Type** dropdown field add a menu item to existing guest check.
11. Enter the Menu Item's Object Number **110004** in the **Number** field and then click **Add Item** button.
12. Select **RequiredCondiment** option from the **Type** dropdown field.
13. Enter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



Combo Meal Ordering

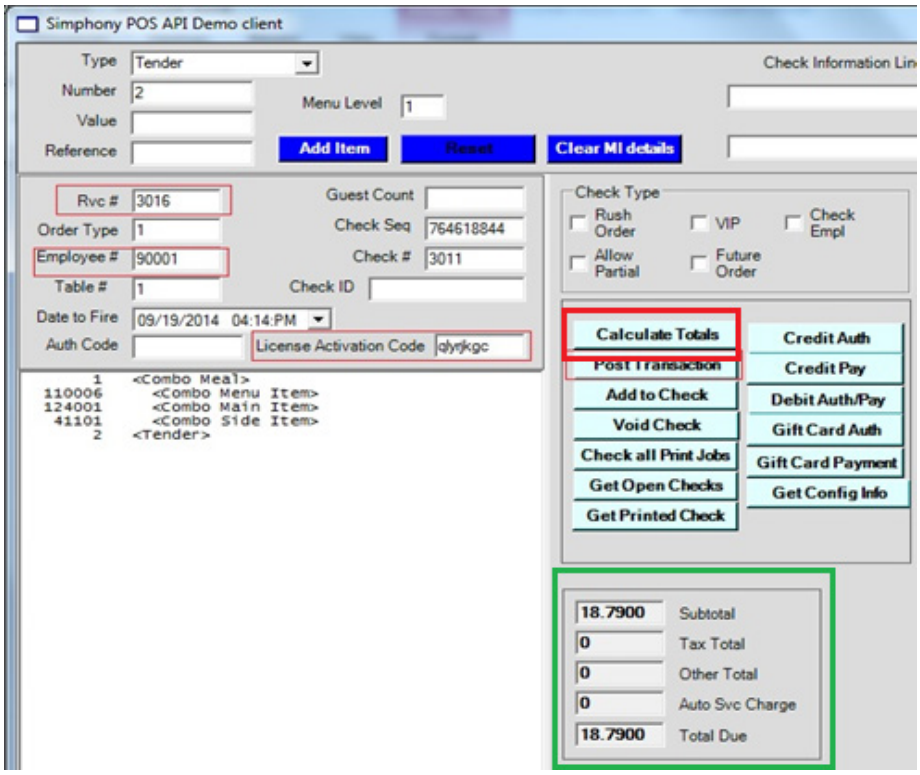
Following steps explains how to add a combo meal to the check. Make sure that a Combo Meal is already configured in the EMC so that it can be rung up on the POS API.

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



2. Then select **ComboMain** from **Type** dropdown to add main item.
3. Enter Combo Meal's Object Number in the Number field and then click **Add Item** button.
4. Select **ComboSide** from **Type** dropdown to add side items.
5. Enter Side Item's Object Number in the **Number** field and select click **Add Item** button.

6. Press **Calculate Totals** button to calculate price of the combo meal, or add a tender and then press “Post Transaction” button to create a guest check.

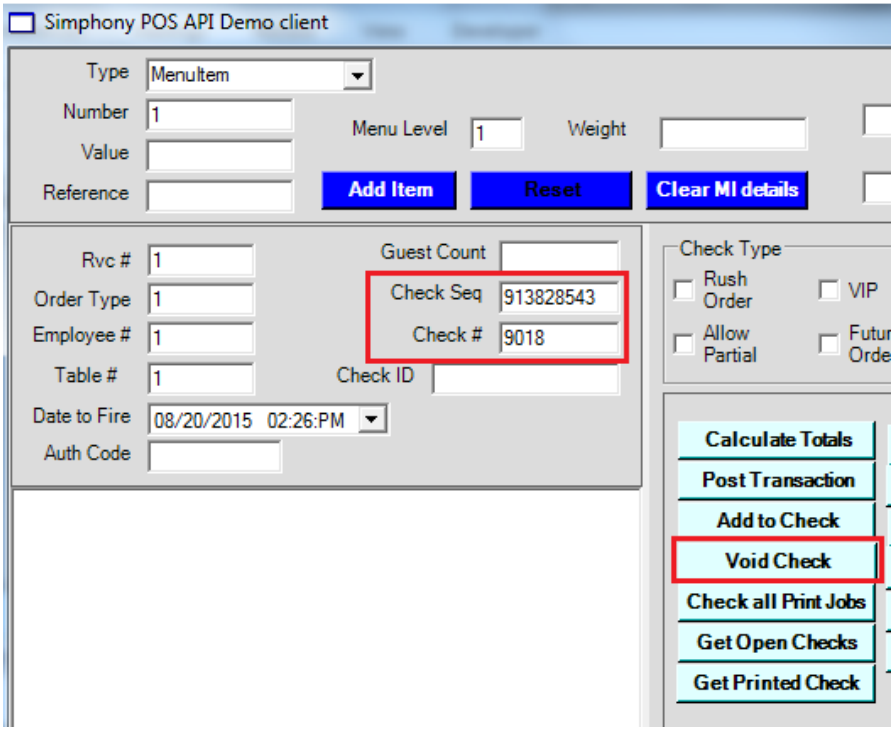


Void All Items of an Open Guest Check

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

7. Enter the **Check Sequence Number** and the **Check Number** of guest check in the relevant fields (highlighted in red) to void all items of that guest check. No other input required to perform this operation.

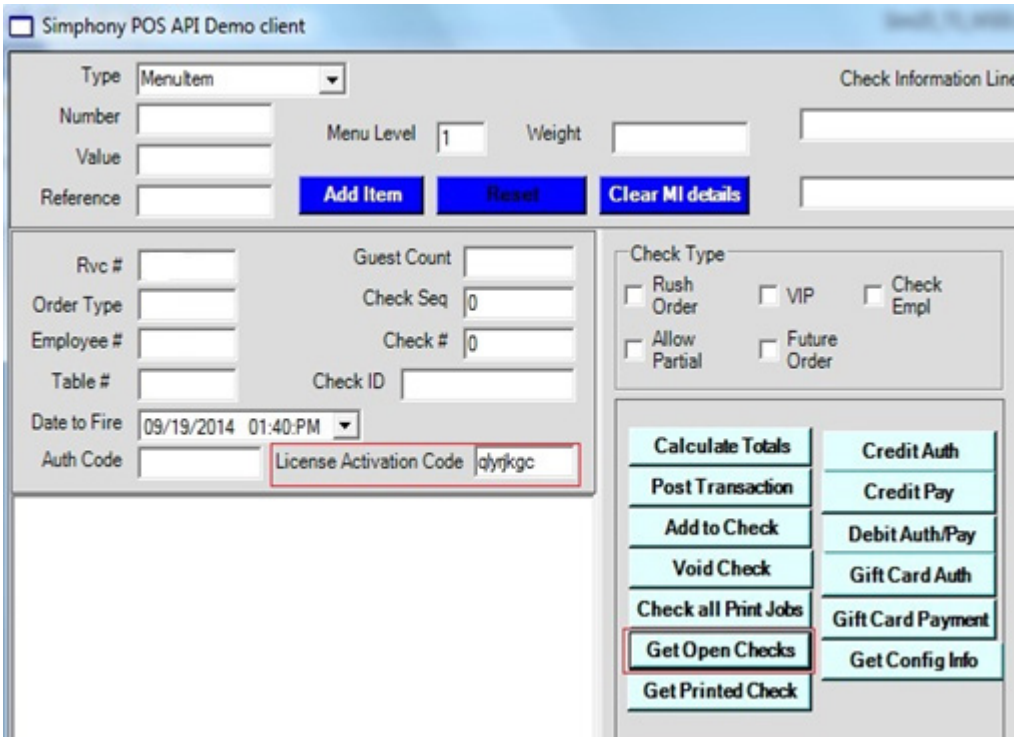
8. Press **Void Check** button to send void request to TS API.



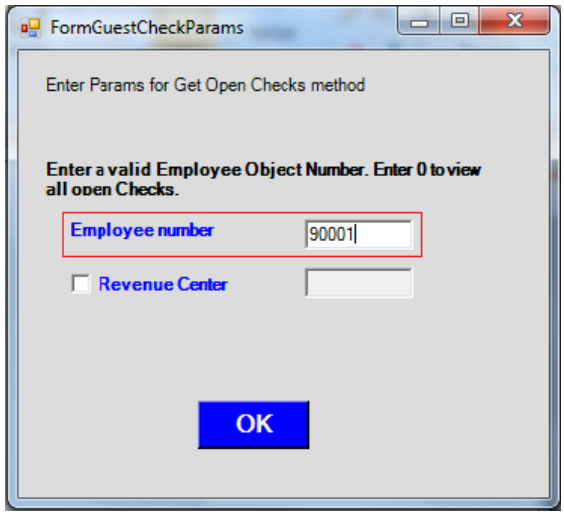
Get Summary of All Open Guest Checks

The **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 TS API based on input given to **Revenue Center** field of *FormGuestCheckParams* dialog, as explained below.

1. Enter the value for the **License Activation Code** field (for pre-Simphony version 2.7 MR3). Then, click the **Get Open Checks** button.



2. The following dialog appears:

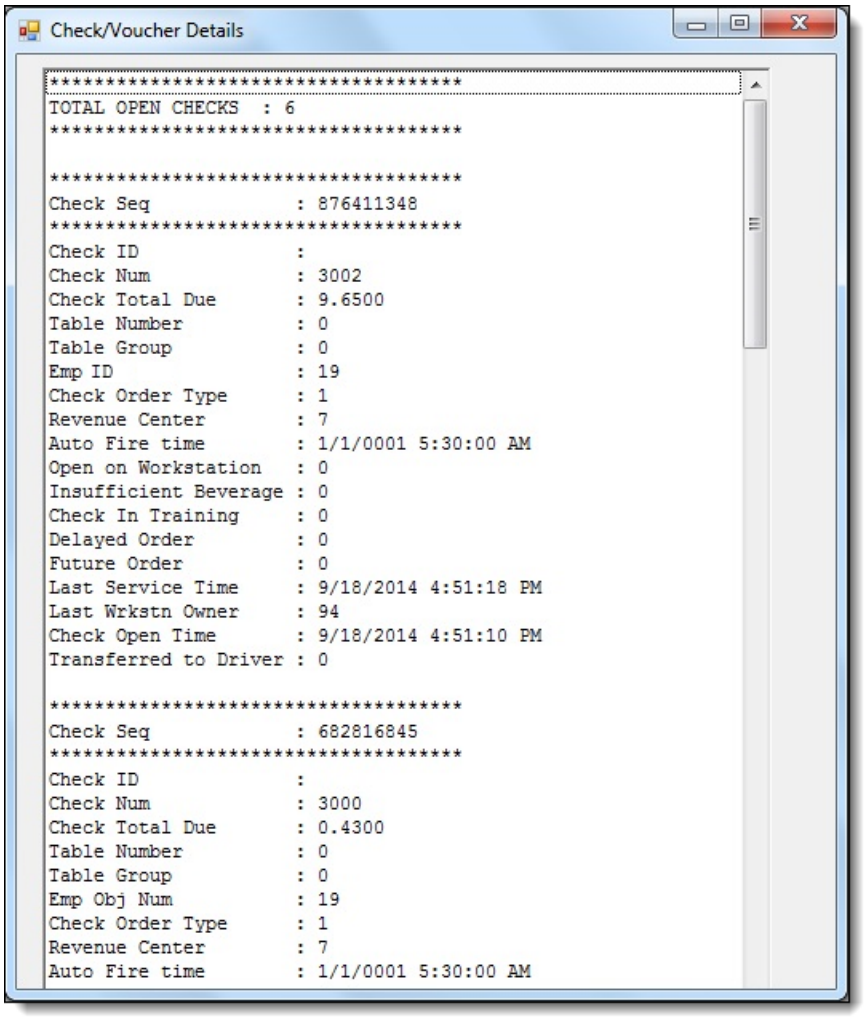


3. Provide **Employee Object Number** in the **Employee number** field to filter checks based on employee (or provide "0" to get all open checks irrespective of who created them) who created it.
4. Then, provide appropriate value for **Revenue Center** field and tick the check box as well:
    a. Provide **-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

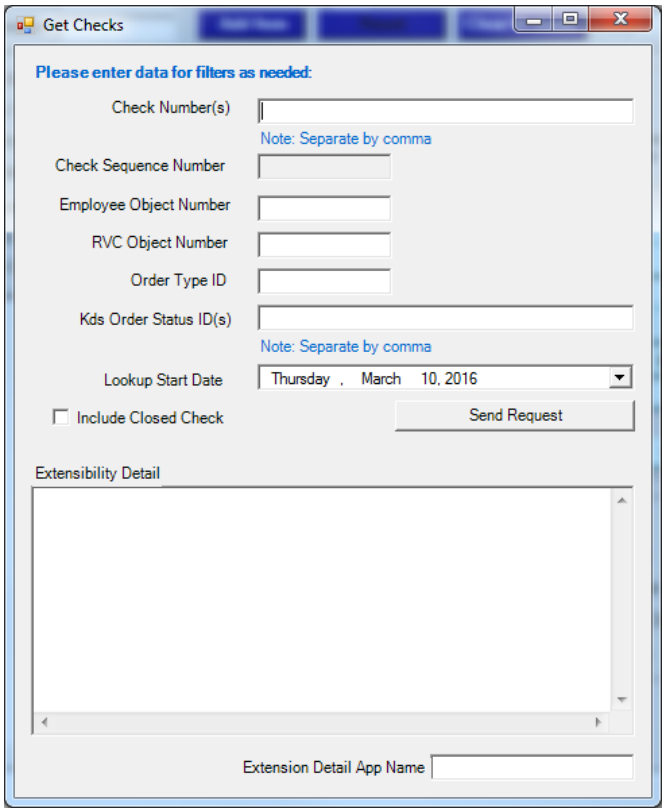
5. Shown below is the output window that shows summary of all open checks.



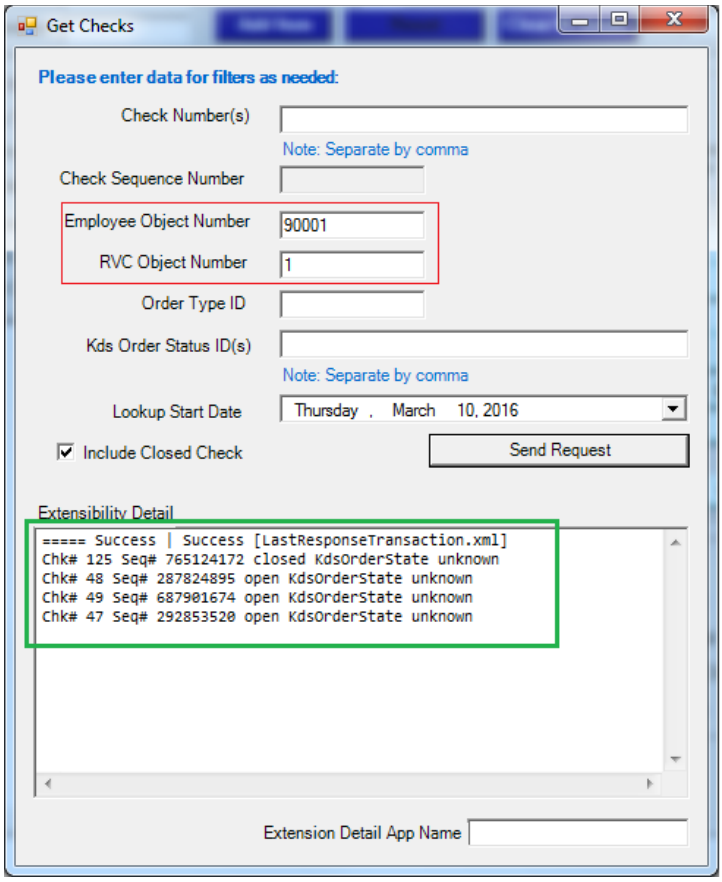
Get Summary of Open and Closed Guest Checks

Get Checks button can be used to send a request to TS API to retrieve summary of all open/closed guest checks from default or a specific revenue center of the property from Simphony POS database. This button calls GetChecks method of TS API.

1. Click the **Get Checks** button from the main window to open the following window.



2. Enter data to one or more filters as needed (e.g. Check Number(s), Employee Object Number, RVC Object Number, Order Type ID, KDS Order Status ID(s), LookupStartDate). Select the **Include Closed Check** checkbox if closed checks need to be retrieved too. Note that the **Check Sequence Numbe**r field is not applicable to this operation, so therefore, it is disabled.
3. Click the **Send Request** button.

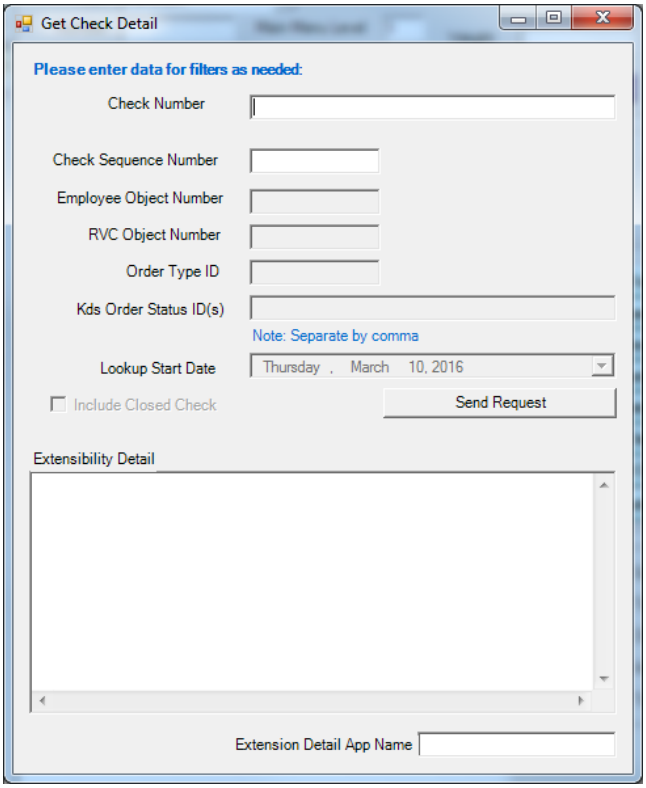


4. The summary of returned checks is displayed in the area highlighted in green above.
5. To see complete details of returned checks, open LastResponseTransaction.xml file that's created by this demo client application under the folder where this application is installed.

Get Check Detail

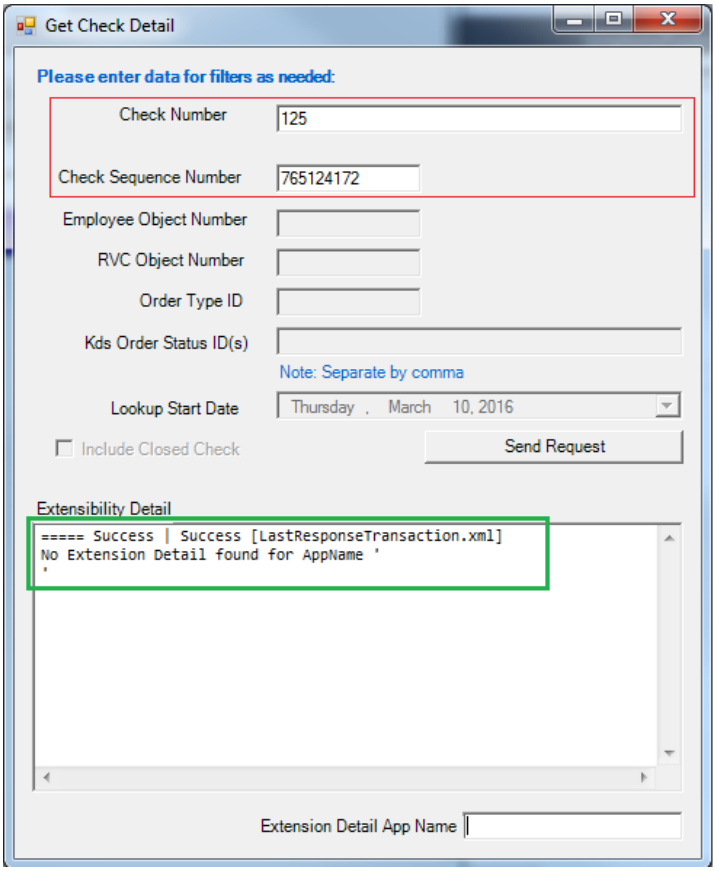
Get Check Detail button can be used to send a request to TS API to retrieve full detail of any open or closed guest check. This button calls GetCheckDetail method of TS API.

1. Click the **Get Check Detail** button from the main window to open following window.



2. Enter data to Check Number and Check Sequence Number fields. Other fields like Employee Object Number, RVC Object Number etc. are not applicable to this operation.
3. Click the **Send Request** button.

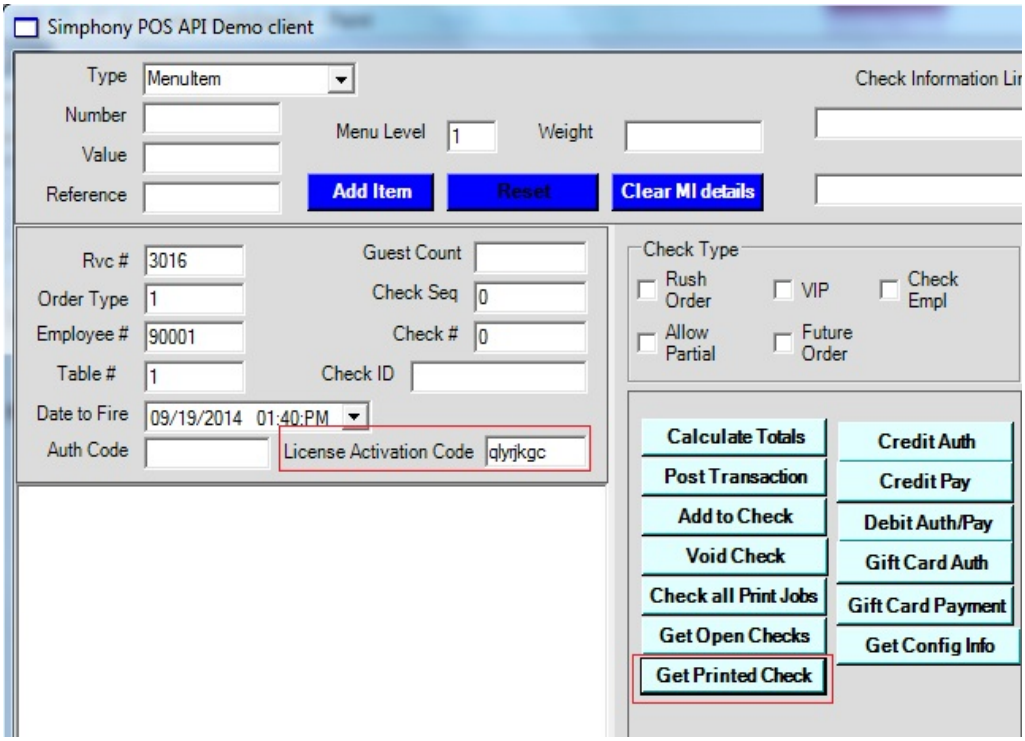
The area highlighted in green will display if the check has any extensibility data. To see the entire check detail, open LastResponseTransaction.xml file that's created by this demo client application under the folder where this application is installed.



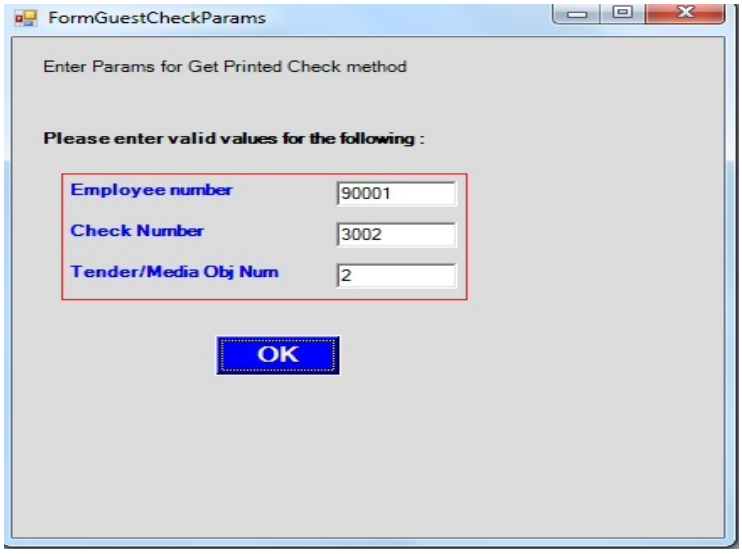
Get Printed Texts of a Guest Check

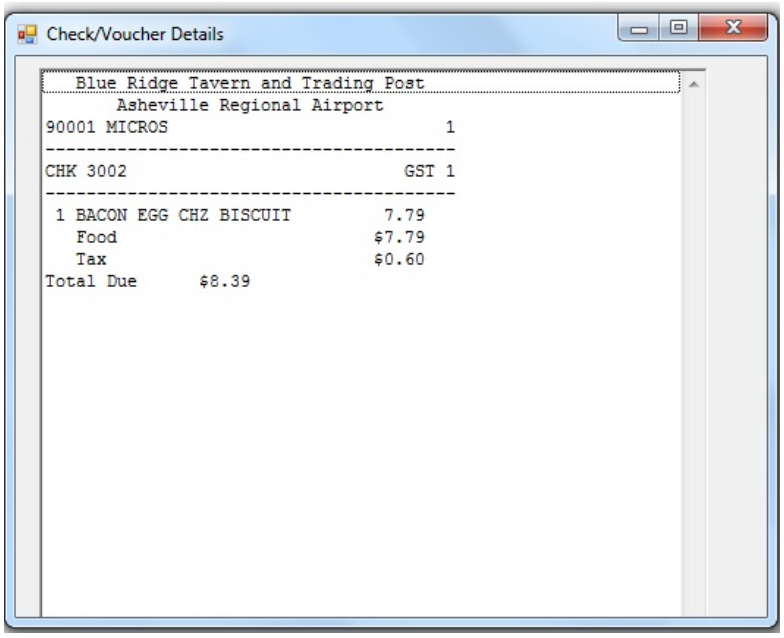
Get Printed Check is used to open posted checks by supplying Check number and the Tender number.

1. Enter the **License Activation Code** (for pre-Simphony version 2.7 MR3).
2. Select the **Get Printed Check** button.



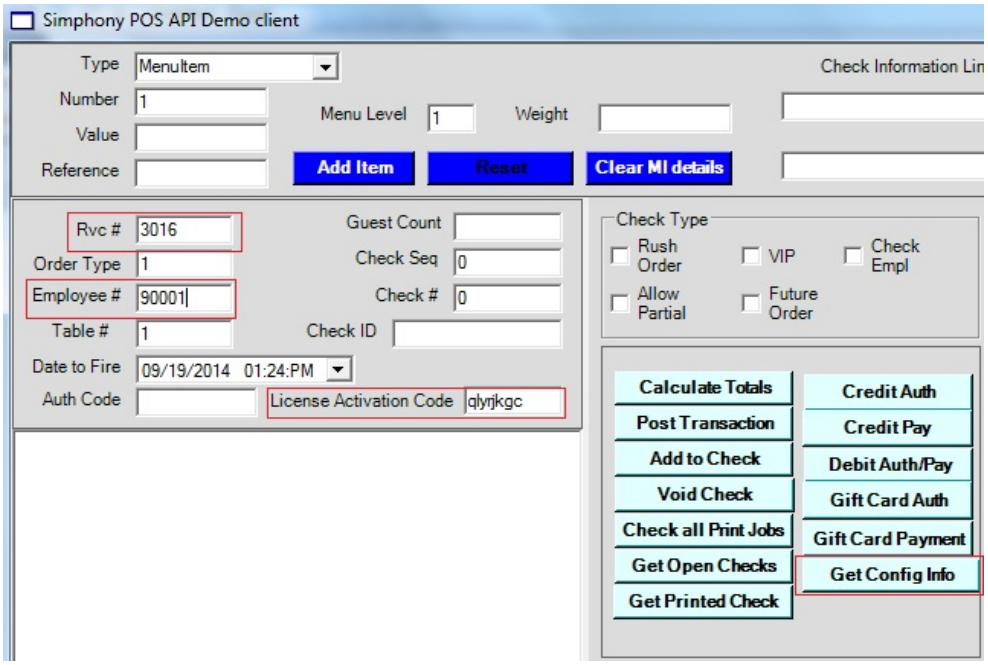
3. Enter an **Employee number**, **Check number** (or **Check Sequence**) and a **Tender Media Object Number** and click **OK**.





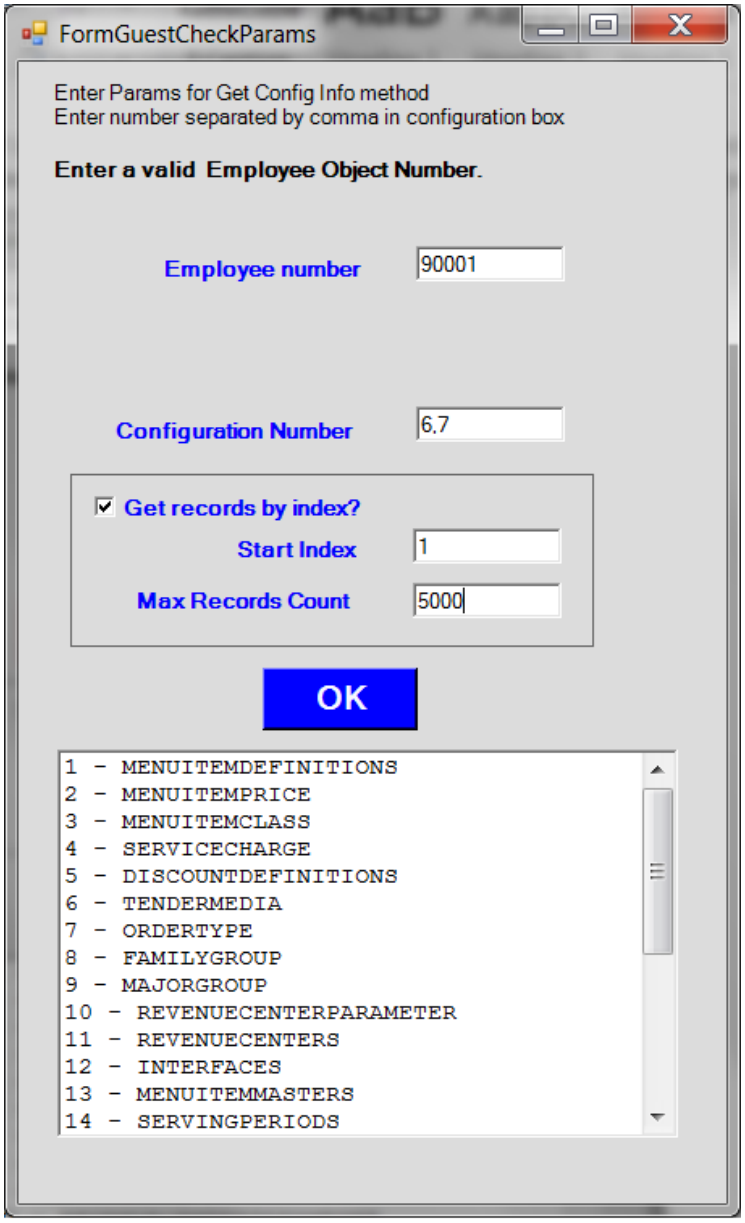
Get Configured Information

1. Enter the RVC Object Number value in the **Rvc #** field.
2. Enter the **License Activation Code**- for pre-Simphony version 2.7 MR3.
3. Click the **Get Config Info** button.

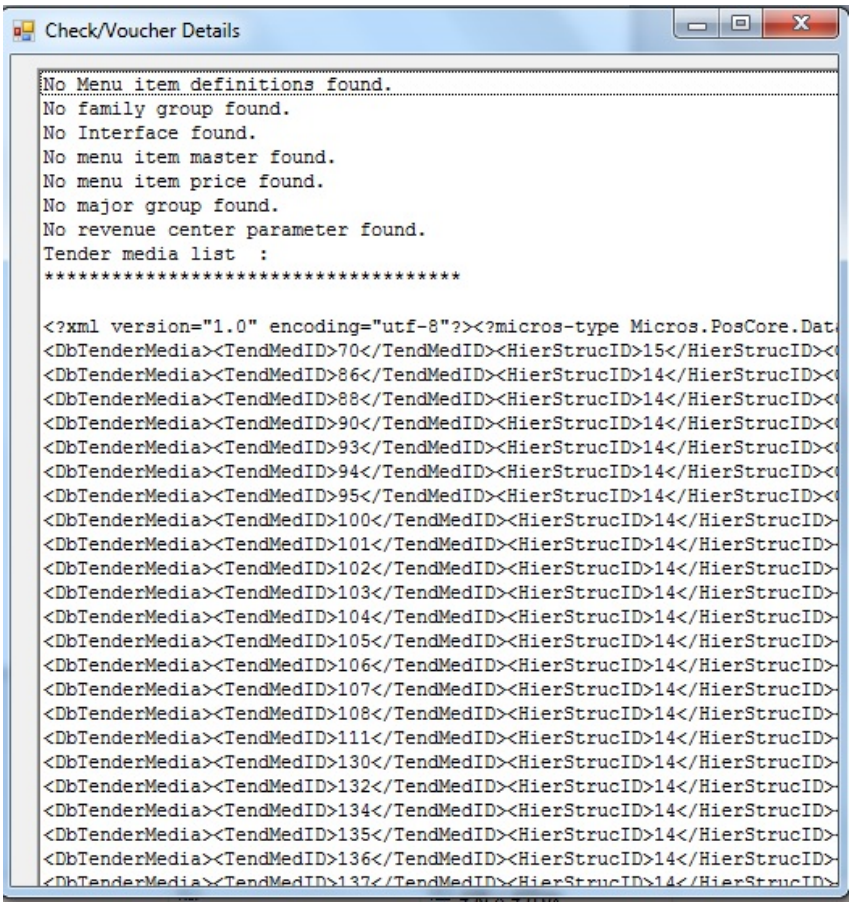


4. Provide an Employee Object Number in the '**Employee number**' field and then enter configuration data type IDs in **'Configuration Number'** textbox. The configuration numbers should be separated by comma when data for more than one configuration data type need to be fetched.

5. If only a specific range of records are required for given configuration data type then select the **Get records by index?** checkbox. Also, specify the start index and maximum number of records required in **Start Index** and **Max Records Count** textboxes. When the checkbox **Get records by index?** is not selected, then the TS API returns all of the records for the specified configuration data types.
6. Click the **OK** button to send request to the TS API.



7. Provided below is the screenshot of output window that shows the configuration data returned by TS API.

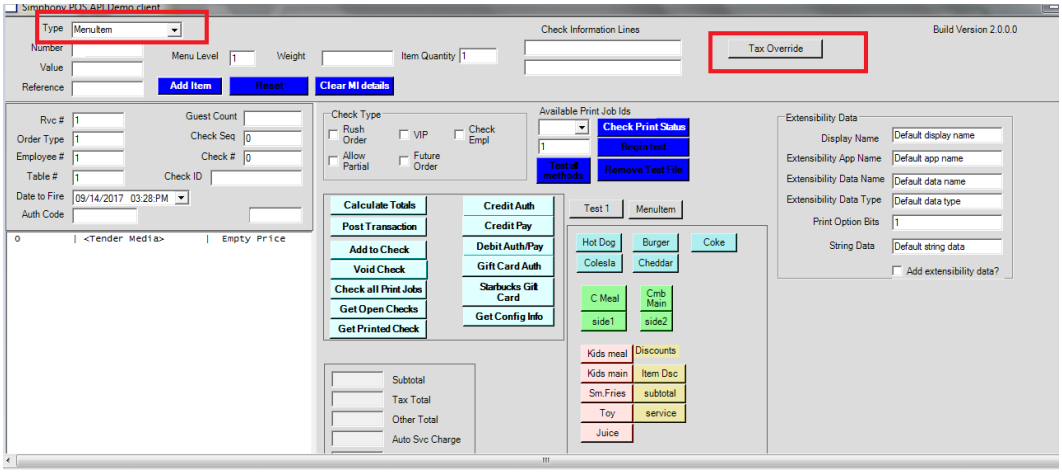


Tax Override

The **"Tax Override"** button can be used to "override" a menu items existing tax rate. These "items" include **Menu Items, Combo Meal, Combo Sides, Combo Main, Required**

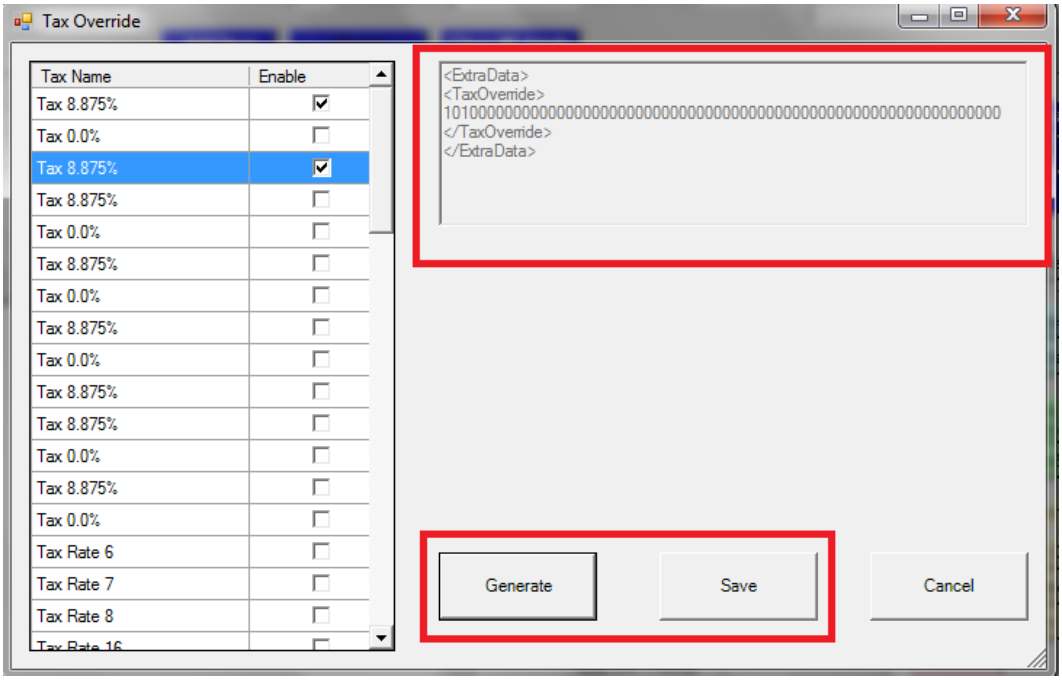
Condiment, and Allowed Condiments. It is assumed that there are one or more tax rates preconfigured in EMC at the Entries or Property level.

1. After adding a “Menu Item” to the check, select the “Tax Override” button.

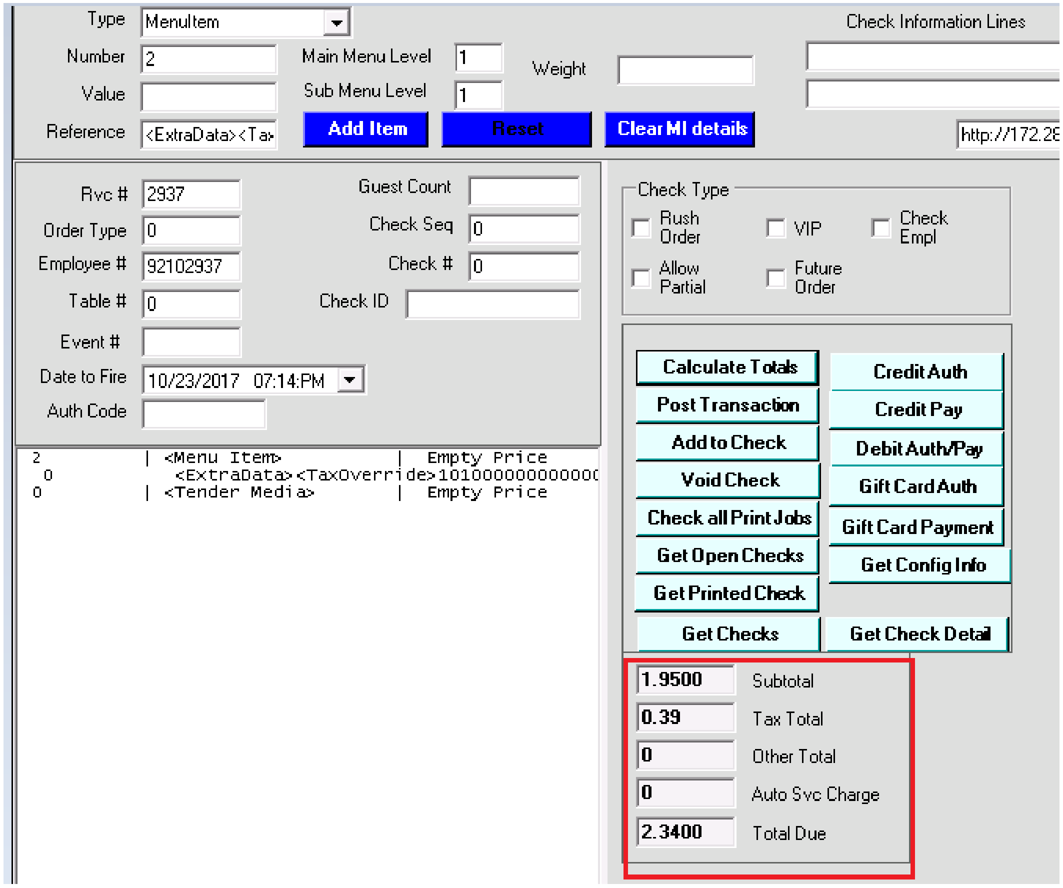


2. Place a checkmark next to the tax rate that you want to apply to this “item”.
3. Press the “”Generate” button and then “Save”.

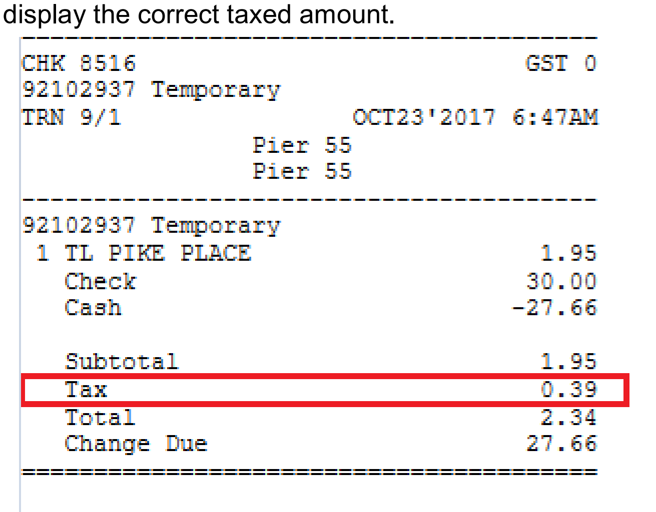
   In the image below, 64 zeros are displayed between “<TaxOverride></TaxOverride>. Each digit represents one of the programmable 64 tax rates in EMC. A number “1” indicates that the particular tax rate record is enabled.



4. Pressing “Calculate Totals” will apply the new rate to the item and added the “taxable amount to the “Tax Total” field.



5. Finally, the user should be able to "Tender" the check and the check printout should display the correct taxed amount.



```
----------------------------------------
CHK 8516                           GST 0
92102937 Temporary
TRN 9/1                OCT23'2017 6:47AM
                Pier 55
                Pier 55

----------------------------------------
92102937 Temporary
 1 TL PIKE PLACE                    1.95
   Check                           30.00
   Cash                           -27.66

   Subtotal                         1.95
   Tax                              0.39
   Total                            2.34
   Change Due                      27.66
========================================
```