## Contents

<table>
<thead>
<tr>
<th>Preface</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1 Introduction</strong></td>
<td>1-1</td>
</tr>
<tr>
<td>Simphony Architecture</td>
<td>1-1</td>
</tr>
<tr>
<td>Hosting Method</td>
<td>1-3</td>
</tr>
<tr>
<td>Quick Installation Roadmap</td>
<td>1-5</td>
</tr>
<tr>
<td>Error Logging</td>
<td>1-6</td>
</tr>
<tr>
<td>TS API Class Hierarchy</td>
<td>1-6</td>
</tr>
<tr>
<td><strong>2 Transaction Operations</strong></td>
<td>2-1</td>
</tr>
<tr>
<td>Calculate Totals</td>
<td>2-1</td>
</tr>
<tr>
<td>Post Transaction</td>
<td>2-2</td>
</tr>
<tr>
<td>Add to Transaction</td>
<td>2-4</td>
</tr>
<tr>
<td>Void Transaction</td>
<td>2-6</td>
</tr>
<tr>
<td>Check Print Job Status</td>
<td>2-6</td>
</tr>
<tr>
<td><strong>3 Guest Check Operations</strong></td>
<td>3-1</td>
</tr>
<tr>
<td>Get Check Summary</td>
<td>3-1</td>
</tr>
<tr>
<td>Get Check Detail</td>
<td>3-1</td>
</tr>
<tr>
<td>Get Printed Check</td>
<td>3-2</td>
</tr>
<tr>
<td><strong>4 Configuration Operations</strong></td>
<td>4-1</td>
</tr>
<tr>
<td>Get Configuration Information</td>
<td>4-1</td>
</tr>
<tr>
<td><strong>5 Structure Reference</strong></td>
<td>5-1</td>
</tr>
<tr>
<td>SimphonyPosAPI_CheckSummary</td>
<td>5-1</td>
</tr>
<tr>
<td>SimphonyPosAPI_CheckSummaryEx</td>
<td>5-2</td>
</tr>
<tr>
<td>SimphonyPosAPI_OpenChecks</td>
<td>5-3</td>
</tr>
<tr>
<td>SimphonyPosAPI_GuestCheck</td>
<td>5-3</td>
</tr>
<tr>
<td>SimphonyPosAPI_CheckRequest</td>
<td>5-4</td>
</tr>
<tr>
<td>SimphonyPosAPI_CheckResponse</td>
<td>5-5</td>
</tr>
<tr>
<td>SimphonyPosAPI_CheckDetailRequest</td>
<td>5-5</td>
</tr>
<tr>
<td>SimphonyPosAPI_CheckDetailResponse</td>
<td>5-5</td>
</tr>
<tr>
<td>SimphonyPosAPI_MenuItem</td>
<td>5-6</td>
</tr>
<tr>
<td>SimphonyPosAPI_MenuItemEx</td>
<td>5-6</td>
</tr>
<tr>
<td>SimphonyPosAPI_MenuItemDefinition</td>
<td>5-6</td>
</tr>
<tr>
<td>SimphonyPosAPI_MenuItemDefinitionEx</td>
<td>5-8</td>
</tr>
<tr>
<td>SimphonyPosAPI_ComboMeal</td>
<td>5-9</td>
</tr>
<tr>
<td>SimphonyPosAPI_ComboMealEx</td>
<td>5-10</td>
</tr>
<tr>
<td>SimphonyPosAPI_Discount</td>
<td>5-10</td>
</tr>
<tr>
<td>SimphonyPosAPI_DiscountEx</td>
<td>5-10</td>
</tr>
</tbody>
</table>
6 Example and Code Snippets

Calculate Totals of a Transaction 6-1
Create a Guest Check 6-12
Add an Item to an Open Guest Check 6-17
Void All Items of an Open Guest Check 6-19
Get Status of a Print Job 6-20
Get Summary of All Open Guest Checks 6-21
Get Open Guest Checks with RVC Object Number 6-22
Get Open Guest Checks from a Specific RVC 6-23
Get Summary and KDS Order Status of Open and Closed Guest Checks 6-25
Get Check Detail 6-26
Get Printed Texts of a Guest Check 6-27
Get Configured Information (Method 1 - GetConfigurationInfo) 6-29
Get Configured Information (Method 2 - GetConfigurationInfoEx) 6-30

7 Symphony Platform Requirements

Symphony Software Version 7-1
Offline Transaction Support 7-1
Printing Services 7-1
Calling Conventions 7-1

8 Demo Client for Transaction Services API

Application Path 8-1
Prerequisites 8-1
Initial Setup 8-1
Demonstration 8-2
Preface

This document is intended for use by software engineers developing applications that interface with Simphony using Transaction Services (TS) API.

Purpose
Introduction to the document.

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

<table>
<thead>
<tr>
<th>Acronym/Abbreviation</th>
<th>Full Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>CAL</td>
<td>Client Application Loader</td>
</tr>
<tr>
<td>CAPS</td>
<td>Check and Posting Service</td>
</tr>
<tr>
<td>DB</td>
<td>Database</td>
</tr>
<tr>
<td>EMC</td>
<td>Enterprise Management Console</td>
</tr>
<tr>
<td>IIS</td>
<td>Microsoft Internet Information Services</td>
</tr>
<tr>
<td>OPS</td>
<td>Operations Software (POS Client)</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>PDA</td>
<td>Personal Digital Assistance</td>
</tr>
<tr>
<td>POS</td>
<td>Point of Sale</td>
</tr>
</tbody>
</table>
**Preface**

<table>
<thead>
<tr>
<th>Acronym/Abbreviation</th>
<th>Full Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVC</td>
<td>Revenue Center</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>SVC</td>
<td>Stored Value Card</td>
</tr>
<tr>
<td>TS</td>
<td>Transaction Services</td>
</tr>
</tbody>
</table>

**Customer Support**

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

https://support.oracle.com

When contacting Customer Support, please provide the following:

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

**Documentation**

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

**Revision History**

<table>
<thead>
<tr>
<th>Date</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>December 2018</td>
<td>Initial publication</td>
</tr>
</tbody>
</table>
Introduction

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

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

All guest checks posted via Transaction Services API can be opened with the POS client user interface, which is configured to run on any workstation from the same property.

The following are sample business scenarios in which Transaction Services API can be used to integrate a third-party application with the Simphony POS.

- 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 the Simphony CAL package is executed. The CAL package downloads the required file contents from the Simphony application server to the actual workstation and installs it. As part of the installation, the CAL package creates a shortcut to ServiceHost.exe on the desktop. Launching ServiceHost.exe ensures that the TS web service is hosted (along with other services and the POS client user interface), and is ready to process requests from any client.

Simphony Architecture

As mentioned in the previous section, the Transaction Services API is a web service hosted by ServiceHost.exe that is designed to run on each POS workstation of the Simphony system. The Service Host application is a custom built web server by Oracle that hosts various services, such as transaction services, print controller, cash management, check and posting, and the POS client user interface. These services are required on a workstation for end-to-end POS operations. The user can remove the POS client user interface from Service Host (using the Simphony EMC application) if only services are required to be hosted for third-party integration.
The POS client provides a user interface for all POS operations in the Simphony system. The order taker can log on to the POS client using valid credentials and add a transaction to create a guest check. As part of a transaction, he or she can add menu items, discounts, service charges, multiple tax rates, and tender details to the guest check and save it to the POS database. The guest checks that are created on one workstation of the property can be accessed from other workstations of the same property using the CAPS (Check and Posting) service.

TS web service uses the same underlying business libraries that are used by the POS client for all POS operations. Technically TS web service is a version of the POS client without the UI.

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

All configuration changes made by the EMC application at the enterprise server are downloaded to the local POS database of each workstation using a DB Sync component. By default, the DB Sync operation runs every 30 minutes. Thus, any configuration changes made at the enterprise server for a specific property or workstation will be available for Transaction Services and the POS client within 30 minutes. The DB Sync can also be manually initiated anytime using a function button in the POS client.

The following image shows the Simphony system architecture, including Transaction Services API/web service. The image depicts how data flows from a TS client application to the reporting database.
Direct Posting Services is a Windows service that runs on the application server to upload data from the transaction database (MCRSPOS) to one of the reporting databases 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 the EMC application will allow for the successful hosting of Transaction Services web service at the POS workstation.

The Service Host, which is 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 the 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 by referring to the correct URL of the TS web service.

To configure a POS API client for the TS web service:

1. Log on to the EMC application.
2. Select a property to configure.
3. Click the Setup tab.
4. Click the Workstation link under Hardware/Interfaces.
5. Insert a new workstation as a POS API client (for example, TS API).
6. Double-click the record to open the created workstation in form view.
7. On the General tab under the General Settings section, select 3 - POSAPI Client from the Type drop-down list. This enables the workstation to run only TS inside the Service Host process without the POS client.
8. If the POS client needs to run on the same workstation that is run by a Service Host, you need to create two workstations: one for the POS workstation client and another for the TS client type. From the Service Host link, select the Service Host of the POS client when the drop-down list appears during Pos Api client WS type selection. This ensures that there will be two workstation identities—one WorkstationID for the POS client and another API_Workstation ID for the TS workstation.
9. Click Save.

To confirm that the TS web service is hosted correctly:
1. Launch ServiceHost.exe on the POS workstation.
2. Go to the web service URL above using any web browser. If WSDL details are shown in the web browser, the web service is hosted properly.

Apart from this method (that is, hosting TS web service on Service Host), TS web service can be hosted on the application server as well, but it will never be applied because the application server is beyond the boundary of third-party clients. By default, the Simphony application server has the TS web service installed and hosted after successful execution of Simphony installation media. This web service can be accessed with the right URL to the application server. It is normally hosted on the following URL. Replace the placeholder with the IP address of the application server below.


To create a stub or proxy or WSDL for the client application to integrate with the 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 is hosted on the workstation.
Quick Installation Roadmap

1. Log on to EMC and select a property that needs to be configured

2. Click Setup tab, click Workstation link under Hardware/Interfaces

3. Add a new workstation for POS API client, on General tab select 3 – POSAPI Client for Type

   - If POS client UI does not need to be hosted by Service Host, click Save, to make TS without UI
   - If POS client UI is needed, select the Service Host ID of corresponding POS client from Service Host ID drop-down list of Service Host fields section that appears after changing type of WS to POS API Client

4. Let CAL download ServiceHost.xml, then go to Web.config.txt located under MICROS > Symphony > EGatewayService in Workstation to confirm the type change made through as new entry api_workstationID is displayed

5. Launch ServiceHost.exe and then navigate to the web service URL given below using a web browser to see if TS API hosted properly

Error Logging

Transaction Services API writes all errors and other informational messages to a flat file in the following folder on the POS workstation for diagnostics purposes. “Simphony Pos Api:” is appended before any logs in TS.

`<ROOT_INSTALL_DRIVE>\MICROS\Simphony\WebServer\wwwroot\EGateway\EgatewyLog\`

Example path: C:\MICROS\Simphony\WebServer\wwwroot\EGateway\EgatewyLog\`

TS API Class Hierarchy

Table 1-1 - Interfaces Table

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITransactionServices Interface</td>
<td>The main interface that supports transaction related operations (for example, 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).</td>
</tr>
<tr>
<td>IGetCheckInfo Interface</td>
<td>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.</td>
</tr>
<tr>
<td>SimphonyPosAPI_CheckSummary Structure</td>
<td>Structure that defines summary of a guest check.</td>
</tr>
<tr>
<td>SimphonyPosApi_CheckSummaryEx Structure</td>
<td>This structure derives from SimphonyPosAPI_CheckSummary and holds a couple of fields on KDS order status.</td>
</tr>
<tr>
<td>SimphonyPosAPI_OpenChecks Structure</td>
<td>Structure that represents check summary of all open checks.</td>
</tr>
<tr>
<td>SimphonyPosAPI_GuestCheck Structure</td>
<td>Structure that defines guest check details such as Check ID, Order Type, Guest Count, Table Number, and so on.</td>
</tr>
<tr>
<td>SimphonyPosApi_CheckRequest Structure</td>
<td>Structure that defines input parameters required to call GetChecks web method.</td>
</tr>
<tr>
<td>SimphonyPosApi_CheckResponse Structure</td>
<td>Structure that holds the response of GetChecks web method.</td>
</tr>
<tr>
<td>SimphonyPosApi_CheckDetailRequest Structure</td>
<td>Structure that defines input parameters required to call GetCheckDetail web method.</td>
</tr>
<tr>
<td>Interface</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SimphonyPosApi_CheckDetailResponse Structure</td>
<td>Structure that holds the response of GetCheckDetail web method.</td>
</tr>
<tr>
<td>SimphonyPosAPI_MenuItem Structure</td>
<td>Structure that holds the definition of menu item and its condiments.</td>
</tr>
<tr>
<td>SimphonyPosAPI_MenuItemDefinition Structure</td>
<td>Structure that defines details of a menu item, such as menu item object number, price, discount, and so on.</td>
</tr>
<tr>
<td>SimphonyPosAPI_ComboMeal Structure</td>
<td>Structure that defines a Combo Meal (main and side items).</td>
</tr>
<tr>
<td>SimphonyPosAPI_Discount Structure</td>
<td>Structure used to represent a discount in the Simphony POS system.</td>
</tr>
<tr>
<td>SimphonyPosAPI_SvcCharge Structure</td>
<td>Structure used to represent a Service Charge in the Simphony POS system.</td>
</tr>
<tr>
<td></td>
<td>This has details such as service charge amount or percentage.</td>
</tr>
<tr>
<td>SimphonyPosAPI_EPayment Structure</td>
<td>Structure that defines Advanced Electronic Payment details such as credit card account, tip amount, cash back amount, and so on.</td>
</tr>
<tr>
<td>SimphonyPosAPI_TmedDetailItemEx Structure</td>
<td>Structure to represent a tender media which has mode of payment.</td>
</tr>
<tr>
<td>SimphonyPosAPI_TotalsResponse Structure</td>
<td>Structure that holds various totals, such as subtotal, due amount, tax amount and automatic service charges of a transaction.</td>
</tr>
<tr>
<td>SimphonyPos_Api_ConfigInfoRequest Structure</td>
<td>Structure that defines input parameters required to call GetConfigurationInfoEx web method.</td>
</tr>
<tr>
<td>SimphonyPosApi_ConfigInfo Structure</td>
<td>Structure that holds filter criteria to be used to retrieve configuration data using GetConfigurationInfoEx method.</td>
</tr>
<tr>
<td>SimphonyPosApi_ConfigInfoResponse Structure</td>
<td>Structure that holds configuration details of items, such as menu item definitions, menu item price, currency, discounts, employees, order type, revenue centers, tender media, service charge, and so on.</td>
</tr>
<tr>
<td>SimphonyPosApi_CheckPrintResponse Structure</td>
<td>Structure that holds the response of Get Printed Check method call. This structure holds operation (success/failure) results along with printed check lines.</td>
</tr>
<tr>
<td>SimphonyPrintApi_PrintJobStatus Structure</td>
<td>Structure that holds the response of Get Printed Job Status method call. This structure holds the operation result (success/failure) along with details of print</td>
</tr>
<tr>
<td>Interface</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SimphonyPosAPI_OperationalResult Structure</td>
<td>Structure used to represent result (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.</td>
</tr>
<tr>
<td>jobs and status code and error/success message.</td>
<td></td>
</tr>
</tbody>
</table>
TS API provides several transaction related POS operations. As the parameters required by an operation are updated, the operation name is extended by adding to the operation name. For example, `CalculateTransactionTotals` and `CalculateTransactionTotalsEx` both perform the same operation; however, the newer “Ex” method accepts different parameters to support additional functionality.

Calculate Totals

The calculate totals operation is used to compute totals of an order without creating a guest check in the Simphony POS database.

The following Calculate Totals method is available in Simphony version 2.7 and later.

```csharp
void CalculateTransactionTotals
(   string vendorCode,
    ref SimphonyPosApi_MenuItem[] ppMenuItems,
    ref SimphonyPosApi_ComboMeal[] ppComboMeals,
    ref SimphonyPosApi_SvcCharge pSvcCharge,
    ref SimphonyPosApi_Discount pSubtotalDiscount,
    int revenueCenter,
    short orderType,
    int employeeNumber,
    ref SimphonyPosApi_TotalsResponse pTotalsResponse
)
```

The following Calculate Totals method is available beginning with Simphony version 18.2. This version of the method has updated structures that add support for multiple discounts, the ability to specify menu item quantity and definition sequence, and specify extension data with menu items.

```csharp
void CalculateTransactionTotalsEx
(   ref SimphonyPosApi_MenuItemEx[] ppMenuItemsEx,
    ref SimphonyPosApi_ComboMealEx[] ppComboMealsEx,
    ref SimphonyPosApi_SvcChargeEx pSvcChargeEx,
    ref SimphonyPosApi_DiscountEx[] pSubTotalDiscountEx,
    int revenueCenterObjectNum,
    short orderType,
    int employeeObjectNum,
    int checkGuestcount,
    ref SimphonyPosApi_TotalsResponseEx pTotalsResponseEx
)
```
Chapter 2
Transaction Operations

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vendorCode</td>
<td>Vendor code for license validation. In Simphony version 2.7 MR3 or later, set this parameter to an empty string.</td>
</tr>
<tr>
<td>revenueCenterObjectNum</td>
<td>Object number of given revenue center.</td>
</tr>
<tr>
<td>orderType</td>
<td>Number of the order type for the transaction (for example, Dine-in or Carry-out).</td>
</tr>
<tr>
<td>employeeObjectNum</td>
<td>Object number of employee authorized to perform the operation.</td>
</tr>
<tr>
<td>checkGuestCount</td>
<td>The number of guests on the guest check. This value is used to compute automatic service charges.</td>
</tr>
</tbody>
</table>

The guest check is defined by creating a set of menu items, combo meals, service charge, and discount. Refer to the Structure Reference chapter for details on how they are specified.

Response

The method calculates discounts (subtotal, item, and automatic discounts), service charges and taxes. The results of the calculations are returned in the same structures used in the request parameters and indicate the price of items and the results of applying discounts and service charges.

The pTotalsResponse/Ex parameter contains the success or failure result of operations, results of tax calculations, and summary totals.

Post Transaction

The post transaction operation is used to create a new guest check.

The following post transaction method is available in Simphony version 2.7 and later.

```csharp
void PostTransactionEx
(
    string vendorCode,
    ref SymphonyPosApi_GuestCheck pGuestCheck,
    ref SymphonyPosApi_MenuItem[] ppMenuItems,
    ref SymphonyPosApi_ComboMeal[] ppComboMeals,
    ref SymphonyPosApi_SvcCharge pServiceChg,
    ref SymphonyPosApi_Discount pSubTotalDiscount,
    ref SymphonyPosApi_TmedDetailItemEx pTmedDetail,
    ref SymphonyPosApi_TotalsResponse pTotalsResponse,
    ref string[] ppCheckPrintLines,
    ref string[] ppVoucherOutput
)
```

The following Post Transaction method is available beginning with Simphony version 18.2. This version of the method has updated structures that add support for multiple
discounts, the ability to specify menu item quantity and definition sequence, and specify extension data with menu items.

```c
void PostTransactionEx2
(
    ref SimphonyPosApi_GuestCheck pGuestCheck,
    ref SimphonyPosApi_MenuItemEx[] ppMenuItemsEx,
    ref SimphonyPosApi_ComboMealEx[] ppComboMealsEx,
    ref SimphonyPosApi_SvcChargeEx pSvcChargeEx,
    ref SimphonyPosApi_DiscountEx[] pSubTotalDiscountEx,
    ref SimphonyPosApi_TmedDetailItemEx2[] pTmedDetailEx2,
    ref SimphonyPosApi_TotalsResponseEx pTotalsResponseEx,
    ref string[] ppCheckPrintLines,
    ref string[] ppVoucherOutput,
    SimphonyPosApi_Extensibility[] checkExtensibilityDetails
)
```

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>vendorCode</strong></td>
<td>Vendor code for license validation. In Simphony version 2.7 MR3 or later, set this parameter to an empty string.</td>
</tr>
<tr>
<td><strong>revenueCenterObjectNum</strong></td>
<td>Object number of given revenue center.</td>
</tr>
<tr>
<td><strong>orderType</strong></td>
<td>Number of the order type for the transaction (for example, Dine-in or Carry-out).</td>
</tr>
<tr>
<td><strong>employeeObjectNum</strong></td>
<td>Object number of employee authorized to perform the operation.</td>
</tr>
</tbody>
</table>

The guest check is defined by creating a set of menu items, combo meals, service charges, and discounts. The `pGuestCheck` parameter is used to specify additional attributes for the guest check, Guest Count, Check ID, Event Number, and so on.

A tender media is required. If a tender media of type Service Total is supplied, the guest check is created in the open state. If a tender media of type Payment is supplied, the check is closed.

Refer to the Structure Reference chapter for details on how they are specified.

### Response

The guest check is printed to the local and remote order devices defined for the POS API workstation. The printed check details are returned in the `ppCheckPrintLines` parameter, while credit voucher details are filled in the `ppVoucherOutput` parameter.

The price of menu items, discounts, and service charges are returned in the same structures used to create the request. The `pGuestCheck` parameter contains a summary of totals.

The system does not create the guest check if payment or another interim operation fails. Inspect the `OperationalResult` property of the `pGuestCheck` structure in the response to verify that the operation has completed successfully.
When the operation completes successfully, the CheckNum and CheckSeq properties are set on the `pGuestCheck` parameter. These values are used to reference the guest check in subsequent operations.

## Add to Transaction

This operation is used to add items or payments to a guest check previously created using a post transaction operation. This operation can be used to

- add one or more menu items or combo meals to an existing open guest check
- apply partial or full payment on an existing open guest check
- applied a coupon discount to an existing open guest check

The following post transaction method is available in Simphony version 2.7 and later.

```csharp
void AddToExistingCheckEx
(
    string vendorCode,
    ref SymphonyPosApi_GuestCheck pGuestCheck,
    ref SymphonyPosApi_MenuItem[] ppMenuItems,
    ref SymphonyPosApi_ComboMeal[] ppComboMeals,
    ref SymphonyPosApi_SvcCharge pServiceChg,
    ref SymphonyPosApi_Discount pSubTotalDiscount,
    ref SymphonyPosApi_TmedDetailItemEx[] pTmedDetail,
    ref SymphonyPosApi_TotalsResponse pTotalsResponse,
    ref string[] ppCheckPrintLines,
    ref string[] ppVoucherOutput
)
```

The following post transaction method is available beginning with Simphony version 18.2. This version of the method has updated structures that add support for multiple discounts, the ability to specify menu item quantity and definition sequence, and specify extension data with menu items.

```csharp
void AddToExistingCheckEx2
(
    ref SymphonyPosApi_GuestCheck pGuestCheck,
    ref SymphonyPosApi_MenuItemEx[] ppMenuItemsEx,
    ref SymphonyPosApi_ComboMealEx[] ppComboMealsEx,
    ref SymphonyPosApi_SvcChargeEx pSvcChargeEx,
    ref SymphonyPosApi_DiscountEx[] pSubTotalDiscountEx,
    ref SymphonyPosApi_TmedDetailItemEx2[] pTmedDetailEx2,
    ref SymphonyPosApi_TotalsResponseEx pTotalsResponseEx,
    ref string[] ppCheckPrintLines,
    ref string[] ppVoucherOutput,
    SymphonyPosApi_Extensibility[] checkExtensibilityDetails
)
```

### Parameters

| **vendorCode** | Vendor code for license validation. In Simphony version 2.7 MR3 or later, set this parameter to an empty string. |
**Chapter 2**

**Transaction Operations**

<table>
<thead>
<tr>
<th>revenueCenterObjectNum</th>
<th>Object number of given revenue center.</th>
</tr>
</thead>
<tbody>
<tr>
<td>orderType</td>
<td>Number of the order type for the transaction (for example, Dine-in or Carry-out).</td>
</tr>
<tr>
<td>employeeObjectNum</td>
<td>Object number of employee authorized to perform the operation.</td>
</tr>
</tbody>
</table>

The guest check is defined by creating a set of menu items, combo meals, service charges, and discounts. The `pGuestCheck` parameter is used to specify additional attributes for the guest check, Guest Count, Check ID, Event Number, and so on.

A tender media is required. If a tender media of type Service Total is supplied, the guest check is created in the open state. If a tender media of type Payment is supplied, the check is closed.

When this method is invoked, the guest check structure (`pGuestCheck`) is evaluated and changed where appropriate. The following properties of `SimphonyPosApi_GuestCheck` can be modified and updated to reflect the new information during execution of this method:

- CheckID
- CheckTableObjectNum (when supported)
- CheckOrderType
- CheckEmployeeObjectNum
- CheckDateToFire
- pCheckInfoLines

The `CheckNum`, `CheckSeq` and `CheckRevenueCenterObjectNum` properties are not modified by this method.

Refer to the Structure Reference chapter for details on how they are specified.

**Response**

The price of menu items, discounts, and service charges are returned in the same structures used to create the request. The `pGuestCheck` parameter contains a summary of totals.

The guest check is printed to the local and remote order devices defined for the POS API workstation. The printed check details are returned in the `ppCheckPrintLines` parameter, while credit voucher details are filled in the `ppVoucherOutput` parameter.

The operation does not complete if payment or another interim operation fails. Inspect the `OperationalResult` property of the `pGuestCheck` structure in the response to verify that the operation completed successfully.

When the operation completes successfully, the `CheckNum` and `CheckSeq` properties are set on the `pGuestCheck` parameter. These values are used to reference the guest check in subsequent operations.
Void Transaction

This operation voids all items (for example, menu items, tender media, service charge, and discount) in the given guest check and then closes the check. This method works only if the guest check is in the open state. This method fails if the check is already in the closed state.

The following post transaction method is available in Simphony version 2.7 and later.

```csharp
void VoidTransaction
(
    string vendorCode,
    ref SymphonyPosApi_GuestCheck pGuestCheck
)
```

Parameters

<table>
<thead>
<tr>
<th>vendorCode</th>
<th>Vendor code for license validation. In Simphony version 2.7 MR3 or later, set this parameter to an empty string.</th>
</tr>
</thead>
</table>

The request must specify both the CheckNum and CheckSeq properties of the pGuestCheck parameter. All other properties of pGuestCheck can be set to their default values.

Response

Check the OperationalResult property of the pGuestCheck parameter from the response to verify that the operation succeeded.

Check Print Job Status

This method is used to obtain the status of a specific print job (for example, guest check print and credit voucher print) of a transaction.

The following post transaction method is available in Simphony version 2.7 and later.

```csharp
void CheckPrintJobStatus
(
    string vendorCode,
    int ppJobId,
    ref SymphonyPrintApi_PrintJobStatus ppJobStatus
)
```
Parameters

<table>
<thead>
<tr>
<th><strong>vendorCode</strong></th>
<th>Vendor code for license validation. In Simphony version 2.7 MR3 or later, set this parameter to an empty string.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ppJobId</strong></td>
<td>The ID of a print job for which status is retrieved. This value comes from the PPrintJobIds property of the SimphonyPosApi_GuestCheck structure.</td>
</tr>
<tr>
<td><strong>ppJobStatus</strong></td>
<td>The response contains the status of the print job.</td>
</tr>
</tbody>
</table>

Response

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

- Job Pending
- Job Complete
- Job Aborted
- Job Sent to backup printer
- Job Failed
- Job Not found

The status of the print job is returned in the ppJobStatus parameter.
## Guest Check Operations

TS API provides support for two check related and one configuration related operations. One web method is exposed to support each of these operations. The following table describes the web methods.

### Get Check Summary

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GetOpenChecks</code></td>
<td>Gets summary of all open guest checks from all revenue centers of the property from Simphony POS database.</td>
</tr>
<tr>
<td><code>GetOpenChecksEx</code></td>
<td>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 <code>CheckRevenueCenterObjectNum</code> member with ID of revenue center while this method populates Object Number of revenue center.</td>
</tr>
<tr>
<td><code>GetOpenChecksByRVC</code></td>
<td>Gets summary of open guest checks for a specific revenue center from Simphony POS database</td>
</tr>
<tr>
<td><code>GetChecks</code></td>
<td>Gets summary of both open and closed guest checks after applying given filter condition</td>
</tr>
</tbody>
</table>

### Get Check Detail

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GetCheckDetail</code></td>
<td>Gets completes details of a guest check in xml format</td>
</tr>
</tbody>
</table>
Get Printed Check

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

*Gets printed texts of an open guest check*

The following sections provide details about each operation.

## Get Summary of All Open Guest Checks

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

**Business Purpose**
The user wants to view a summary of all open guest checks from all revenue centers within the property.

**Method Description**
This method gets a summary of all open guest checks from all revenue centers within the property from the POS database. Guest checks that are created by a specific employee can be fetched by passing the appropriate value to the `employeeId` parameter. However, when 0 is passed to `employeeId`, it fetches all open guest checks irrespective who created the check. The `CheckRevenueCenterObjectNum` field of the `openChecks.SimphonyPosApi_CheckSummary` structure is mislabeled; it holds the value of Revenue Center ID instead of Revenue Center Object Number. If this field is expected to hold Object Number of Revenue Center, the new method named `GetOpenChecksEx` can be used instead.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>vendorCode</code></td>
<td>Vendor code for license validation. In Simphony version 2.7 MR3 or later, set this parameter to an empty string.</td>
</tr>
<tr>
<td><code>employeeId</code></td>
<td>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.</td>
</tr>
<tr>
<td><code>openChecks</code></td>
<td>Holds open checks retrieved from the POS database (output parameter).</td>
</tr>
</tbody>
</table>
Return Value
Void. Result is encapsulated in openChecks reference parameter.

Get Open Guest Checks With RVC Object Number

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

Business Purpose
The user wants to view a summary of all open guest checks from all revenue centers within the property, and wants to have the object number of the revenue center (instead of ID) for each guest check. The object number of the revenue center is different from that of ID.

Method Description
This method is another version of the GetOpenChecks method that is described in the previous section. This was introduced in Simphony version 2.7 MR5 to retrieve all open guest checks from all revenue centers within the property. The only difference compared to the GetOpenChecks method is that the CheckRevenueCenterObjectNum property of openChecks.SimphonyPosApi_CheckSummary holds the Revenue Center Object Number instead of the Revenue Center ID. All open guest checks created by a specific employee can be fetched by passing the appropriate value to the employeeObjectNum parameter. However, when 0 is passed to employeeObjectNum, it fetches all open guest checks irrespective of who created them.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vendorCode</td>
<td>Vendor code for license validation. In Simphony version 2.7 MR3 or later, set this parameter to an empty string.</td>
</tr>
<tr>
<td>employeeObjectNum</td>
<td>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.</td>
</tr>
<tr>
<td>openChecks</td>
<td>Holds open checks retrieved from the POS database (output parameter).</td>
</tr>
</tbody>
</table>

Return Value
Void. Result is encapsulated in the openChecks reference parameter.

Get Open Guest Checks From a Specific RVC

```c
void GetOpenChecksByRVC
(
    String vendorCode,
    int employeeObjectNum,
    Int revenueCenterObjectNum
    ref SimphonyPosAPI_OpenChecks openChecks
)
```
Business Purpose
The user wants to view a summary of open guest checks from a specific revenue center.

Method Description
This method was introduced in Simphony version 2.7MR4 to get all open guest checks in a specific revenue center from the Simphony POS database. All open guest checks created by a specific employee in a specific revenue center can be fetched by passing the appropriate value to the employeeObjectNum and revenueCenterObjectNum parameters. However, when 0 is passed for employeeObjectNum, it fetches all open guest checks from the specified revenue center irrespective of who created it. Also, note that the other two related methods named GetOpenChecks and GetOpenChecksEx return a summary of all open checks from all revenue centers within the property.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vendorCode</td>
<td>Vendor code for license validation. In Simphony version 2.7 MR3 or later, set this parameter to an empty string.</td>
</tr>
<tr>
<td>employeeObjectNum</td>
<td>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.</td>
</tr>
<tr>
<td>revenueCenterObjectNum</td>
<td>Object number of revenue center for which checks needs to be retrieved.</td>
</tr>
<tr>
<td>openChecks</td>
<td>Holds open checks retrieved from the POS database (output parameter).</td>
</tr>
</tbody>
</table>

Return Value
Void. Result is encapsulated in the openChecks reference parameter.

Get Printed Texts of a Guest Check

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

Business Purpose
The 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 works only on open guest checks and throws an exception for a 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.
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>vendorCode</strong></td>
<td>Vendor code for license validation. In Simphony version 2.7 MR3 or later, set this parameter to an empty string.</td>
</tr>
<tr>
<td><strong>CheckSeq</strong></td>
<td>Check number of the guest check (and not the check sequence number as the name implies).</td>
</tr>
<tr>
<td><strong>EmplObjectNum</strong></td>
<td>Object number of the employee who wants to perform this operation.</td>
</tr>
<tr>
<td><strong>TmedObjectNum</strong></td>
<td>Object number of the tender media to print the check with.</td>
</tr>
<tr>
<td><strong>ppCheckPrintLines</strong></td>
<td>This holds an array of printed lines of the check along with response code.</td>
</tr>
</tbody>
</table>

Return Value
Void. Result is encapsulated in the `ppCheckPrintLines` reference parameter.

Get Summary and KDS Order Status of Open and/or Closed Guest Checks

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

Business Purpose
The user wants to view a summary of both open and closed guest checks that satisfy one or more filter conditions. This method can also be used when the user wants to know the KDS order status of one or more guest checks.

Method Description
This method was introduced in Simphony version 18.1 to get a summary of both open and closed guest checks that satisfy one or more filter conditions. If no filter is passed, this method will return all open guest checks that are created on the current business date in the default revenue center assigned to the POS API workstation. If closed checks need to be returned too, 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

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

- **RvcObjectNum**
  Pass Object Number of an RVC that is currently assigned to a POS API workstation to get only checks created in 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 hold those status ID as their current KDS order status. Possible KDS order status IDs include:
   - 30 means DS_SENT (order has been sent to the kitchen with at least 1 menu item)
   - 50 means DS_PREP_DONE (order has been prepared by at least one station)
   - 60 means DS_READY (order has been expo done and is ready to be distributed to the customer)
   - 100 means DS_CANCELLED (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 False or value not specified, this method will return only open guest checks.

None of these fields mandats 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.

**Parameters**

<table>
<thead>
<tr>
<th><strong>ppCheckFilter</strong></th>
<th>Criteria based on which guest checks need to be filtered.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ppChecksResponse</strong></td>
<td>Holds status of operation along with a summary of filtered guest checks. The summary includes KDS order status as well.</td>
</tr>
</tbody>
</table>

**Return Value**

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

Get Configuration Information

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.

Get Configured Information (method 1)

void GetConfigurationInfo
        (string vendorCode, int employeeObjectNum, int[] configurationInfoType,
        int revenueCenter, ref SimphonyPosAPI_ConfigInfoResponse configInfoResponse)

Business Purpose
The user wants to fetch configuration data from the Simphony POS database.

Method Description
This method returns configuration data, such as menu item definition and menu item price found in the POS database. If the volume of configuration data is found to be large, this method may throw a timeout error. 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 returns configuration data for only a specified range of records. The caller has to call this new method multiple times to retrieve a complete list of records. Review the next section for more details on the new version of this method. The following list of configuration data can be retrieved using the 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. Event Types
25. Event SubTypes
26. Event Definitions
27. TAX

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vendorCode</td>
<td>Vendor code for license validation. In Simphony version 2.7 MR3 or later, set this parameter to an empty string.</td>
</tr>
<tr>
<td>employeeObjectNum</td>
<td>Object number of employee who wants to perform this operation.</td>
</tr>
<tr>
<td>configurationInfoType</td>
<td>Array of information types for which details need to be fetched from the POS.</td>
</tr>
<tr>
<td>revenueCenter</td>
<td>Object number of revenue center.</td>
</tr>
</tbody>
</table>
configInfoResponse  Structure that holds the results. This holds configuration information along with the operation result. There is one field for each information type requested.

Return Value
Void. Result is encapsulated in the configInfoResponse reference parameter.

Get Configured Information *(method 2)*

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

Business Purpose
The user wants to fetch configuration data for a specified range of records from the Simphony POS database.

Method Description
As mentioned in the previous section, this is a new version of the GetConfigurationInfo method. The main difference between these two methods is that this new method returns 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 a large volume of configuration data for one or more types (for example, menu item definition has more than 30,000 records).

Parameters

<table>
<thead>
<tr>
<th>configInfoRequest</th>
<th>Request parameter that holds filter conditions to be applied while retrieving configuration data. This includes configuration data type and ranges (for example, StartIndex and MaxRecordCount).</th>
</tr>
</thead>
<tbody>
<tr>
<td>configInfoResponse</td>
<td>Structure that holds the results. This holds configuration information along with the operation result. There is one field for each information type requested.</td>
</tr>
</tbody>
</table>

Return Value
Void. Result is encapsulated in the configInfoResponse reference parameter.
Structure Reference

This section describes the various structures used for pass data for request and response.

**SimphonyPosAPI_CheckSummary**

This structure is to encapsulate summary details of a check.

**Public Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int CheckSeq</td>
<td>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.</td>
</tr>
<tr>
<td>int CheckNum</td>
<td>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.</td>
</tr>
<tr>
<td>int CheckEmployeeObjectNum</td>
<td>ID of employee who created the check</td>
</tr>
<tr>
<td>int CheckRevenueCenterObjectNum</td>
<td>ID of revenue center this check is currently active</td>
</tr>
<tr>
<td>int CheckLastWorkstationOwner</td>
<td>Object number of workstation that owned the check last time</td>
</tr>
<tr>
<td>int CheckCurrentlyOpenOnWorkstation</td>
<td>Object number of workstation that has this check currently opened</td>
</tr>
<tr>
<td>int CheckTableObjectNum</td>
<td>Object number of dining table for which the check created.</td>
</tr>
<tr>
<td>int CheckTableGroup</td>
<td>ID of table group in which dining table of this check falls under</td>
</tr>
<tr>
<td>int CheckOrderType</td>
<td>Order type ID of the check. E.g. Dine In and Eat Out</td>
</tr>
<tr>
<td>string CheckID</td>
<td>Name to identify a check. Duplicate check names on open checks are not allowed.</td>
</tr>
<tr>
<td>string CheckTotalDue</td>
<td>Due or balance amount to be paid by the customer for the check</td>
</tr>
<tr>
<td>DateTime CheckLastServiceTime</td>
<td>The time when the check was submitted last time to POS database</td>
</tr>
<tr>
<td>Field Type</td>
<td>Field Name</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>DateTime</td>
<td>CheckOpenTime</td>
</tr>
<tr>
<td>DateTime</td>
<td>CheckAutoFireTime</td>
</tr>
<tr>
<td>short</td>
<td>CheckInTraining</td>
</tr>
<tr>
<td>short</td>
<td>CheckInsufficientBeverage</td>
</tr>
<tr>
<td>short</td>
<td>CheckTransferedToDriver</td>
</tr>
<tr>
<td>short</td>
<td>CheckIsDelayedOrder</td>
</tr>
<tr>
<td>short</td>
<td>CheckIsFutureOrder</td>
</tr>
</tbody>
</table>

**SimphonyPosAPI_CheckSummaryEx**

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

**Public Attributes**

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SimphonyPosApi_KdsOrderStatus</td>
<td>LastKnownKdsOrderStatus</td>
<td>Last status reported by KDS device (e.g. DS_SENT, DS_PREP_DONE, DS_CANCELLED)</td>
</tr>
<tr>
<td>ARRAY(SimphonyPosApi_KdsOrderStatus)</td>
<td>KdsOrderStatusHistory</td>
<td>List of order status reported by KDS for the history of the order</td>
</tr>
</tbody>
</table>
SimphonyPosAPI_OpenChecks

This structure is to encapsulate details of all open checks.

Public Attributes

<table>
<thead>
<tr>
<th>ARRAY(SimphonyPosAPI_CheckSummary)</th>
<th>CheckSummary</th>
</tr>
</thead>
<tbody>
<tr>
<td>A structure to hold summary of all open checks</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SimphonyPosAPI_OperationalResult</th>
<th>OperationalResult</th>
</tr>
</thead>
<tbody>
<tr>
<td>A structure that indicates whether the current operation succeeded or not. In case of failure, this will have appropriate error code and error message</td>
<td></td>
</tr>
</tbody>
</table>

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.

It is possible to create a future check (a.k.a. auto fire check), by mentioning the appropriate value for CheckStatusBits property. The system will apply all applicable automatic discounts and service charges while creating the guest check. In addition, the system calculates the tax amount based on how the order type is configured in the EMC.

The details, such as Check Number and Check Sequence Number of the created check, will be filled in the pGuestCheck parameter. This method prints the guest check and credit voucher if configured to do so in EMC for the given workstation.

Public Attributes

<table>
<thead>
<tr>
<th>string</th>
<th>CheckID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name to identify a check. Duplicate check names on open checks are not allowed</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>CheckTableObjectNum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object number of dining table for which the check opened</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>CheckRevenueCenterID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object number of revenue center</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>CheckOrderType</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order type ID of the check. E.g. Dine In and Eat Out</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>CheckEmployeeObjectNum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object number of employee who opened the check</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>CheckSeq</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>CheckNum</th>
</tr>
</thead>
</table>

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 the 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
<table>
<thead>
<tr>
<th>Lookup start date. All checks that were created on/after this date would be returned in the result.</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bool IncludeClosedCheck</code></td>
</tr>
<tr>
<td>A boolean flag to specific whether or not closed checks need to be included in the result</td>
</tr>
</tbody>
</table>

**SimphonyPosAPI_CheckResponse**

This structure is to encapsulate the response of the 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 the 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 the 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**

<table>
<thead>
<tr>
<th>SimphonyPosAPI_MenuItemDefinition</th>
<th>MenuItem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structure that defines details of a main menu item. The details include object number of menu item, price, discount etc.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ARRAY(SimphonyPosAPI_MenuItemDefinition)</th>
<th>Condiments</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of a structure that defines details of condiment added to menu item</td>
<td></td>
</tr>
</tbody>
</table>

SimphonyPosAPI_MenuItemEx

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).

This structure is similar to SimphonyPosAPI_MenuItem. However, it uses the newer “Ex” structure for MenuItem and Condiments and adds support for the Extensibility attribute.

**Public Attributes**

<table>
<thead>
<tr>
<th>SimphonyPosAPI_MenuItemDefinitionEx</th>
<th>MenuItem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structure that defines details of a main menu item. The details include object number of menu item, price, discount etc.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SimphonyPosAPI_MenuItemDefinitionEx[]</th>
<th>Condiments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array of a structure that defines details of condiment added to menu item</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SimphonyPosApi_Extensibility[]</th>
<th>Extensibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array of a structure that defines extensibility detail that can be added to the menu item.</td>
<td></td>
</tr>
</tbody>
</table>

SimphonyPosAPI_MenuItemDefinition

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

**Public Attributes**

<table>
<thead>
<tr>
<th>int MiObjectNum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object number of given menu item</td>
</tr>
</tbody>
</table>
### Chapter 5
**Structure Reference**

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int MiMenuLevel</code></td>
<td>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.</td>
</tr>
<tr>
<td><code>int MiSubLevel</code></td>
<td>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.</td>
</tr>
<tr>
<td><code>int MiPriceLevel</code></td>
<td>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.</td>
</tr>
<tr>
<td><code>string MiOverridePrice</code></td>
<td>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.</td>
</tr>
<tr>
<td><code>string MiWeight</code></td>
<td>Weight of given item. This is not currently supported in the API.</td>
</tr>
<tr>
<td><code>string MiReference</code></td>
<td>A text that needs to be added as reference to given menu item</td>
</tr>
<tr>
<td><code>SimphonyPosApi_Discount ItemDiscount</code></td>
<td>Discount that needs to applied to given menu item</td>
</tr>
</tbody>
</table>

**Remarks**

The `MiReference` value can contain extra data to control additional behavior. This is used to specify a quantity, to override active tax rates, and to indicate a default condiment.

Format: `<ExtraData><!- extra data specified here --></ExtraData>reference-text`  
Example: `<ExtraData><MiQuantity>3</MiQuantity></ExtraData>VIP`

### List of Extra Data Elements

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>MiQuantity</code></td>
<td>Used in v1 of API to specify quantity other than one. If not specified, the default quantity of 1 is used.</td>
</tr>
<tr>
<td><code>Example:</code></td>
<td><code>&lt;MiQuantity&gt;2&lt;/MiQuantity&gt;</code></td>
</tr>
<tr>
<td><code>TaxOverride</code></td>
<td>Used to override tax rates active for menu item. The value is a string of up-to 64 zeros or ones. A one indicates the tax rate for the ordinal position is enable, whereas zero indicates the rate is disabled. If value are only provided for the first 10 rates, the remaining rates use the configured behavior.</td>
</tr>
<tr>
<td><code>Example:</code></td>
<td><code>&lt;TaxOverride&gt;101.....&lt;/TaxOverride&gt;</code></td>
</tr>
<tr>
<td><code>AsDefaultCondiment</code></td>
<td></td>
</tr>
</tbody>
</table>
Use to indicate condiment is a default condiment. This extra data only applies to Condiments.

Example: <AsDefaultCondiment/>

Here is a more complete example:

“<ExtraData><MiQuantity>3</MiQuantity></ExtraData>”

In this example, 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 it 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 on the screen. Refer to the code snippet section of this document for more details.

**SimphonyPosAPI_MenuItemDefinitionEx**

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

### Public Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int MiObjectNum</td>
<td>Object number of given menu item</td>
</tr>
<tr>
<td>int MiMenuLevel</td>
<td>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</td>
</tr>
<tr>
<td>int MiSubLevel</td>
<td>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</td>
</tr>
<tr>
<td>int MiPriceLevel</td>
<td>Sequence number to be used while picking up a price definition from the list. <strong>This is not currently supported in Transaction Services web service.</strong> That is, price definition will always be picked up based on the value of Sub level or Main level mentioned above</td>
</tr>
<tr>
<td>string MiOverridePrice</td>
<td>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.</td>
</tr>
<tr>
<td>string MiWeight</td>
<td>Weight of given item. <strong>This is not currently supported in the API.</strong></td>
</tr>
<tr>
<td>string MiReference</td>
<td>A text that needs to be added as reference to given menu item</td>
</tr>
<tr>
<td>decimal MiQuantity</td>
<td>The quantity of item to add to the guest check.</td>
</tr>
</tbody>
</table>
int MiDefinitionSeqNum

The number of the definition to use when ordering the item. Different definitions may have different behavior, prices depending on configuration in EMC.

SimphonyPosApi_DiscountEx[] ItemDiscount

List of discount to apply to a given menu item. In the response, this attribute indicates the discounts applied to this item by coupon and/or automatic discounts.

Remarks

The MiReference value can contain extra data to control additional behavior. See the Remarks section for SimphonyPosAPI_MenuItemDefinition for more information.

List of Extra Data Elements

TaxOverride

Used to override tax rates active for menu item. The value is a string of up-to 64 zeros or ones. A one indicates the tax rate for the ordinal position is enable, whereas zero indicates the rate is disabled. If value are only provided for the first 10 rates, the remaining rates use the configured behavior.

Example: <TaxOverride>101.....</TaxOverride>

AsDefaultCondiment

Use to indicate condiment is a default condiment. This extra data only applies to Condiments.

Example: <AsDefaultCondiment/>

SimphonyPosAPI_ComboMeal

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

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

Remarks

When ordering combo meals, TS API is strict in checking all combo meal linkages. 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.

**SimphonyPosAPI_ComboMealEx**

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

**Public Attributes**

Inherits all attributes from SimphonyPosAPI_ComboMeal, and includes the following additional attributes.

<table>
<thead>
<tr>
<th>SimphonyPosApi_Extensibility</th>
<th>Extensibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provides extra information about detail.</td>
<td></td>
</tr>
</tbody>
</table>

**SimphonyPosAPI_Discount**

This structure is to encapsulate the details of a discount.

**Public Attributes**

Inherits all attributes from SimphonyPosAPI_Discount, and includes the following additional attributes.

<table>
<thead>
<tr>
<th>int</th>
<th>DiscObjectNum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discount Object Number</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>string</th>
<th>DiscAmountOrPercent</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>string</th>
<th>DiscReference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference text to be added to given discount for reference purpose</td>
<td></td>
</tr>
</tbody>
</table>

**SimphonyPosAPI_DiscountEx**

This structure is to encapsulate the details of a discount to be applied on a guest check.

**Public Attributes**

Inherits all attributes from SimphonyPosAPI_Discount, and includes the following additional attributes.

<table>
<thead>
<tr>
<th>SimphonyPosApi_Extensibility</th>
<th>Extensibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provides extra information about detail.</td>
<td></td>
</tr>
</tbody>
</table>
SimphonyPosAPI_SvcCharge

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

Public Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int SvcChgObjectNum</td>
<td>Object Number of Service Charge that needs to be applied on guest check</td>
</tr>
<tr>
<td>string SvcChgAmountOrPercent</td>
<td>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 “Predefined Service Charge”, the amount or percent will be taken from POS database with the help of Service Charge Object Number.</td>
</tr>
<tr>
<td>string SvcChgReference</td>
<td>Reference text to be added to given Service Charge item</td>
</tr>
</tbody>
</table>

SimphonyPosAPI_SvcChargeEx

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

Public Attributes

Inherits all attributes from SimphonyPosAPI_SvcCharge, and includes the following additional attributes.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SimphonyPosApi_Extensibility</td>
<td>Extensibility Providing extra information about detail.</td>
</tr>
</tbody>
</table>

SimphonyPosAPI_TmedDetailItemEx

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

Public Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int TmedObjectNum</td>
<td>Object number of tender media chosen for payment</td>
</tr>
<tr>
<td>string TmedPartialPayment</td>
<td>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.</td>
</tr>
<tr>
<td>string TmedReference</td>
<td>Tender Media reference information</td>
</tr>
<tr>
<td>SimphonyPosAPI_EPayment</td>
<td>TmedEPayment Electronic Payment details</td>
</tr>
</tbody>
</table>
SimphonyPosAPI_TmedDetailItemEx2

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

**Public Attributes**

Inherits all attributes from SimphonyPosAPI_TmedDetailItemEx, and includes the following additional attributes.

```markdown
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SimphonyPosApi_Extensibility</td>
<td>Extensibility provides extra information about detail.</td>
</tr>
</tbody>
</table>
```

SimphonyPosApi_Extensibility

This structure is to encapsulate the details of extension data that can be added to menu items, discounts, service charges, or tender detail items.

**Public Attributes**

```markdown
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string DisplayName</td>
<td>A string displayed and/or printed if this item is displayed or printed.</td>
</tr>
<tr>
<td>string ExtensibilityAppName</td>
<td>A string indicating the name of the application associated with this data.</td>
</tr>
<tr>
<td>string ExtensibilityDataName</td>
<td>A string selected by the user that names the data stored by this extensibility item.</td>
</tr>
<tr>
<td>string ExtensibilityDataType</td>
<td>A string selected by the user that describes the type of data stored by this extensibility item.</td>
</tr>
<tr>
<td>string PrintOptionBits</td>
<td>This attribute is not currently used.</td>
</tr>
<tr>
<td>string StringData</td>
<td>This attribute is used to store the data payload for the extensibility data.</td>
</tr>
<tr>
<td>string DataID</td>
<td>This attribute is not currently used.</td>
</tr>
</tbody>
</table>
```

SimphonyPosAPI_EPayment

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

**Public Attributes**

```markdown
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPaymentDirective PaymentCommand</td>
<td>Enumeration on payment method (for example, credit authorization only, credit authorization and pay, debit authorization only, debit authorization and pay, SVC authorization, SVC redeem). Possible values are:</td>
</tr>
<tr>
<td></td>
<td>• NO_E_PAYMENT</td>
</tr>
</tbody>
</table>
```
- **AUTHORIZE_AND_PAY**
- **DEBIT_AUTHORIZE_AND_PAY**

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

<table>
<thead>
<tr>
<th><strong>EAccountDataSource</strong></th>
<th><strong>AccountDataSource</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Enumeration on source of payment details (for example, magnetic stripe, RFID card, or manually keyed). Possible values are:</em></td>
<td></td>
</tr>
<tr>
<td>- SOURCE_UNDEFINED</td>
<td></td>
</tr>
<tr>
<td>- RFID_TRACK_DATA_RULES</td>
<td></td>
</tr>
<tr>
<td>- RFID_M_CHIP_RULES</td>
<td></td>
</tr>
<tr>
<td>- MANUALLY_KEYED_TRACK_1_CAPABLE</td>
<td></td>
</tr>
<tr>
<td>- MANUALLY_KEYED_TRACK_2_CAPABLE</td>
<td></td>
</tr>
<tr>
<td>- MANUALLY_KEYED_NO_CARD_READER</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>EAccountType</strong></th>
<th><strong>AccountType</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Type of account (for example, checking and savings). Possible values are:</em></td>
<td></td>
</tr>
<tr>
<td>- ACCOUNT_TYPE_UNDEFINED</td>
<td></td>
</tr>
<tr>
<td>- CHECKING</td>
<td></td>
</tr>
<tr>
<td>- SAVINGS</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>string</strong></th>
<th><strong>AcctNumber</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Account number of payment card.</em></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>string</strong></th>
<th><strong>AuthorizationCode</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Authorization code of payment card.</em></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>DateTime</strong></th>
<th><strong>StartDate</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Start date as mentioned in payment card.</em></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>short</strong></th>
<th><strong>IssueNumber</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Issue number as mentioned in payment card.</em></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>string</strong></th>
<th><strong>Track1Data</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Magnetic stripe data for Track 1.</em></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>string</strong></th>
<th><strong>Track2Data</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Magnetic stripe data for Track 2.</em></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>string</strong></th>
<th><strong>Track3Data</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Magnetic stripe data for Track 3.</em></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>string</strong></th>
<th><strong>BaseAmount</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Base amount to be debited. This does not include tip or cash back amount.</em></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>string</strong></th>
<th><strong>TipAmount</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Amount to be debited for tip.</em></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>string</strong></th>
<th><strong>CashBackAmount</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Cash back amount.</em></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>string</strong></th>
<th><strong>KeySerialNum</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Debit key serial number for given transaction. Maximum length is 20 characters.</em></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>string</strong></th>
<th><strong>DeviceId</strong></th>
</tr>
</thead>
</table>
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**
Card Verification Value (CVV) 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 of 32 characters.

### 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 the current operation succeeded or failed. In case of failure, this will have appropriate error code and message.
SimphonyPosAPI_TotalsResponseEx

This structure is to encapsulate the details on totals of a transaction.
Inherits all attributes from SimphonyPosAPI_TotalsResponse, and includes the following additional attributes.

**Public Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CheckTaxDataPerRate[]</td>
<td>List of tax rate data for each tax rate.</td>
</tr>
</tbody>
</table>

SimphonyPosAPI_ConfigInfoRequest

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

**Public Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VendorCode</td>
<td>The vendor code for license validation (pass empty for Simphony version 2.7 MR3 or later. )</td>
</tr>
<tr>
<td>EmployeeObjectNumber</td>
<td>Employee object number for validation purposes only.</td>
</tr>
<tr>
<td>RVCObjectNumber</td>
<td>Object number of the revenue center for which configuration data is needed.</td>
</tr>
<tr>
<td>ConfigurationInfo</td>
<td>This holds the IDs of configuration data type along with start index and maximum records to be returned for each configuration data type.</td>
</tr>
</tbody>
</table>

SimphonyPosAPI_ConfigInfo

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

**Public Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConfigurationInfoType</td>
<td>The type of configuration data that needs to be fetched from the Simphony POS database (for example, menu item definition, service charge definition, discount definition, and so on).</td>
</tr>
<tr>
<td>StartIndex</td>
<td>Index of first record to be fetched from the POS database.</td>
</tr>
<tr>
<td>MaxRecordCount</td>
<td>Maximum number of records to be fetched.</td>
</tr>
</tbody>
</table>
SimphonyPosAPI_ConfigInfoResponse

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

**Public Attributes**

<table>
<thead>
<tr>
<th>ARRAY(\texttt{EConfigurationInfoType}) ConfigInfoType</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of type of configuration information for which details need to be fetched from the Simphony POS database (for example, menu item definition, service charge definition, discount definition, and so on). Possible enumeration values are:</td>
</tr>
<tr>
<td>- UNDEFINED = 0</td>
</tr>
<tr>
<td>- MENUITEMDEFINITIONS = 1</td>
</tr>
<tr>
<td>- MENUITEMPRICE = 2</td>
</tr>
<tr>
<td>- MENUITEMCLASS = 3</td>
</tr>
<tr>
<td>- SERVICECHARGE = 4</td>
</tr>
<tr>
<td>- DISCOUNTDEFINITIONS = 5</td>
</tr>
<tr>
<td>- TENDERMEDIA = 6</td>
</tr>
<tr>
<td>- ORDETYPE = 7</td>
</tr>
<tr>
<td>- FAMILYGROUP = 8</td>
</tr>
<tr>
<td>- MAJORGROUP = 9</td>
</tr>
<tr>
<td>- REVENUECENTERPAMETER = 10</td>
</tr>
<tr>
<td>- REVENUECENTERS = 11</td>
</tr>
<tr>
<td>- INTERFACES = 12</td>
</tr>
<tr>
<td>- MENUITEMMASTERS = 13</td>
</tr>
<tr>
<td>- SERVINGPERIODS = 14</td>
</tr>
<tr>
<td>- CURRENCY = 15</td>
</tr>
<tr>
<td>- VERSION = 16</td>
</tr>
<tr>
<td>- EMPLOYEES = 17</td>
</tr>
<tr>
<td>- TABLES = 18</td>
</tr>
<tr>
<td>- LANGUAGEINFORMATION = 19</td>
</tr>
<tr>
<td>- MENULEVEL = 20</td>
</tr>
<tr>
<td>- MENUITEMSLU = 21</td>
</tr>
<tr>
<td>- MAINMENULEVEL = 22</td>
</tr>
<tr>
<td>- SUBMENULEVEL = 23</td>
</tr>
<tr>
<td>- EVENTDEFINITIONS = 24</td>
</tr>
<tr>
<td>- TAX = 25</td>
</tr>
</tbody>
</table>

**string** MenuItemDefinitions
Details of all menu item definitions configured in the EMC for a given revenue center.

**string** MenuItemPrice
Details of all menu item price records (for example, menu item definition ID, price, preparation cost, and so on) configured in the EMC.

**string** MenuItemClass
Details of all menu item classes (for example, tax class, sales, discount and service charge itemizers, pricing calculation, and so on) as configured in the EMC.

**string** ServiceCharge
Details of service charges (for example, service charge amount/percent, tips)

**string** Discounts
Details of all discounts configured in the EMC at the enterprise level.

<table>
<thead>
<tr>
<th>string</th>
<th>TenderMedia</th>
</tr>
</thead>
<tbody>
<tr>
<td>Details of tender media configured in the EMC for payment (for example, cash and credit cards)</td>
<td></td>
</tr>
<tr>
<td><strong>Note:</strong> Transaction Services only supports the <strong>MCreditDebit Payment</strong> driver for credit/debit card payment.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>string</th>
<th>OrderType</th>
</tr>
</thead>
<tbody>
<tr>
<td>Details of order types (for example, Dine-in and Carry out) configured in the EMC for a given property.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>string</th>
<th>FamilyGroups</th>
</tr>
</thead>
<tbody>
<tr>
<td>Details of all Family Groups (that is, category of menu items) configured in the EMC.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>string</th>
<th>MajorGroup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Details of all Major Groups configured in the EMC for menu items (for example, food and beverages).</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>string</th>
<th>RevenueCenterParameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Details of revenue center parameters that are configured in the EMC (for example, secondary print language, minimum and maximum check number, database update frequency, options, and so on).</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>string</th>
<th>RevenueCenters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Details of revenue centers configured in the EMC for a given property.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>string</th>
<th>Interfaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>Details of all interfaces configured in the EMC at the enterprise level.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>string</th>
<th>MenuItemMasters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Details of all menu item master records (that is, property level menu item record).</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>string</th>
<th>ServingPeriod</th>
</tr>
</thead>
<tbody>
<tr>
<td>Details of serving period (for example, Breakfast 4am to 11am)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>string</th>
<th>Currency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Details of all currencies (for example, US Dollar, Peso) configured in the EMC at enterprise level.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>string</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current version of the Transaction Services web service (for example, 2.700.0.77).</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>string</th>
<th>Employees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Details of all employees configured in the EMC at the enterprise level.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>string</th>
<th>Tables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Details of dining tables configured in the EMC for a given property.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>string</th>
<th>LanguageInformation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Details of languages (for example, English, Spanish) that are configured in the EMC.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>string</th>
<th>MenuLevel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Details of menu level sets configured (for example, Main, Sub, and Custom levels).</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>string</th>
<th>MenuItemSlu</th>
</tr>
</thead>
</table>
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 the EMC at enterprise and property levels (maximum of 64 tax rates).

**SimphonyPosApi_OperationResult** OperationalResult
A structure that indicates whether the current operation succeeded or failed. 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**

<table>
<thead>
<tr>
<th>ARRAY(string) CheckPrintLines</th>
</tr>
</thead>
<tbody>
<tr>
<td>Printed lines of a guest check.</td>
</tr>
</tbody>
</table>

**SimphonyPosAPI_OperationalResult** OperationalResult
A structure that indicates whether the current operation succeeded or failed. 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 the status of a print job.

**Public Attributes**

<table>
<thead>
<tr>
<th>SimphonyPrintApi_Status</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>An enumerator that indicates the current status of a specified print job. Possible values are:</td>
<td></td>
</tr>
<tr>
<td>• JobPending = 0</td>
<td></td>
</tr>
<tr>
<td>• JobComplete = 1</td>
<td></td>
</tr>
<tr>
<td>• JobAborted = 2</td>
<td></td>
</tr>
<tr>
<td>• JobSentToBackup = 3</td>
<td></td>
</tr>
<tr>
<td>• JobFailed = 4</td>
<td></td>
</tr>
<tr>
<td>• JobNotFound = 5</td>
<td></td>
</tr>
</tbody>
</table>

**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**

A structure that indicates whether the current operation succeeded or failed. 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
- DiscountNotAllowedFilterActive
- DiscountOnParentCombo
- DuplicateLineNumber
- EGatewayClientStartError
- EGatewayClientStopError
- EGatewayConnectionError
- EGatewayConnectionNotInPool
- InvalidAuthCode
- 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
CheckTaxDataPerRate

This structure is to encapsulate the details of tax applied for a tax rate.

**Public Attributes**

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>Index</td>
<td>The tax rate number (range 1 – 64).</td>
</tr>
<tr>
<td>decimal</td>
<td>Tax</td>
<td>The total amount of tax applied for this rate.</td>
</tr>
<tr>
<td>bool</td>
<td>Exempt</td>
<td>When True, indicates that tax for this rate has been exempted.</td>
</tr>
<tr>
<td>bool</td>
<td>AnyApplied</td>
<td>When True, indicates that tax for this rate has been applied to one or more items.</td>
</tr>
</tbody>
</table>
Example and Clode Snippets

Calculate Totals of a Transaction

The following scenario describes a user who wants to find out the total amount of a transaction for items that are being ordered by the customer.

- Add two menu items
  - Add two condiments to the first menu item, and add one condiment to the second menu item
  - Quantity of 1st menu item is 3, and 2nd menu item is 1
  - Override the price of the 1st menu item
  - For the 1st menu item, instead of the default definition, pick up a specific menu item definition based on main and sub menu levels
  - Apply an open discount to the 1st menu item
  - Add a reference text to the 1st menu item
- Add a combo meal
- Apply an Open service charge
  - Add a reference text on the service charge
- Apply a Closed discount on the subtotal (that is, at the check level)

<table>
<thead>
<tr>
<th>S. No</th>
<th>Type of Data</th>
<th>Parameter Name</th>
<th>Sample Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Vendor code</td>
<td>vendorCode</td>
<td>yzsroloq</td>
</tr>
</tbody>
</table>
| 2     | Menu Items and Condiments along with item level Discount | ppMenuItems | Object number of two menu items are 110003 and 110004
Object numbers of two condiments of first menu item are 41103 and 44502. Object number of condiment of second menu item is 41103.
Overridden price for first menu item is $10
Menu levels to pick up first menu item is *Main Menu Level - 2 and Sub Menu Level - 3*
Open discount percent for first menu item is 7%
Reference text for first menu item is “Chef’s favorite” |
| 3     | Combo Meal   | ppComboMeals   | Combo meal details
Object number of combo meal is 10
Object number of main item is 110003
Object numbers of side items are 41103 and 44502 |
Order type is Dine-in

The following sample data is provided for the scenario mentioned above.

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

**Calculate Transaction Totals Method Signature**

```csharp
void CalculateTransactionTotals
(
    string vendorCode,  // Service Charge
    ref ARRAY(SimphonyPosAPI_MenuItem) ppMenuItems,  // ppMenuItems
    ref ARRAY(SimphonyPosAPI_ComboMeal) ppComboMeals,  // ppComboMeals
    ref SimphonyPosAPI_SvcCharge pServiceChg,  // pServiceChg
    ref SimphonyPosAPI_Discount pSubTotalDiscount,  // pSubTotalDiscount
    int revenueCenterObjectNum,  // revenueCenterObjectNum
    short orderType,  // orderType
    int employeeObjectNum,  // employeeObjectNum
    ref SimphonyPosAPI_TotalsResponse pTotalsResponse  // pTotalsResponse
)
```

The following code snippet demonstrates how data for input parameters of the `CalculateTransactionTotals` method can be constructed and used to invoke the method.
SimphonyPosAPIWebSoapClient mTSApi = new SimphonyPosAPIWebSoapClient();

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

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

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

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

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

**Vendor Code**

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 the EMC in the following location for TS API to work properly.

**Enterprise** level, **Setup** tab, **Parameters** section, **Enterprise Parameters** module, **Licensing** tab.
This requirement was removed in Simphony version 2.7 MR3 and later. However, this parameter still exists 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 can pass a valid Vendor Code that was distributed to them.

The following image shows the EMC dialog where you can configure the Vendor Activation Code for the Transaction Services API.

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 the enterprise, property, or revenue center level in the EMC using the following module.

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

**Parameter Signature**

<table>
<thead>
<tr>
<th>String</th>
<th>vendorCode</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>e.g.</em></td>
<td><code>string vendorCode = &quot;yzsroioq&quot;;</code></td>
</tr>
</tbody>
</table>

**SimphonyPosAPI_MenuItem Signature**

```java
public class SimphonyPosApi_MenuItem
{
    public SimphonyPosApi_MenuItemDefinition[] Condiments;
    public SimphonyPosApi_MenuItemDefinition MenuItem;
}
```
A menu item is the core foundation of all POS transactions. Everything ordered or added to the system is a menu item. In restaurant revenue centers, drinks and entrees are menu items. In retail revenue centers, shirts and hats are also considered menu items. Therefore, in Symphony, it can be said that any item being sold is a menu item.

The following code snippet demonstrates how to construct input data for the menu item list parameter. This code adds two menu items and respective condiments to the list as required. It also applies a discount (at the item level) to the first menu item.

```csharp
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.MenuItem.Condiments = new SimphonyPosApi_MenuItemDefinition[2];
    firstMenuItem.MenuItem.Condiments[0] = GetFirstCondimentItem();
    menuItemList.Add(firstMenuItem);

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

    return menuItemList.ToArray();
}
```

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

```csharp
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;
}
```
The following code demonstrates how to construct an object for two condiment menu items with given input data.

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

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

The following code demonstrates how to construct a discount object for given input data. Each discount configured in the EMC is identified by a unique identifier called Discount Object Number. For a preset discount, the amount or percentage of the discount is taken from a value configured in the EMC. However, for an open discount, the amount or percentage of the discount should be supplied by the caller. The property DiscAmountOrPercent could be an amount or percent based on how the given discount is configured using the 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 items explicitly in this way, while API applies an automatic discount implicitly by itself.

```csharp
private SymphonyPosApi_Discount GetItemDiscount(bool isOpenDiscount)
{
    SymphonyPosApi_Discount discount = new SymphonyPosApi_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

A combo meal is a combination meal (for example, a burger with fries and a drink, or pancake with ham and coffee) offered at a lower price than the menu item’s cost individually. Configure a combo meal in the EMC before adding it through the TS API.

Combo meals can be found in EMC at the **Property** level, **Configuration** tab, **Sales** section, **Combo Meals** module.

To configure a combo meal:

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

### Parameter Signature

```java
ref ARRAY(SimphonyPosAPI_ComboMeal) ppComboMeals
```

### SimphonyPosApi_ComboMeal Signature

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

The following code snippet demonstrates how to construct a 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. Log in to the EMC to obtain the object numbers of the combo meal and related items.

```java
private SimphonyPosApi_ComboMeal[] GetComboMealList() {
    SimphonyPosApi_ComboMeal[] comboMeal = new SimphonyPosApi_ComboMeal[1];
    SimphonyPosApi_ComboMeal comboMeal1 = new SimphonyPosApi_ComboMeal();
    comboMeal1.ComboMealObjectNum = 10;
```
// Add a Main item
SimphonyPosApi_MenuItem mainItem = new SimphonyPosApi_MenuItem();
mainItem.MenuItem = new SimphonyPosApi_MenuItemDefinition();
mainItem.MenuItem.MiObjectNum = 110003;
comboMeal1.ComboMealMainItem = mainItem;

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

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

comboMeal[0] = comboMeal1;
return comboMeal;

Service Charge

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

- Automatic Service Charge
- Manual Service Charge

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

A manual service charge should be added to the input parameter of the TS API.

Configure a service charge in the EMC at the Enterprise or Property level, Configuration tab, Sales, Service Charges module

Parameter Signature

```
ref SimphonyPosAPI_SvcCharge  pServiceChg
```
SimphonyPosApi_SvcCharge Signature

```csharp
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 a given input data. Each service charge configured in the EMC is identified by a unique identifier called Service Charge Object Number. For a preset service charge, the amount or percentage of the service charge is taken from a value configured in the EMC. For an open service charge, the amount or percentage of the service charge should be supplied by the caller. The field SvcChgAmountOrPercent can be an amount or percent based on how it 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 the guest check explicitly in this way, while API applies an automatic service charge implicitly by itself.

```csharp
private SimphonyPosApi_SvcCharge GetServiceCharge(bool isOpenServiceCharge)
{
    SimphonyPosApi_SvcCharge serviceCharge = new SimphonyPosApi_SvcCharge();
    serviceCharge.SvcChgObjectNum = 12;
    if (isOpenServiceCharge)
    {
        serviceCharge.SvcChgAmountOrPercent = "6"
        serviceCharge.SvcChgReference = "6% service charge including tips";
    }
    return serviceCharge;
}
```

Subtotal Discount

A discount reduces the price of an item or items on a check. Discounts are generally used for promotional purposes (for example, a coupon for a free dessert) or for customer satisfaction. Discounts can be configured 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 the 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 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 adds items, the workstation continually looks for items that will trigger an 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

```csharp
ref SimphonyPosAPI_Discount pSubTotalDiscount
```

`SimphonyPosApi_Discount` Signature

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

The following code snippet demonstrates how to construct a Subtotal discount object with the given input data. Each discount configured in EMC is identified by a unique identifier called Discount Object Number. For a Preset discount, the amount or percentage of the discount will be taken from a value configured in EMC. For an Open discount, the amount or percentage of the discount should be supplied by the caller. The property `DiscAmountOrPercent` could be an amount or percent based on how the discount is configured in EMC. This example demonstrates that 11 is the Discount Object Number of an open Subtotal discount and the caller is applying a 5% discount to the guest check for all triggered menu item groups. All manual Subtotal discounts should be added to the guest check explicitly in this way, while API applies automatic Subtotal discounts implicitly.

```csharp
private SimphonyPosApi_Discount GetSubtotalDiscount(bool isOpenDiscount)
{
    SimphonyPosApi_Discount subTotalDiscount = new SimphonyPosApi_Discount();
    subTotalDiscount.DiscObjectNum = 11;
    // percentage or amount based on how it's configured in EMC
    if (isOpenDiscount)
    {
        subTotalDiscount.DiscAmountOrPercent = "5";
        subTotalDiscount.DiscReference = "Weekend discount";
    }
    return subTotalDiscount;
}
```
Revenue Center Object Number

Revenue centers are a distinctly identifiable department, division, or unit of a firm that generates revenue through the sale of goods and/or services. For example, the rooms department and food-and-beverages department of a hotel are 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 the EMC at the **Property** level, **Setup** tab, **Property Configuration, RVC Configuration** module. In the following example, 3016 is the Object Number of a revenue center.

```java
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 common-order types.

```java
Parameter Signature
short orderType
e.g. short orderType = 1; // e.g. Dine-in
```

Employee Object Number

Each employee at 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 to associate an employee for a given transaction.

Employee details, including their object number can be found in EMC: **Property** level, **Configuration** tab, **Personal, Employee Maintenance** module.

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

```java
Parameter Signature
int employeeObjectNum
e.g. int employeeObjectNum = 90001;
```
API Response

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

Parameter Signature

```csharp
ref SimphonyPosAPI_TotalsResponse pTotalsResponse
```

SimphonyPosApi_TotalsResponse Signature

```csharp
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

After the user calculates and reviews the total amount of a transaction, he or she may want to post that transaction and create a guest check in the Simphony POS database by providing tender/payment details. The tender can be of any type (for example, cash or 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 the auto-fire feature on guest checks. Auto-fire means that the guest check will be immediately created in the system, but it will fire only when the time specified to fire at the time of guest check creation is attained. This example demonstrates the auto-fire feature by firing the guest check only after 12 hours from when the guest check is posted. The auto-fire feature is used in hotels where a guest wants to order food in the morning to be consumed at dinner. In this case, the guest check will be created in the system as soon as the transaction posts, but will fire only at the specified time in the evening, so that the chef can prepare ordered food for dinner for the specific customer.
The input data for menu items, combo meals, discounts, and service charges are described in the previous section for calculating totals. The example is taken for this method as well. However, this method needs data for the following additional parameters.

Tender/Payment Details

The Post Transaction method posts current transactions to the Simphony POS database to create a guest check. Like CalculateTransactionTotals, this method calculates the transaction totals. If the payment/tender media is of Service Total, the system will create the guest check and keep it in the Open state. When the tender media with appropriate payment details (cash, credit/debit) are passed for full payment, the check will be created and changed to a Closed state at the end of the call. A tender with partial payment will still have the created check in the Open state only. Another tender with payment for the balance amount can be added later to that check using a method called AddToExistingCheckEx to close the check.

Post Transaction Method Signature

```csharp
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 the PostTransactionEx method can be constructed and used to invoke the method. This example demonstrates creating a guest check with the same input data mentioned in the previous section for calculate totals.

```csharp
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 listed in the Calculate Totals Transaction method in the previous section. Refer to the previous section for all other parameters.

**Guest Check**

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

- CheckEmployeeObjectNum
- CheckRevenueCenterID
- CheckOrderType

The following properties are optional:

- CheckGuestCount
- CheckTableObjectNum
- CheckStatusBits
- CheckDateToFire

The following properties will be populated by the API when the 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 the future for updating a specific guest check. For example, this method can create a guest check with a partial payment, and another method called AddToExistingCheckEx can be invoked later to add another tender for the balance amount to the same check by specifying the Check Number and Check Sequence Number of that check. An example of the AddToExistingCheckEx method is provided 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 must be configured in EMC before it can be used in the TS API.

The following code snippet demonstrates how tender media with sample data can be constructed based on the tender media type. This code indicates the data expected by the payment driver while creating a guest check.

```csharp
private SimphonyPosApi_TmedDetailItemEx GetTenderMedia(TenderMediaType tmType)
```
```java
SimphonyPosApi_TmedDetailItemEx tenderMedia = new SimphonyPosApi_TmedDetailItemEx();
SimphonyPosApi_EPayment ePayment = new SimphonyPosApi_EPayment();
switch (tmType)
{
    case TenderMedia.Type.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 TenderMedia.Type.CreditCard:
    {
        tenderMedia.TmedObjectNum = 30;
        ePayment.PaymentCommand = EPaymentDirective.CREDIT_AUTHORIZE_AND_PAY;
        // Track2 has most of the data required by the payment driver
        ePayment.Track2Data = "77766655554444=00010002000370783149"
        ePayment.BaseAmount = "30"; // Base amount excluding tip
        ePayment.TipAmount = "5"; // Amount for tip
        ePayment.CashBackAmount = "15";
        tenderMedia.TmedReference = "Total amount to be deducted from CREDIT CARD is $50";
        break;
    }
    case TenderMedia.Type.DebitCard:
    {
        tenderMedia.TmedObjectNum = 31;
        ePayment.PaymentCommand = EPaymentDirective.DEBIT_AUTHORIZE_AND_PAY;
        // Track2 has most of the data required by the payment driver
        ePayment.Track2Data = "8887776665555444=00020003000470783249"
        ePayment.BaseAmount = "30"; // Base amount excluding tip
        ePayment.TipAmount = "5"; // Amount for tip
        ePayment.CashBackAmount = "15";
        tenderMedia.TmedReference = "Total amount to be deducted from DEBIT CARD is $50";
        break;
    }
    case TenderMedia.Type.StoredValueCard:
    {
        tenderMedia.TmedObjectNum = 35;
        // Track2 has most of the data required by the payment driver
        ePayment.Track2Data = "99988887776666=10020003000470783249";
```
Chapter 6
Example and Code Snippets

```csharp
    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, refer to the Calculate Transaction Totals Method Signature section of this document.

**API Response**

If the post transaction operation succeeded, the output details (for example, Check ID, Check Number, and Check Sequence Number) of the created check will be populated in the appropriate fields of the pGuestCheck parameter. The operation also populates the print receipt of the guest check in the ppCheckPrintLines property of pGuestCheck, while it populates the credit voucher of the current transaction in the ppVoucherOutput property of the same parameter. If this method encounters a problem (for example, payment failure), it will not create a guest check and will throw an appropriate exception to the caller. In addition, the OperationalResult property of pTotalsResponse parameter will hold the appropriate error code and message in that case.

**Add an Item to an Open Guest Check**

After the user posts a transaction or check to the Simphony POS database, sometimes, he or she may need to update it. For example, when a transaction was posted to create a guest check with a Service Total as tender media. That is, payment has not been made yet for the transaction. In this case, the user may want to add a tender to the guest check later, in order to make payment for the amount due. In such cases, the AddToExistingCheckEx method can be used.

The following code snippet demonstrates adding a tender media to an existing open guest check to make payment so the check can be closed. Similar to a tender media, other items such as menu item, combo meal, service charge, or discount can be added to an open guest check.

```csharp
    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_Discout pSubtotalDiscount = new SimphonyPosApi_Discout();
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
{
}

Like the tender media given in this example, one or more items such as a menu item, 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 will throw an exception if the caller tries to add an item a closed check.

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

API Response

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

**Void All Items of an Open Guest Check**

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

**VoidTransaction Method Signature**

```csharp
void VoidTransaction(
    string vendorCode, 
    ref SymphonyPosAPI_GuestCheck pGuestCheck
)
```

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

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

public void InvokeVoidTransaction()
{
    SymphonyPosApi_GuestCheck guestCheck = new SymphonyPosApi_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
    {
    }
}
```

**API Response**

If the operation succeeded, the OperationalResult.Success field will hold True. In addition the, OperationalResult.ErrorCode will hold the error code, while OperationalResult.ErrorMessage holds the reason for failure.
Get Status of a Print Job

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

If a guest check did not print due to any reason, the status of the corresponding print job can be retrieved using the CheckPrintJobStatus method of TS API. This method accepts the ID of the 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.

```csharp
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 the operation succeeds, the OperationalResult.Success field will hold True. The printJobStatus will hold the status of queries print job. The following enumerator 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, the user may want to get a summary of all open guest checks from all revenue centers of the property.

GetOpenChecks Method Signature

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

The following code snippet demonstrates retrieving a summary of all open guest checks created by a specific employee whose Employee Object Number is 90001. The caller has to pass 0 (zero) to the Employee Object Number parameter if he or she wishes to fetch all open checks irrespective of the owner who created the check.

```csharp
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)
        }
    }
}
```
Example and Code Snippets

```csharp
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 the operation succeeds, the OperationalResult.Success field will hold True and a summary of open guest checks will be populated in the openChecks parameter. 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, the OperationalResult.Success field will hold False, while the OperationalResult.ErrorCode holds the 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), the GetOpenChecksEx method can be used instead of GetOpenChecks, which is explained in the previous section. GetOpenChecks populates CheckRevenueCenterObjectNum with the ID of the revenue center, while GetOpenChecksEx populates the same field with Object Number.

GetOpenChecksEx Method Signature

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

The following code snippet demonstrates retrieving a summary of all open guest checks created by a specific employee whose Employee Object Number is 90001. The caller has to pass 0 (zero) to the Employee Object Number parameter if he or she wishes to fetch all open checks irrespective of the owner who created the check. The property CheckRevenueCenterObjectNum of response object returns the object number of RVC instead of 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:");
            Console.WriteLine("Check Sequence Number: ");
            Console.WriteLine("Check Total Due:");
            Console.WriteLine("RVC Object Number: ");
        }
    }
    else
    {
        Console.WriteLine("Get Open Check failed. Error Code: {0}, Error Message: {1}",
            openChecks.OperationalResult.ErrorCode,
            openChecks.OperationalResult.ErrorMessage);
    }
}

API Response

If the operation succeeds, the OperationalResult.Success field will hold True and a 
summary of open guest checks will be populated in the openChecks parameter. 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 the reason for failure.

Get Open Guest Checks from a Specific RVC

Sometimes a user may want to view a summary of all open guest checks from a specific 
revenue center within the property. The following method can be used.

GetOpenChecksByRVC Method Signature

void GetOpenChecksByRVC
(
    string vendorCode,
    int EmployeeObjectNum,
    int revenueCenterObjectNum,
    ref SimphonyPosAPI_OpenChecks openChecks
)
The following code snippet demonstrates retrieving a 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 (zero) to the Employee Object Number parameter if he or she wishes to fetch all open checks irrespective of the owner who created the check.

```csharp
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 the operation succeeds, the OperationalResult.Success field will hold True and a summary of open guest checks will be populated in the `openChecks` parameter. In case of failure, the OperationalResult.Success field will hold False, while OperationalResult.ErrorCode holds the error code and OperationalResult.ErrorMessage holds the reason for failure.
Get Summary and KDS Order Status of Open and Closed Guest Checks

If a user wants to view a summary of guest checks that satisfies few filter conditions, the following method can be used.

**GetChecks Method Signature**

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

The following code snippet demonstrates how to retrieve both open and closed guest checks that were created in the last 5 days by an employee whose Employee Object Number is 90001.

```csharp
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 the operation succeeds, the OperationalResult.Success field will hold True and a summary of filtered guest checks will be populated in the response.Checks parameter. In case of failure, OperationalResult.Success field will hold False, while OperationalResult.ErrorCode holds the error code and OperationalResult.ErrorMessage holds the reason for failure.

Get Check Detail

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

GetCheckDetail Method Signature

```csharp
void GetCheckDetail(
    SimphonyPosApi_CheckDetailRequest ppCheckDetailFilter,
    SimphonyPosApi_CheckDetailResponse ppCheckDetailResponse
)
```

The following code snippet demonstrates retrieving complete details of a guest check in XML format and printing the most recent status of the order.

```csharp
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));
```

This snippet can be easily modified to work with different extensibility data names and formats.
if (xml1 != null && xml1.Count > 0)
{
    string mostRecentOrderStatus = string.Empty;
    DateTime mostRecentTimeStamp = DateTime.MinValue;

    foreach (XmlNode node in xml1)
    {
        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
{
}

API Response

If the operation succeeds, the OperationalResult.Success field will hold True and the check XML will be populated in the 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 the Check Number and a few other required details. This is often required where an external printer is used to print an open guest check. The method GetPrintedCheck can be used for this purpose. This method will retrieve the print lines in the output parameter without actually printing the guest check on a printer. This method works only on open guest checks.
GetPrintedCheck Method Signature

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

The following code snippet demonstrates retrieving print lines of a guest check.

```csharp
SimphonyPosAPIWebSoapClient mTSApi = new SimphonyPosAPIWebSoapClient();
string vendorCode = "lzsoioq"; // 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 the operation succeeds, the OperationalResult.Success field will hold True, and print lines of the guest check will populate in the checkPrintLines parameter. In case of failure, the OperationalResult.Success field will hold False, while the OperationalResult.ErrorCode holds error code and OperationalResult.ErrorMessage holds reason for failure.
Get Configured Information (Method 1 - GetConfigurationInfo)

Some of the data configured in the EMC can be retrieved from the 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 the Simphony POS database using the GetConfigurationInfo method. This method returns all records of the 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 large.

GetConfigurationInfo Method Signature

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

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

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

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

    SimphonyPosApi_ConfigInfoResponse configInfoResponse = new SimphonyPosApi_ConfigInfoResponse();

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

API Response
If the operation succeeds, the OperationalResult.Success field will hold True, and
configured data for queried type can be found in the appropriate fields of the
configInfoResponse object. In case of failure, the 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 the GetConfigurationInfo method to retrieve
configuration data batch by batch or in full. This method is useful when the POS
database has a large volume of configuration data for one or more types, such as Menu
Item Definition, Menu Item Master, and so on. This method is designed to return all
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 (start index and maximum records count) are
specified.

GetConfigurationInfoEx Method Signature

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

The following code snippet demonstrates retrieving configured data for the menu item
definition and menu item price.

SimphonyPosAPIWebSoapClient mTSApi = new SimphonyPosAPIWebSoapClient();
```csharp
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 miPriceRecordsCount = 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.IsNullOrEmpty(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.IsNullOrEmpty(response.MenuItemPrice))
{
    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;
        miPriceRecordsCount += 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 - " + miPriceRecordsCount);
System.Console.ReadLine();

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

If the operation succeeds, the OperationalResult.Success field will hold True, and configured data for queried type can be found in the appropriate fields of the 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.

Offline 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, checks will not be posted to the CAPS machine unless it is restarted.

Printing Services

The API supports printing to remote and local order devices. When a check is opened through 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 through 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, you can 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.
8

Demo Client for Transaction Services API

The Demo client is a Windows application used to demonstrate or test the features of the 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 a request to TS API and displays the response in the UI.

Application Path

The demo client application is located in the following folder of install media:

<InstallMediaFolder>\Install\Symphony2\Tools\PosAPIDemoClient

Prerequisites

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

Initial Setup

To configure and run the demo client application:

1. Copy the PosAPIDemoClient folder from install media to a local folder.
2. Launch POSAPI_WebClient.exe.
3. Enter the full address of the workstation hosting the TS service in the text box (highlighted in the following image).
Demonstration

Calculate Totals of a Transaction

The Demo client has a Calculate Totals button to invoke the \textit{CalculateTransactionTotals} method of TS API.

To pass input data through the UI and invoke the method:

1. Select the MenuItem option from the Type drop-down list.
2. Enter the menu item’s Object Number 110003 in the Number field, and then click the Add Item button.
   
You can obtain the Menu Item Object Number from the EMC, at the Property level, Configuration tab, Menu Items, Menu Item Maintenance module.

3. Enter 3016 in the RVC # textbox.

You can obtain the RVC number from the EMC, Property level, Setup tab, Property Configuration, RVC Configuration module.

4. Enter 90001 in the Employee # textbox.

You can obtain the Employee Number from the EMC, Property level, Configuration tab, Personnel, Employee Maintenance module.

5. Enter a valid vendor code in the License Activation Code field. This can be left blank for Simphony version 2.7 MR3 or later.

6. Click the Calculate Totals button to send the request to the TS API.

The status of the operation will appear in the lower area of the UI, while results appear in the block highlighted in green as shown in the following image.
Create a Guest Check

Use the **Post Transaction** button to send a request to create a new guest check in the Simphony database.

1. Select **MenuItem** from the **Type** drop-down list.
2. Enter the menu item’s Object Number **110003** in the **Number** field, and then click the **Add Item** button.
3. Select **RequiredCondiment** from c.
4. Enter the Object Number **41103** in the **Number** textbox, and then click the **Add Item** button.
5. For payment, select **Tender** from the **Type** drop-down list.
   a. For a cash payment, enter the Tender Media Object Number of Cash (for example, 2) in the **Number** field. Then enter the amount (for example, 10) in the **Value** textbox.
   b. For a debit or credit payment, enter the Tender Media Object Number of Credit/Debit, and then click the **Credit Auth** button to provide payment card details in the popup.

   **NOTE:**

   Transaction Services only supports the **MCreditDebit Payment** driver for credit/debit card payments.

6. To add tender details, click the **Add Item** button.
7. If you are using versions earlier than Simphony 2.7 MR3, enter values in the **RVC #**, **Employee #**, and **License Activation Code** fields.
8. To send a request to the TS API, click the **Post Transaction** button. The result is populated in the fields highlighted in green as shown in the following image.
Add an Item to an Open Guest Check

Use the Add to Check button 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 the check using the Add To Check button.

1. Select the MenuItem option from the Type drop-down list.
2. Enter the menu item’s Object Number 110003 in the Number field, and then click the Add Item button.
3. Select RequiredCondiment from the Type drop-down list.
4. Enter the Object Number 44502 in the Number field, and then click the Add Item button.
5. Select Tender from the Type drop-down list.
6. Enter the Tender Media Object Number of Service Total in the Number field, and then click the Add Item button.
7. If you are using versions earlier than Simphony 2.7 MR3, enter values in the RVC #, Employee #, and License Activation Code fields.
8. Click the Post Transaction button to create the guest check. As the tender type is Service Total, the check that is created will be in the Open state. One or more items can be added to the open check by following further steps below.

![Symphony POS API Demo client screenshot](image)
9. Click the **Clear MI details** button to clear current details.

10. Select **MenuItem** from the **Type** drop-down list, and then add a menu item to the existing guest check.

11. Enter the menu item’s Object Number **110004** in the **Number** field and then click the **Add Item** button.

12. Select **RequiredCondiment** from the **Type** drop-down list.

13. Enter the Object Number **41103** in the **Number** field, and then click the **Add Item** button.

14. Ensure that the value of Check Sequence Number and Check Number of original check still appears in the **Check Seq** and the **Check #** fields.

15. Click the **Add to Check** button.
Combo Meal Ordering

The following steps describe how to add a combo meal to the check. Ensure that a combo meal is already configured in the EMC so that it can be added on the POS API.

1. Select **ComboMeal** from the **Type** drop-down list, enter the combo meal Object Number in the **Number** field, enter the Object Number in the **Combo Menu Item Number** field, and then click the **Add Item** button.

2. To add a main item, select **ComboMain** from the **Type** drop-down list.
3. Enter the combo meal’s Object Number in the Number field, and then click the Add Item button.

4. To add side items, select ComboSide from the Type drop-down list.

5. Enter the side item’s Object Number in the Number field, and click the Add Item button.

6. Click the Calculate Totals button to calculate the price of the combo meal or add a tender, and then click the Post Transaction button to create a guest check.

---

**Void All Items of an Open Guest Check**

Use the Void Check button to send a request to void all items of a guest check.

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

To send a void request to TS API, click the Void Check button.
Get Summary of All Open Guest Checks

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

1. If you are using versions earlier than Simphony 2.7 MR3, enter the value for the License Activation Code field, and then click the Get Open Checks button.
The following dialog appears:

![Dialog Image]

2. Enter the **Employee Object Number** in the **Employee number** field to filter checks based on the employee who created it (or enter **0** to get all open checks irrespective of who created them).

3. Enter the appropriate value in the **Revenue Center** field, and enable the check box:
   a. Enter **-1** 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. Enter **-2** 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. Enter the **Object Number of any RVC** 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.
The following image shows a summary of all open checks.
Get Summary of Open and Closed Guest Checks

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

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

![Get Checks Window](image)

2. Enter data for one or more filters as needed (for example, 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.

   The **Check Sequence Number** field is not applicable to this operation, so it is disabled.

3. Click the **Send Request** button.
4. The summary of returned checks appears in the area highlighted in green in the image above.

5. To view the complete details of returned checks, open the LastResponseTransaction.xml file that is created by this demo client application in the folder where this application is installed.
Get Check Detail

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

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

2. Enter the Check Number and Check Sequence Number fields. Other fields, such as Employee Object Number, RVC Object Number, and so on are not applicable to this operation.

3. Click the Send Request button.

The area highlighted in green appears if the check has extensibility data. To see the entire check detail, open the LastResponseTransaction.xml file that is created by this demo client application in the folder where this application is installed.
Get Printed Texts of a Guest Check

Use the Get Printed Check button to open posted checks by supplying the check number and the tender number.

1. If you are using versions earlier than Simphony 2.7 MR3, enter the License Activation Code.
2. Click the Get Printed Check button.
3. Enter the **Employee number**, **Check Number** (or Check Sequence), and **Tender Media Object Number**, and then click **OK**.
Get Configured Information

1. Enter the RVC Object Number value in the Rvc # field.

2. If you are using versions earlier than Simphony 2.7 MR3, enter the License Activation Code.

3. Click the Get Config Info button.

4. Enter the Employee Object Number in the Employee number field, and then enter configuration data type IDs in the Configuration Number textbox. The configuration numbers should be separated by a comma when data for more than one configuration data type needs to be fetched.

5. If a specific range of records are required for given configuration data type, select the Get records by index? checkbox. Specify the start index and maximum number of records required in the Start Index and Max Records Count textboxes. When the checkbox Get records by index? is not selected, the TS API returns all records for the specified configuration data types.

6. Click the OK button to send a request to the TS API.
The following image shows the configuration data returned by TS API.
Tax Override

Use the Tax Override button to override a menu item’s existing tax rate. These items include menu items, combo meals, 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 enterprise or property level.

1. After adding a menu item to the check, click the Tax Override button.

2. Place a checkmark next to the tax rate to apply to this item.
3. Click the **Generate** button, and then click the **Save** button.

In the following image, 64 zeros appear 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. Clicking **Calculate Totals** will apply the new rate to the item and add the taxable amount to the **Tax Total** field.
Finally, the user should be able to tender the check and the check printout should display the correct taxed amount.