Oracle® Hospitality Suites Management

Suites API

Release 3.7 and later

E90897-02

May 2018

Contents

Preface

The Oracle Hospitality Suites Management API is an extension to the Suites Management Application. The API allows developers to create custom integrations to the Suites Application.

Audience

This document is intended to be used by software engineers developing applications that interface to the Suites Management API.

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 Suites Management v3.6 and later
- Understanding of application development concepts

Revision History

Date	Description of Change
May 2017	• Initial publication
October 2017	• Made grammar and formatting corrections
May 2018	• Updated title page to include later versions

Glossary

The following acronyms and abbreviations are used within this document:

Acronym / Abbreviation	Full Text
API	Application Programming Interface
IIS	Microsoft Internet Information Services
POS	Point of Sale
SQL	Microsoft Structured Query Language
CC	Credit Card
PC	Personal Computer
DB	Database
LSM	Suites Management application DB
MI	Menu Item(s)

Compatible Versions

This release of the Suites API requires Suites Management version 3.7 or higher.

Naming Conventions

Operation names appear in bold **blue**. Example: **GetAllData**, **GetOwners, …**

Links to operation names or structures appear in light blue. Example: GetAllData, Owner, Event, …

Data types defined by the WSDL appear in blue. Example: Owner, string, long, …

Parameters, variables and values for data types appear in red accent italics. Example: *pRequest, 123, ...*

1 Introduction and Overview

Suites API is a set of two web services (Configuration Data and Order Processing) that help integrating a third-party application to Suites Management system. It leverages core business functionalities of Suites Management system and exposes them via web methods for third-party integration. These web services provide methods to perform the following operations.

1. Retrieve all necessary information from the Suites Management system pertinent to creation of a Suites Order.

   This information can be separated in 4 main groups:

   - Events Data
   - Owner Data
   - Suite Information
   - Menu Item Information

2. Create an Order for a Suite Owner
3. Modify menu items on an open Suites order
4. Void menu item from an open Suites order
5. Retrieve summary of open orders

All Suites Orders posted via the Suites API can be opened in Suites Management.

Below are a few sample business scenarios in which Suites API could be used to integrate to a third-party application.

- Remote Suites ordering from a kiosk or mobile phone application
- Remote ordering or centralized order dispatch via a web application

Suites API web services are installed on the same machine where Suites Management application is installed and running.

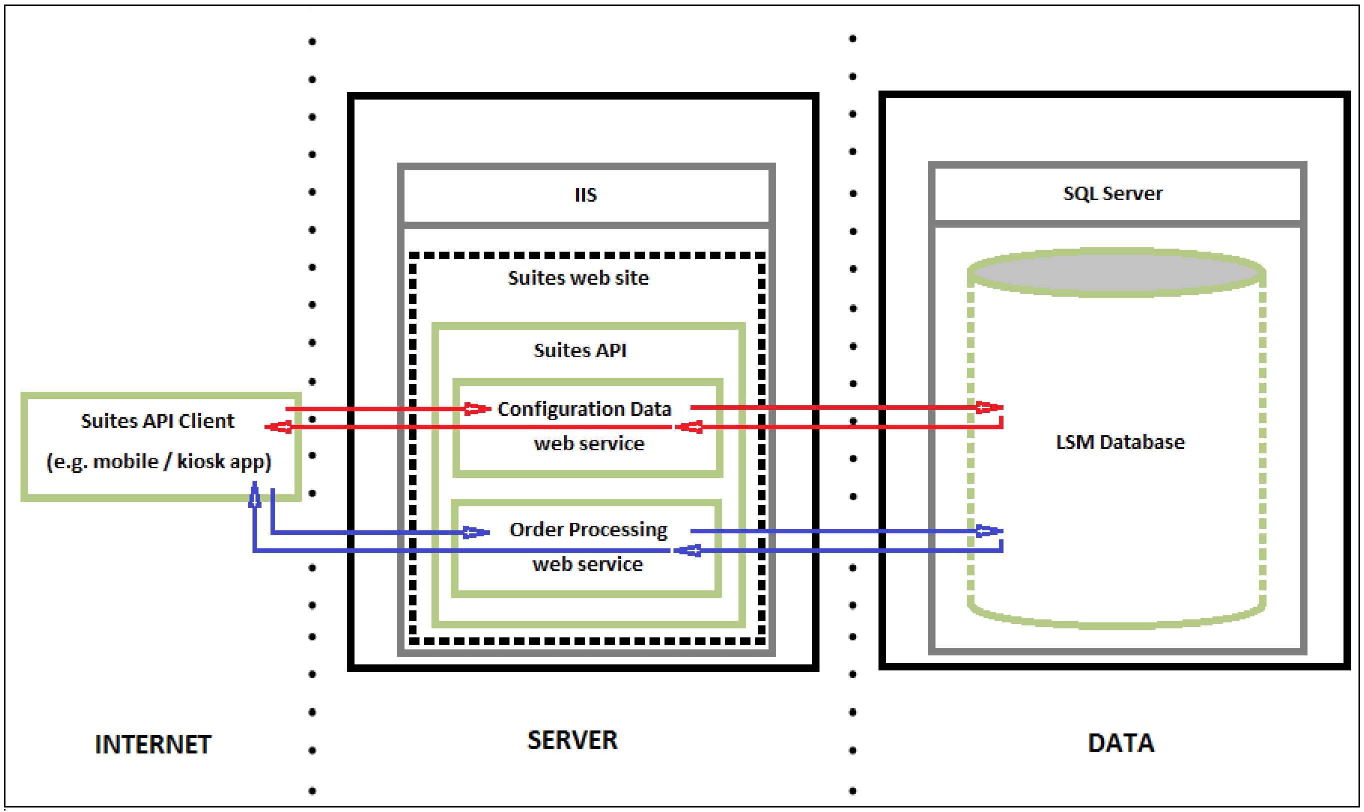
2 Suites Management Architecture

Suites API consists of two web services that are part of Suites Management web site hosted by IIS. Suites API web services use the same underlying business libraries that are being used by the Suites Management application for all the Suites operations.

The Suite Owner Order created via Suites API is posted to an enterprise database (LSM) for reporting purposes. All transactions made via Suites API web services are stored in a Suites Management database named LSM.

Any configuration changes made in the Suites Management application at (enterprise) server would be available to the Suites API client the moment the changes are saved.

The picture depicts the data flow between a Suites API client application and an LSM database.



3 Developing with the Suites API

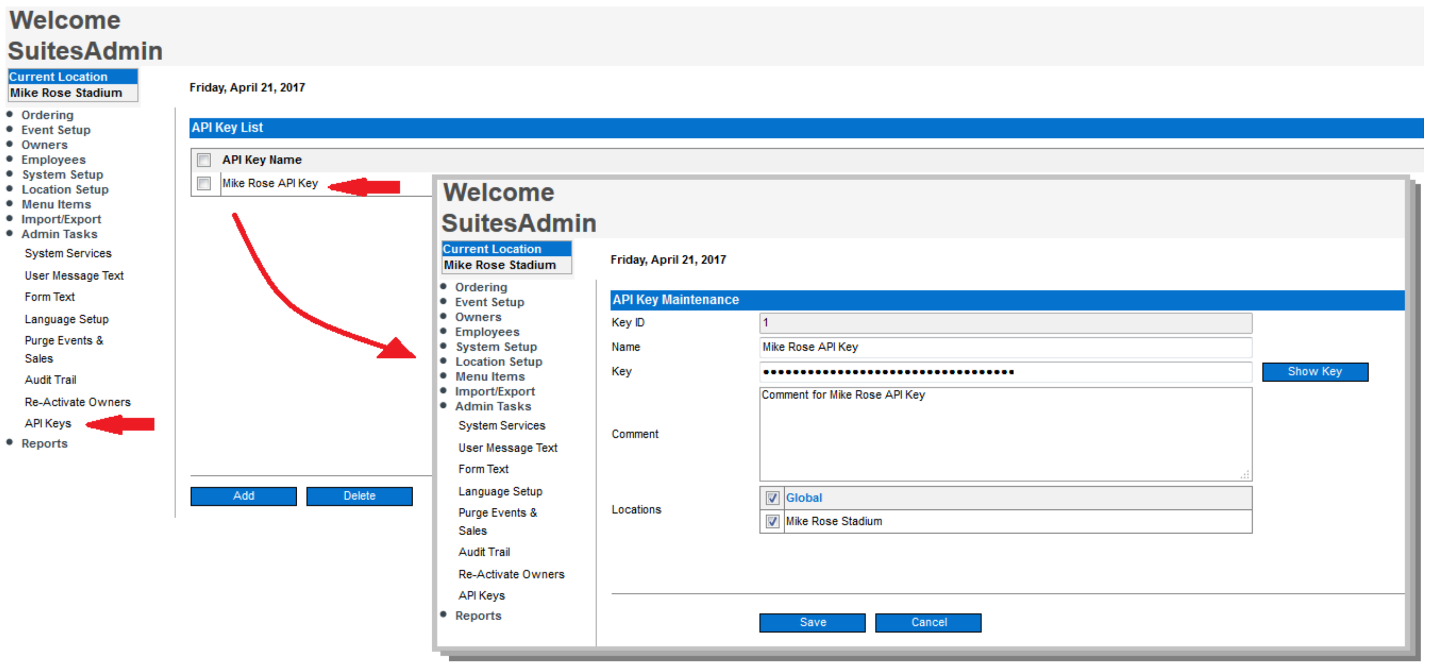
Overview

Calling Conventions

There is only one type of parameter passed to the Suites API: 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.

API Key Setup

An application consuming the Suites Managament API requires an API Key to communicate. The Administrator can configure the system and generate an API Key by accessing the Suites Managements' API Keys web page. The client application must authenticate by providing the API key in order to be able to consume the API provided services.



Friday, April 21, 2017

API Key List

	API Key Name
☐	Mike Rose API Key

Friday, April 21, 2017

API Key Maintenance

Key ID	1
Name	Mike Rose API Key
Key	••••••••••••••••••••••••••••••••••
Comment	Comment for Mike Rose API Key
Locations	✓ Global
	✓ Mike Rose Stadium

4 Function Reference

Configuration Data Operations

Suites Configuration Data API provides support for nine different configuration related operations. One web method is exposed to support each of those operations. All those web methods are explained in the table below.

Summary

ClientAuthenticatorResponse **ClientAuthenticator** (ClientAuthenticator pRequest) *This function authenticates the client for the Suites Configuration API. The information is presented as a single response xml file to the requestor.*
GetAllDataResponse **GetAllData** (GetAllData pRequest) *This function extracts all-of-the-data for all of the locations from Suites Management database, known as LSM db. By 'all-of-the-data' we mean: locations, events, event types, delivery times, suites, sections, major groups, family groups, menu item categories, menu price levels, menu items, owners, bar par levels and questions. The extracted information is presented as a single response xml file to the requestor.*
GetEventsResponse **GetEvents** (GetEvents pRequest) *This function extracts all, some or just a single Event from Suites Management database, depending of the request parameters.*
GetOwnersResponse **GetOwners** (GetOwners pRequest) *This function extracts all, some or just a single Owner from Suites Management DB, depending of the request parameters.*
GetSuitesResponse **GetSuites** (GetSuites pRequest) *This function extracts all, some or just a single Suite from Suites Management DB, depending of the request parameters.*
GetMenuItemsResponse **GetMenuItems** (GetMenuItems pRequest) *This function extracts all, some or just a single Menu Item from Suites Management DB, depending of the request parameters.*
GetBarParLevelResponse **GetBarParLevel** (GetBarParLevel pRequest) *Not supported function.*
BarParLevelUpdateResponse **BarParLevelUpdate** (BarParLevelUpdate pRequest) *Not supported function.*
OwnerUpdateResponse **OwnerUpdate** (OwnerUpdate pRequest) *This function updates an existing Owner record from Suites Management database.*

Client Authentication

```
public ClientAuthenticatorResponse ClientAuthenticator (ClientAuthenticator
pRequest)
```

Business purpose

API client needs to authenticate in order to use retrieve data operations.

Method description

This method returns authentication status data for API client. The caller will have to call this method with pRequest attributes fully specified in order to retrieve his or her Configuration API operations availability status.

Given below are the list of authentication data structures that can be retrieved using ClientAuthenticator method.

1. OperationResult
2. Session ID
3. IsAuthenticated
4. Location's ID list

Parameter

pRequest Structure to hold *the ClientAuthenticator request (input parameter)*

ClientAuthenticatorRequest (*pRequest*)

Name	Type	Data Type	Use	Comments
apiPass	Element	string	Req	Indicates client API Key value.

Return value

ClientAuthenticatorResponse. Data result encapsulated in *ClientAuthenticator* response structure.

ClientAuthenticatorResponse

Name	Type	Data Type	Use	Comments
operationResult	Element	OperationResult	Req	Indicates request id, req. processing time, req. processing time specified, response error code, resp. error message
SessionID	Element	string	Req	Indicates a session id.
IsAuthenticated	Element	boolean	Req	Indicates True or False.
LocationsID	Element	string	Req	String containing the locationIDs for which client is authenticated.

Get All Data

```
public GetAllDataResponse  GetAllData(GetAllData pRequest)
```

Business purpose

Enterprise client wants to retrieve data for all the locations from Suites Management's LSM enterprise ready database.

Method description

This method returns configuration data for all the locations found in the Suites LSM database. The caller will have to call this method with ***pRequest*** attribute **locationIdSpecified** set to ***false*** in order to retrieve complete list of records.

Given below are the list of configuration data structures that can be retrieved using GetAllData method.

1. Locations
2. Events
3. Event Types
4. Delivery Times
5. Suites
6. Sections
7. Major Groups
8. Family Groups
9. Menu Item Categories
10. Menu Prices Levels
11. Menu Items
12. Owners
13. Bar Par Levels
14. Questions

Parameter

pRequest Structure to hold the *GetAllData request (input parameter)*

GetAllDataRequest (*pRequest*)

Name	Type	Data Type	Use	Comments
locationId	Element	long	Req	Indicates location id.
locationIdSpecified	Element	bool	Req	Indicates if location id is specified

Return value

GetAllDataResponse. Data result encapsulated in *GetAllData* response structure.

GetAllDataResponse

Name	Type	Data Type	Use	Comments
operationResult	Element	OperationResult	Req	Indicates request id, request processing time, request processing time specified, response error code, response error message
locations	Element	Location	Req	Indicates location info
events	Element	Event	Req	Indicates event info.
eventTypes	Element	EventType	Req	Indicates event type info.
deliveryTimes	Element	DeliveryTime	Req	Indicates delivery time info.
suites	Element	Suite	Req	Indicates suite info.
sections	Element	Section	Req	Indicates section info.
majorGroups	Element	MajorGroup	Req	Indicates major group info.
familyGroups	Element	FamilyGroup	Req	Indicates family group info.
menuItemCategories	Element	MenuItemCategory	Req	Indicates list of Menu Item Categories.

Name	Type	Data Type	Use	Comments
menuPriceLevels	Element	MenuPriceLevel	Req	Indicates list of Menu Item Price Levels.
menuItems	Element	MenuItem	Req	Indicates menu item info.
owners	Element	Owner	Req	Indicates owner info.
barParLevels	Element	BarParLevel	Req	Indicates list of Bar Par Levels.
questions	Element	Question	Req	Indicates list of available questions.

Get Events

```
public GetEventsResponse  GetEvents(GetEvents pRequest)
```

Business purpose

Gives ability to the client to retrieve all events, events for a specific location or a single specified event from Suites Management's LSM database.

Method description

This method returns event related data for: - all the locations, - a specified location, or - a single event found in the Suites LSM database. The caller will have to call this method with ***pRequest*** attribute locationIdSpecified set to ***false*** and empty value for eventId attribute in order to retrieve complete list of all the existing event records.

For the list of events in the specific location, the ***pRequest*** attribute locationIdSpecified must be set to ***true*** and existing value for locationId attribute must be provided. To get just one event related data, the eventId attribute value must be of an existing event, and the locationIdSpecified attribute value should be set to ***false.***

Parameter

pRequest Structure to hold the *GetEvents request (input parameter)*

GetEventsRequest (*pRequest*)

Name	Type	Data Type	Use	Comments
eventId	Element	string	Req	Indicates event id.
locationId	Element	long	Req	Indicates location id.
locationIdSpecified	Element	bool	Req	Indicates if location id is specified.

Return value

GetEventsResponse. Data result encapsulated in *GetEvents* response structure.

GetEventsResponse

Name	Type	Data Type	Use	Comments
operationResult	Element	OperationResult	Req	Indicates request id, request processing time, request processing time specified, response error code, response error message
events	Element	Event	Req	Indicates event info.

Get Owners

```
public GetOwnersResponse GetOwners(GetOwners pRequest)
```

Business purpose

Gives ability to the client to retrieve all owners, owners from a specific location or a single specified owner from Suites Management's LSM database.

Method description

This method returns owner related data for: - all the locations, - a specified location, or - a single owner found in the Suites LSM database. The caller will have to call this method with ***pRequest*** attribute locationIdSpecified set to ***false*** and empty value for ownerId attribute in order to retrieve complete list of all the existing owner records.

For the list of owners in the specific location, the ***pRequest*** attribute locationIdSpecified must be set to ***true*** and existing value for locationId attribute must be provided. To get just one owner related data, the ownerId attribute value must be of an existing owner, and the locationIdSpecified attribute value should be set to ***false.***

Parameter

pRequest Structure to hold the *GetOwners request (input parameter)*

GetOwnersRequest (*pRequest*)

Name	Type	Data Type	Use	Comments
ownerId	Element	string	Req	Indicates owner id.
locationId	Element	long	Req	Indicates location id.
locationIdSpecified	Element	bool	Req	Indicates if location id is specified.

Return value

GetOwnersResponse. Data result encapsulated in *GetOwners* response structure.

GetOwnersResponse

Name	Type	Data Type	Use	Comments
operationResult	Element	OperationResult	Req	Indicates request id, request processing time, request processing time specified, response: error code and error message
owners	Element	Owner	Req	Indicates owner info.

Get Suites

```
public GetSuitesResponse  GetSuites(GetSuites pRequest)
```

Business purpose

Clients wants the ability to retrieve entire suites data, suites from a specific location or a single specified suite from Suites Management's LSM database.

Method description

This method returns suite related data for: - all the locations, - a specified location, or - a single suite found in the Suites LSM database. The requestor will have to call this method with ***pRequest*** attribute locationIdSpecified set to ***false*** and empty value for suiteId attribute in order to retrieve complete list of all the existing suite records.

For the list of suites in the specific location the ***pRequest*** attribute locationIdSpecified must be set to ***true*** and existing value for locationId attribute must be provided. To get a single suite related data, the suiteId attribute value must be of an existing suite, and the locationIdSpecified attribute value should be set to ***false.***

Parameter

pRequest Structure to hold the *GetSuites request (input parameter)*

GetSuitesRequest (*pRequest*)

Name	Type	Data Type	Use	Comments
suiteId	Element	string	Req	Indicates suite id.
locationId	Element	long	Req	Indicates location id.
locationIdSpecified	Element	bool	Req	Indicates if location id is specified.

Return value

GetSuitesResponse. Data result encapsulated in *GetSuites* response structure.

GetSuitesResponse

Name	Type	Data Type	Use	Comments
operationResult	Element	OperationResult	Req	Indicates request id, request processing time, request processing time specified, response: error code and error message
suites	Element	Suite	Req	Indicates suite info.

Get Menu Items

```
public GetMenuItemsResponse  GetMenuItems(GetMenuItems pRequest)
```

Business purpose

Clients want the ability to retrieve entire menu items data, menu items from a specific location or a single specified menu item from Suites Management's LSM database.

Method description

This method returns to the requestor menu item related data for: - all the locations, - a specified location, or - a single menu item found in the Suites LSM database. The requestor will have to call this method with ***pRequest*** attribute locationIdSpecified set to ***false*** and empty value for menuItemId attribute in order to retrieve complete list of all the existing menu item records.

For the list of menu items in the specific location the ***pRequest*** attribute locationIdSpecified must be set to ***true*** and existing value for locationId attribute must be provided. To get a single menu item related data, the menuItemId attribute value must be of an existing menu item, and the locationIdSpecified attribute value should be set to ***false.***

Parameter

pRequest Structure to hold the *GetMenuItems request (input parameter)*

GetMenuItemsRequest (*pRequest*)

Name	Type	Data Type	Use	Comments
menuItemId	Element	long	Req	Indicates menu item id.
locationId	Element	long	Req	Indicates location id.
locationIdSpecified	Element	bool	Req	Indicates if location id is specified.

Return value

GetMenuItemsResponse. Data result encapsulated in *GetMenuItems* response structure.

GetMenuItemsResponse

Name	Type	Data Type	Use	Comments
operationResult	Element	OperationResult	Req	Indicates request id, request processing time, request processing time specified, response: error code and error message
menuItems	Element	MenuItem	Req	Indicates menu item info.

Owner Update

```
public OwnerUpdateResponse OwnerUpdate (OwnerUpdate pRequest)
```

Business purpose

The client would like to update the record of a suite owner, for example, address modifications or a status change.

Method description

This method is used to modify an existing suite owner information. The caller has to pass the details of the owner record as part of the request that would need to be applied to the suite owner record in the LSM database.

The method’s response will return the status of the update operation together with the owner structure containing the newly made changes, assuming a successful operation. Note that this method will not create any new owner records in the LSM database.

Parameter

pRequest Structure to hold the UpdateOwner request (input parameter)

UpdateOwnerRequest (*pRequest*)

Name	Type	Data Type	Use	Comments
owner	Element	Owner	Req	Indicates owner info.

Return value

UpdateOwnerResponse. Data result encapsulated as *UpdateOwner* response structure.

UpdateOwnerResponse

Name	Type	Data Type	Use	Comments
operationResult	Element	OperationResult	Req	Indicates request id, request processing time, request processing time specified, response error code, response error message
owner	Element	Owner	Req	Indicates owner info.

Suites Order Processing Operations

Suites Order Processing API provides support for eight order related operations. One web method is exposed to support each of those operations. All those web methods are explained in the table below.

Summary

ClientAuthenticatorResponse **ClientAuthenticator** (ClientAuthenticator pRequest) *This function authenticates the client for the Suites Order Processing API. The information is presented as a single response xml file to the requestor.*
EventOrderCreateResponse **EventOrderCreate** (EventOrderCreate pRequest) *This function alowes Suites API client to submit a request for event order creation. The request will cary all the needed information and the Order Processing API will use it in an attempt to create an event order. A response will be sent back to the client with the status of the create operation and if successful with the order header id of the newly created event order.*
EventOrderDeleteResponse **EventOrderDelete** (EventOrderDelete pRequest) *This function alowes Suites API client to submit a request for event order deletion. The request will cary all the needed information and the Order Processing API will use it in an attempt to delete an event order. A response will be sent back to the client with the status of the delete operation.*
EventOrderUpdateResponse **EventOrderUpdate** (EventOrderUpdate pRequest) *Suites API client can submit a request for event order update. The request will cary all the needed information and the Order Processing API will use it in an attempt to update respective event order. A response will be sent back to the client with the status of the update operation.*
GetEventOrdersResponse **GetEventOrders** (GetEventOrders pRequest) *Gets summary of both open and closed guest checks after applying given filter condition*
GetStandingOrdersResponse **GetStandingOrders** (GetStandingOrders pRequest) *Gets completes details of a guest check in xml format*
StandingOrderCreateResponse **StandingOrderCreate** (StandingOrderCreate pRequest) *This function alowes Suites API client to submit a request for standing order creation. The request will cary all the needed information and the Order Processing API will use it in an attempt to create a standing order. A response is sent back to the client with the status of the create operation and if successful with the order header id of the newly created standing order.*
StandingOrderDeleteResponse **StandingOrderDelete** (StandingOrderDelete pRequest) *This function alowes Suites API client to submit a request for standing order deletion. The request will cary all the needed information and the Order Processing API will use it in an attempt to delete a standing order. A response is sent back to the client with the status of the delete operation.*
StandingOrderUpdateResponse **StandingOrderUpdate** (StandingOrderUpdate pRequest)

Suites API client can submit a request for standing order update. The request will cary all the needed information and the Order Processing API will use it in an attempt to update respective standing order. A response will be sent back to the client with the status of the update operation.

EventOrderCreate

```
public EventOrderCreateResponse EventOrderCreate (EventOrderCreate pRequest)
```

Business purpose

Allows Suites API client to create new event order for suite Owner.

Method description

This method is used to create a new event order inside the LSM database. The request object will have the appropriate information needed for the successful event order record creation. Information will include:

- *order header* values like owner id, suite id, event id, seat id, subtotal, contact info, …

- *order details* values identifying menu item, quantity, price level, delivery time id…

- *answer* values used as instructions for event order preparation and delivery.

Parameter

pRequest Structure to hold the EventOrderCreate request (input parameter)

EventOrderCreateRequest (*pRequest*)

Name	Type	Data Type	Use	Comments
eventOrderHeader	Element	EventOrderHeader	Req	Indicates event order header info.
eventOrderDetail	Element	EventOrderDetail	Req	Indicates event order detail info.
answer	Element	Answer	Req	Indicates event order answer info.

Return value

EventOrderCreateResponse. Data result encapsulated in *EventOrderCreate* response structure.

EventOrderCreateResponse

Name	Type	Data Type	Use	Comments
operationResult	Element	OperationResult	Req	Indicates request id, request processing time, request processing time specified, response error

Name	Type	Data Type	Use	Comments
				code, response error message
eventOrder	Element	EventOrder	Req	Indicates event order info.

EventOrderDelete

```
public EventOrderDeleteResponse  EventOrderDelete ( EventOrderDelete pRequest )
```

Business purpose

Allows Suites API client to delete an existing event order related to suite Owner.

Method description

This method is used to delete an existing event order inside the LSM database. The request object will have the single piece of information needed for the successful deletion of an event order record and that is the order id.

Please be aware that this method is deleting the entire event order that consists of a header and one or more detail records from LSM database. In this case by deleting we mean removing the record from database table, not just marking (flagging) the record as deleted.

Parameter

pRequest Structure to hold the EventOrderDelete request (input parameter)

EventOrderDeleteRequest (*pRequest*)

Name	Type	Data Type	Use	Comments
orderId	Element	string	Req	Indicates order id info.

Return value

EventOrderDeleteResponse. Data result encapsulated in *EventOrderDelete* response structure.

EventOrderDeleteResponse

Name	Type	Data Type	Use	Comments
operationResult	Element	OperationResult	Req	Indicates request id, request processing time, request processing time specified, response error code, response error message

EventOrderUpdate

public EventOrderUpdateResponse EventOrderUpdate (EventOrderUpdate *pRequest*)

Business purpose

Allows Suites API client to update existing event order related to suite Owner.

Method description

This method is used to update an existing event order inside the LSM database. The request object will have the appropriate information needed for the successful update of an event order record. Information will include:

- modified *order header* values like subtotal, contact info, …

- modified or removed *order details* values like menu item, quantity, price level, delivery time id…

- modified or removed *answer* values used as instructions for event order preparation.

Parameter

pRequest Structure to hold the EventOrderUpdate request (input parameter)

EventOrderUpdateRequest (*pRequest*)

Name	Type	Data Type	Use	Comments
eventOrderHeader	Element	EventOrderHeader	Req	Indicates event order header info.
eventOrderDetail	Element	EventOrderDetail	Req	Indicates event order detail info.
answer	Element	Answer	Req	Indicates event order answer info.

Return value

EventOrderUpdateResponse. Data result encapsulated in EventOrderUpdate response structure.

EventOrderUpdateResponse

Name	Type	Data Type	Use	Comments
operationResult	Element	OperationResult	Req	Indicates request id, request processing time, request processing time specified, response error code, response error message
eventOrder	Element	EventOrder	Req	Indicates updated event order info.

GetEventOrders

```
public GetEventOrdersResponse  GetEventOrders (GetEventOrders pRequest)
```

Business purpose

User wants to see the event order data from LSM database. Retrieved data can be filtered by 5 request parameters depending on user needs.

Method description

This method was introduced to get the event order data from LSM database. The request parameter of this method is carrying 5 filters allowing the user to retrieve:

- all the event orders from a location,
- event orders for specified owner,
- event orders for specified suite,
- specific event order for owner,
- all the event orders for an event and
- any other filtering combination.

The retrieved event order data through this method can be used for user's reporting needs or it can be used as starting entry for some additional (last minute) event order modification.

Parameter

pRequest Structure to hold the GetEventOrders request (input parameter)

GetEventOrdersRequest (*pRequest*)

Name	Type	Data Type	Use	Comments
orderId	Element	string	Req	Indicates order id info.
eventId	Element	string	Req	Indicates event id info.
ownerId	Element	string	Req	Indicates owner id info.
suiteId	Element	string	Req	Indicates suite id info.
locationId	Element	long	Req	Indicates location id info.
locationIdSpecified	Element	boolean	Req	Indicates if location id is specified or not.

Return value

GetEventOrdersResponse. Data result encapsulated in GetEventOrders response structure.

GetEventOrdersResponse

Name	Type	Data Type	Use	Comments
operationResult	Element	OperationResult	Req	Indicates request id, request processing time, request processing time specified, response error code, response error message
eventOrder	Element	EventOrder	Req	Indicates event order info.

GetStandingOrders

```
public GetStandingOrdersResponse GetStandingOrders (GetStandingOrders pRequest)
```

Business purpose

Client wants to see the standing order data from LSM database. Retrieved data can be filtered by 5 request parameters depending on client needs.

Method description

This method was introduced to get the standing order data from LSM database. The request parameter of this method is carrying 5 filters allowing the user to retrieve:

- all the standing orders from a location,
- standing orders for specified owner,
- standing orders for specified suite,
- specific standing order for owner,
- all the standing orders for an event and
- any other filtering combination.

The retrieved standing order data through this method can be used for client's reporting needs or it can be used as starting entry for some additional standing order modification.

Parameter

pRequest Structure to hold the GetStandingOrders request (input parameter)

GetStandingOrdersRequest (*pRequest*)

Name	Type	Data Type	Use	Comments
orderId	Element	string	Req	Indicates order id info.
eventId	Element	string	Req	Indicates event id info.

ownerId	Element	string	Req	Indicates owner id info.
suiteId	Element	string	Req	Indicates suite id info.
locationId	Element	long	Req	Indicates location id info.
locationIdSpecified	Element	boolean	Req	Indicates if location id is specified or not.

Return value

GetStandingOrdersResponse. Data result encapsulated in GetStandingOrders response structure.

GetStandingOrdersResponse

Name	Type	Data Type	Use	Comments
operationResult	Element	OperationResult	Req	Indicates request id, request processing time, request processing time specified, response error code, response error message
standingOrder	Element	StandingOrder	Req	Indicates standing order info.

StandingOrderCreate

```
public StandingOrderCreateResponse StandingOrderCreate (StandingOrderCreate pRequest)
```

Business purpose

Allows Suites API client to create new standing order for suite Owner.

Method description

This method is used to create a new standing order inside the LSM database. The request object will have the appropriate information needed for the successful standing order record creation. Information will include:

- *order header* values like order id, owner id, event type id, comments, …

- *order details* values identifying menu item, quantity, price level, delivery time id…

- *answer* values used as instructions for standing order preparation and/or delivery.

Parameter

pRequest Structure to hold the StandingOrderCreate request (input parameter)

StandingOrderCreateRequest (*pRequest*)

Name	Type	Data Type	Use	Comments
standingOrderHeader	Element	StandingOrderHeader	Req	Indicates standing order header info.
standingOrderDetail	Element	StandingOrderDetail	Req	Indicates standing order detail info.
answer	Element	Answer	Req	Indicates standing order answer info.

Return value

StandingOrderCreateResponse. Data result encapsulated in StandingOrderCreate response structure.

StandingOrderCreateResponse

Name	Type	Data Type	Use	Comments
operationResult	Element	OperationResult	Req	Indicates request id, request processing time, request processing time specified, response error code, response error message
standingOrder	Element	StandingOrder	Req	Indicates standing order info.

StandingOrderDelete

```
public StandingOrderDeleteResponse StandingOrderDelete (StandingOrderDelete pRequest)
```

Business purpose

Allows Suites API client to delete an existing standing order related to suite Owner.

Method description

This method is used to delete an existing standing order inside the LSM database. The request object will have the single piece of information needed for the successful deletion of a standing order record and that is the order id.

Please be aware that this method is deleting the entire standing order that consists of a header and one or more detail records from LSM database. In this case by deleting we mean removing the record from database table, not just marking (flagging) the record as deleted.

Parameter

pRequest Structure to hold the StandingOrderDelete request (input parameter)

StandingOrderDeleteRequest (*pRequest*)

Name	Type	Data Type	Use	Comments
orderId	Element	string	Req	Indicates order id info.

Return value

StandingOrderDeleteResponse. Data result encapsulated in StandingOrderDelete response structure.

StandingOrderDeleteResponse

Name	Type	Data Type	Use	Comments
operationResult	Element	OperationResult	Req	Indicates request id, request processing time, request processing time specified, response error code, response error message

StandingOrderUpdate

```
public StandingOrderUpdateResponse  StandingOrderUpdate (StandingOrderUpdate pRequest)
```

Business purpose

Allows Suites API client to update existing standing order related to suite Owner.

Method description

This method is used to update an existing standing order inside the LSM database. The request object will have the appropriate information needed for the successful update of a standing order record. Information will include:

- modified *order header* values like comments, event type id, …

- modified or removed *order details* values like menu item, quantity, price level, delivery time id…

-modified or removed *answer* values used as instructions for standing order preparation.

Parameter

pRequest Structure to hold the StandingOrderUpdate request (input parameter)

StandingOrderUpdateRequest (*pRequest*)

Name	Type	Data Type	Use	Comments

standingOrderHeader	Element	StandingOrderHeader	Req	Indicates standing order header info.
standingOrderDetail	Element	StandingOrderDetail	Req	Indicates standing order detail info.
answer	Element	Answer	Req	Indicates standing order answer info.

Return value

StandingOrderUpdateResponse. Data result encapsulated in StandingOrderUpdate response structure.

StandingOrderUpdateResponse

Name	**Type**	**Data Type**	**Use**	**Comments**
operationResult	Element	OperationResult	Req	Indicates request id, request processing time, request processing time specified, response error code, response error message
standingOrder	Element	StandingOrder	Req	Indicates updated standing order info.

5 Class Reference

Configuration Data Class Summary

ClientAuthenticator	Structure that represents client authentication for API services.
Event	This structure represents a single Suites Event.
EventType	Represents a type of Suites Event (i.e. Basketball, Baseball, Concert, etc...).
DeliveryTime	Structure that defines a point in time during a Suites Event when items will be delivered to the Suite/Seat.
Location	This structure represents a venue (i.e. Stadium, Arena, Concert Hall, etc...).
Suite	Structure that represents a Suite location within a venue.
Section	Structure that represents a section of Seats within a venue.
Row	Structure that represents a row of Seats within a venue.
Seat	Represents a single Seat within a venue.
MajorGroup	Represents a top level categorization for menu items.
FamilyGroup	Represents a sub level categorization for menu items.
MenuItemCategory	Structure that holds a single level of categorization of menu items used during ordering.
MenuPriceLevel	Structure that defines a price level for a menu item.
MenuItem	Structure that represents a menu item.
MenuItemPrice	This structure represents the price of a menu item.
AssociateMenuItemLink	This structure represents the association of associate menu item to menu item.
AssociateMenuItem	Structure that defines an associate menu item.
Owner	Represents an entity (company, division, or individual) that can place orders for events.
Address	Structure that holds an address of an Owner.
BarParLevel	Represents a single Par value for a single menu item.

Question	Structure represents a question to be answered by the guest when placing an Order.
TextQuestionType	Structure represents a type of question which requires a textual answer to be entered.
ListQuestionType	Structure that represents the list of possible answers for a question that requires the user to pick the answer from a list.
AllDataRequest	Represents a request for configuration data for all Locations.
AllDataResponse	This structure contains all of the LSM data for all of the Locations.
LocationDataRequest	Represents a request for configuration data for a single Location.
LocationDataResponse	This structure contains all of the LSM data for a single Location.
AllMenuItemsRequest	Represents a request for all menu items for all Locations.
AllMenuItemsResponse	This structure contains all menu items for all Locations.
MenuItemsByLocationRequest	Represents a request for all menu items for a single Location.
MenuItemsByLocationResponse	This structure contains all the menu items for a single Location.
AllOwnersRequest	Structure represents a request for all the Owners.
AllOwnersResponse	This structure contains all Owners for all Locations.
OwnersByLocationRequest	Represents a request for all Owners for a single Location.
OwnersByLocationResponse	This structure contains all Owners for a single Location.

Order Processing Class Summary

CreditCard	This structure represents a credit card number and expiration date.
OrderContactInfo	Represents the contact information for a single Order.
Answer	Structure represents an answer to an Event Order question.
EventOrderHeader	This structure represents an Event Order summary.
EventOrderDetail	Structure represents an Event Order Detail line.
EventOrder	Represents a single Event Order for an Owner.
StandingOrderHeader	This structure represents a Standing Order summary.
StandingOrderDetail	Structure represents a Standing Order Detail line.

StandingOrder	Represents a single Standing Order for an Owner and event type.

Configuration Data Classes

ClientAuthenticator

This structure represents client authentication needed for consumption of API services.

Public Attributes

string **apiPass** ***apiPass*** *is a unique value that holds an API key for a client. The apiPass is generated by the system administartor and its assigned to one or more locations in the LSM databse.*
string **sessionID** *Represents unique **sessionID** assigned to an authenticated client for a specific time length.*
boolean **isAuthenticated** *Represents the client status based of supplied apiPass.*
string **csvLocationID** *Represents a string holding comma separated location IDs that have assigned same API Key.*
boolean **isGlobal** *Identifies an apiPass (API Key) as a global key for entire organization.*

XML Instance Representation

```
<...>
<lsm:apiPass> xs:string (length = 36) </lsm:apiPass> [1]
<lsm:sessionID> xs:string (length = 36) </lsm:sessionID> [1]
<lsm:isAuthenticated> xs:boolean </lsm:isAuthenticated> [1]
<lsm:csvLocationID> xs:string (length <= 100) </lsm:csvLocationID> [1]
<lsm:isGlobal> xs:bolean </lsm:isGlobal> [1]
</...>
```

Event

This structure is to encapsulate some details of an Event record.

Public Attributes

string **id** ***id*** *is a unique alphanumeric value that identifies an event in the LSM database. The event id is auto-generated by the system and its assigned to a newly created event.*
string **name** *Event **name** is a descriptive alphanumeric value that provides meaningful information for a particular event (e.i. "Luciano Pavarotti Concert" or "MLS Soccer – Game 1").*
long **typeId** *Represents the typeId of an event type (links the event to a certain event type).*

DateTime **dateTime** *Represents Date and Time of an event. Time should indicate the beginning of an event.*
boolean **isRestock** *Identifies an event as restock event or not. Restock events are meant for inventory replenishing.*
string **userdefined1** *1st free alphanumeric user defined entry field.*
string **userdefined2** *2nd free alphanumeric user defined entry field.*
string **userdefined3** *3rd free alphanumeric user defined entry field.*
long **locationId** ***locationId** is a unique numeric value that identifies the location for which the event was created.*

XML Instance Representation

```
<...>
<lsm:id> xs:string (length <= 20) </lsm:id> [1]
<lsm:name> xs:string (length <= 50) </lsm:name> [1]
<lsm:typeId> xs:long </lsm:typeId> [1]
<lsm:dateTime> xs:dateTime </lsm:dateTime> [1]
<lsm:isRestock> xs:boolean </lsm:isRestock> [1]
<lsm:userdefined1> xs:string (length <= 100) </lsm:userdefined1> [1]
<lsm:userdefined2> xs:string (length <= 100) </lsm:userdefined2> [1]
<lsm:userdefined3> xs:string (length <= 100) </lsm:userdefined3> [1]
<lsm:locationId> xs:long </lsm:locationId> [1]
</...>
```

EventType

This structure is to encapsulate some details of an Event Type record.

Public Attributes

long **id** ***id** is a unique numeric value that identifies an event type in the LSM database. The event type id is auto-generated by the system and its assigned to a newly created event type.*
string **name** *Event type **name** is a descriptive alphanumeric value that provides meaningful information for a particular event type (e.i. "INDOOR CONCERT", "MLS SOCCER GAME" or "COLLEGE FOOTBAL").*
long **locationId** ***locationId** is a unique numeric value that identifies the location for which the event type was created.*

XML Instance Representation

```
<...>
<lsm:id> xs:long </lsm:id> [1]
<lsm:name> xs:string (length <= 50) </lsm:name> [1]
<lsm:locationId> xs:long </lsm:locationId> [1]
</...>
```

DeliveryTime

This structure is to encapsulate details of a Delivery Time record.

Public Attributes

long **id**	***id*** *is a numeric value that uniquely identifies a delivery time in the LSM database. The delivery time id is auto-generated by the system and its assigned to a newly created delivery time record.*
string **name**	*Delivery time **name** is a descriptive alphanumeric value that provides meaningful information for a particular delivery time (e.i. "Pre-Game 30 minutes" or "At Half Game").*
long **locationId**	***locationId*** *is a unique numeric value that identifies the location for which the delivery time was created.*

XML Instance Representation

```
<...>
<lsm:id> xs:long </lsm:id> [1]
<lsm:name> xs:string (length <= 50) </lsm:name> [1]
<lsm:locationId> xs:long </lsm:locationId> [1]
</...>
```

Location

This structure is to encapsulate two details of a Location record.

Public Attributes

long **id**	***locationId*** *is a unique numeric value that identifies a location in the LSM database. The location id is auto-generated by the system and its assigned to a newly created location.*
string **name**	*Location **name** is a descriptive alphanumeric value that provides meaningful information for a particular location (e.i. "Wembley Stadium" or "AAA Arena").*

XML Instance Representation

```
<...>
<lsm:id> xs:long </lsm:id> [1]
<lsm:name> xs:string (length <= 50) </lsm:name> [1]
</...>
```

Suite

This structure is to encapsulate the details of a Suite record.

Public Attributes

string **id** ***id*** *is a unique alphanumeric value that identifies a suite in the LSM database. The suite id is auto-generated by the system and its assigned to a newly created suite record.*
string **number** ***number*** *is a unique numeric value that identifies a suite in the LSM database. The suite number is assigned by the Suites Administrator when creating a suite record.*
string **name** *Suite* ***name*** *is a descriptive alphanumeric value that provides meaningful information for a particular suite (e.i. "Top Level – Suite 100", "VIP Suites – Unit 10" or "Suite #1").*
boolean **allowPreorder** *Identifies if a suite allows preordering or not.*
boolean **isVIP** *Identifies if a suite is VIP or not. VIP suites may require additional attention.*
string **userdefined1** *1st free alphanumeric user defined entry field.*
string **userdefined2** *2nd free alphanumeric user defined entry field.*
string **userdefined3** *3rd free alphanumeric user defined entry field.*
long **locationId** ***locationId*** *is a unique numeric value that identifies the location in which the suite was created.*

XML Instance Representation

```
<...>
<lsm:id> xs:string (length <= 20) </lsm:id> [1]
<lsm:number> xs:string (length <= 20) </lsm:number> [1]
<lsm:name> xs:string (length <= 50) </lsm:name> [1]
<lsm:allowPreorder> xs:boolean </lsm:allowPreorder> [1]
<lsm:isVIP> xs:boolean </lsm:isVIP> [1]
<lsm:userdefined1> xs:string (length <= 100) </lsm:userdefined1> [1]
<lsm:userdefined2> xs:string (length <= 100) </lsm:userdefined2> [1]
<lsm:userdefined3> xs:string (length <= 100) </lsm:userdefined3> [1]
<lsm:locationId> xs:long </lsm:locationId> [1]
</...>
```

Section

This structure is to encapsulate the details of a Section record.

Public Attributes

long **id**	***id*** *is a unique numeric value that identifies a section in the LSM database. The section* ***id*** *is auto-generated by the system.*
string **name**	*Section* ***name*** *is a descriptive alphanumeric value that provides meaningful information for a particular section.*
long **locationId**	***locationId*** *is a unique numeric value that identifies the location in which the section was created.*
ARRAY(Row) **rows**	*List of a* **Row** *structure that is associated with the section record.*

XML Instance Representation

```
<...>
<lsm:id> xs:long </lsm:id> [1]
<lsm:name> xs:string (length <= 50) </lsm:name> [1]
<lsm:locationId> xs:long </lsm:locationId> [1]
<lsm:rows> [1]
<lsm:row> lsm:Row </lsm:row> [0..*]
</lsm:rows>
</...>
```

Row

This structure is to encapsulate the details of a Row record.

Public Attributes

long **id**	***id*** *is a unique numeric value that identifies a row in the LSM database.*
string **name**	*Row* ***name*** *is a descriptive alphanumeric value that provides meaningful information for a particular row.*
ARRAY(Seat) **seats**	*List of a* **Seat** *structure that is associated with the row record*

XML Instance Representation

```
<...>
<lsm:id> xs:long </lsm:id> [1]
<lsm:name> xs:string (length <= 20) </lsm:name> [1]
<lsm:seats> [1]
<lsm:seat> lsm:Seat </lsm:seat> [0..*]
</lsm:seats>
</...>
```

Seat

This structure is to encapsulate the details of a Seat record.

Public Attributes

long **id**	
id *is a unique numeric value that identifies a seat in the LSM database.*	
string **name**	
*Seat **name** is a descriptive alphanumeric value that provides meaningful information for a particular seat.*	
boolean **allowPreorder**	
Identifies if preordering is allowed or not for a seat.	

XML Instance Representation

```
<...>
<lsm:id> xs:long </lsm:id> [1]
<lsm:name> xs:string (length <= 20) </lsm:name> [1]
<lsm:allowPreorder> xs:boolean </lsm:allowPreorder> [1]
</...>
```

MajorGroup

This structure is to encapsulate the details of a Major Group record.

Public Attributes

long **id**	
id *is a unique numeric value that identifies a major group in the LSM database.*	
string **name**	
name *is a descriptive alphanumeric value that provides meaningful information for a particular major group.*	
long **locationId**	
locationId *is a unique numeric value that identifies the location for which the major group was created.*	

XML Instance Representation

```
<...>
<lsm:id> xs:long </lsm:id> [1]
<lsm:name> xs:string (length <= 50) </lsm:name> [1]
<lsm:locationId> xs:long </lsm:locationId> [1]
</...>
```

FamilyGroup

This structure is to encapsulate the details of a Family Group record.

Public Attributes

long **id**

__id__ is a unique numeric value that identifies a family group in the LSM database.

string **name**

__name__ is a descriptive alphanumeric value that provides meaningful information for a particular family group.

long **locationId**

__locationId__ is a unique numeric value that identifies the location for which the family group was created.

XML Instance Representation

```
<...>
<lsm:id> xs:long </lsm:id> [1]
<lsm:name> xs:string (length <= 50) </lsm:name> [1]
<lsm:locationId> xs:long </lsm:locationId> [1]
</...>
```

MenuItemCategory

This structure is to encapsulate the details of a Menu Item Category record.

Public Attributes

long **id**

__id__ is a unique numeric value that identifies a menu item category in the LSM database.

string **name**

__name__ is a descriptive alphanumeric value that provides meaningful information for a particular category.

long **locationId**

__locationId__ is a unique numeric value that identifies the location in which the MI category was created.

XML Instance Representation

```
<...>
<lsm:id> xs:long </lsm:id> [1]
<lsm:name> xs:string (length <= 50) </lsm:name> [1]
<lsm:locationId> xs:long </lsm:locationId> [1]
</...>
```

MenuPriceLevel

This structure is to encapsulate the details of a Menu Item Price Level record.

Public Attributes

long **id**	***id*** *is a unique numeric value that identifies a menu item price level in the LSM database.*
string **name**	***name*** *is a descriptive alphanumeric value that provides meaningful information for a particular price level.*
long **locationId**	***locationId*** *is a unique numeric value that identifies the location in which the MI price level was created.*

XML Instance Representation

```
<...>
<lsm:id> xs:long </lsm:id> [1]
<lsm:name> xs:string (length <= 50) </lsm:name> [1]
<lsm:locationId> xs:long </lsm:locationId> [1]
</...>
```

MenuItem

This structure is to encapsulate the details of a Menu Item record.

Public Attributes

long **id**	***id*** *is a unique numeric value that identifies a menu item in the LSM database.*
string **name**	***name*** *is a descriptive alphanumeric value that provides meaningful information for a particular menu item.*
string **nlu**	***nlu*** *is a numeric value that links a menu item with a barcode. Note: this is numeric value not alphanumeric.*
long **categoryId**	***categoryId*** *is a unique numeric value that identifies the category of the menu item.*
long **majorGroupId**	***majorGroupId*** *is a unique numeric value that identifies the major group of the menu item.*
long **familyGroupId**	***familyGroupId*** *is a unique numeric value that identifies the family group of the menu item.*
long **locationId**	***locationId*** *is a unique numeric value that identifies the location in which the menu item was created.*
boolean **isFeaturedItem**	*Identifies if menu item is a featured item or not for an event.*
string **userdefined1**	*1st free alphanumeric user defined entry field.*

string **userdefined2** *2nd free alphanumeric user defined entry field.*
string **userdefined3** *3rd free alphanumeric user defined entry field.*
ARRAY(MenuItemPrice) **prices** *List of* **MenuItemPrice** *structure that defines details of prices linked to a menu item.*
ARRAY(AssociateMenuItemLink) **associateItems** *List of* **AssociateMenuItemLink** *structure that defines details of associate MI linked to a menu item.*

XML Instance Representation

```
<...>
<lsm:id> xs:long </lsm:id> [1]
<lsm:name> xs:string (length <= 50) </lsm:name> [1]
<lsm:nlu> xs:string (length <= 10) </lsm:nlu> [1] ?
<lsm:categoryId> xs:long </lsm:categoryId> [1]
<lsm:majorGroupId> xs:long </lsm:majorGroupId> [1]
<lsm:familyGroupId> xs:long </lsm:familyGroupId> [1]
<lsm:locationId> xs:long </lsm:locationId> [1]
<lsm:isFeaturedItem> xs:boolean </lsm:isFeaturedItem> [1] ?
<lsm:userdefined1> xs:string (length <= 100) </lsm:userdefined1> [1]
<lsm:userdefined2> xs:string (length <= 100) </lsm:userdefined2> [1]
<lsm:userdefined3> xs:string (length <= 100) </lsm:userdefined3> [1]
<lsm:prices> [0..1]
<lsm:price> lsm:MenuItemPrice </lsm:price> [1..8]
</lsm:prices>
<lsm:associateItems> [0..1]
<lsm:associateItem> lsm:AssociateMenuItemLink </lsm:associateItem> [1..*]
</lsm:associateItems>
</...>
```

MenuItemPrice

This structure is to encapsulate the details of a Menu Item Price record.

Public Attributes

decimal **priceLevelId** ***priceLevelId*** *is a unique numeric value that identifies a menu item price level in the LSM database.*
decimal **price** *Decimal value that represents the* ***price*** *of a menu item.*

XML Instance Representation

```
<...>
<lsm:priceLevelId> xs:decimal </lsm:priceLevelId> [1]
<lsm:price> xs:decimal </lsm:price> [1]
</...>
```

AssociateMenuItemLink

This structure is to encapsulate the details of an Associate Menu Item Link.

Public Attributes

long **associateMenuItemId** ***associateMenuItemId*** *is a unique numeric value that identifies the associate menu item in the link.*
int **quantity** ***quantity*** *is an integer value that represents quantitative presence of an associate menu item in the link.*

XML Instance Representation

```
<...>
<lsm:associateMenuItemId> xs:long </lsm:associateMenuItemId> [1]
<lsm:quantity> xs:int </lsm:quantity> [1]
</...>
```

AssociateMenuItem

This structure is to encapsulate the details of an Associate Menu Item record.

Public Attributes

long **id** ***id*** *is a unique numeric value that identifies an associate menu item in the LSM database.*
string **name** ***name*** *is a descriptive alphanumeric value that provides meaningful information for an associate menu item.*
long **majorGroupId** ***majorGroupId*** *is a unique numeric value that identifies the major group of the associate menu item.*
long **familyGroupId** ***familyGroupId*** *is a unique numeric value that identifies the family group of the associate menu item.*
long **locationId** ***locationId*** *is a unique numeric value that identifies the location in which the associate menu item was created.*

XML Instance Representation

```
<...>
<lsm:id> xs:long </lsm:id> [1]
<lsm:name> xs:string (length <= 50) </lsm:name> [1]
<lsm:majorGroupId> xs:long </lsm:majorGroupId> [1]
<lsm:familyGroupId> xs:long </lsm:familyGroupId> [1]
<lsm:locationId> xs:long </lsm:locationId> [1]
</...>
```

Owner

This structure is to encapsulate the details of suite Owner record.

Public Attributes

string **id**	***id*** *is a unique alphanumeric value that identifies an owner in the LSM database. The owner id is auto-generated by the system and its assigned to a newly created owner record.*
string **title**	*Owner's title (e.i. Honorable, Sir, Mayor,…)*
string **prefix**	*Owner's prefix (e.i. Mr., Ms.,…)*
string **firstName**	*This holds the owner's first name information.*
string **lastName**	*This holds owner's last name information.*
string **description**	*This holds the alternate value to describe the owner's account.*
string **company**	*This holds the owner's company name.*
string **accountNumber**	*This holds the account number associated with the owner.*
long **defaultDeliveryTimeId**	*This holds the id of a delivery time record associated with the owner.*
long **defaultMenuPriceLevelId**	*This holds the id of a price level record associated with the owner.*
long **locationId**	***locationId*** *is a unique numeric value that identifies the location in which the owner was created.*
string **email**	*This holds the email of the owner.*
boolean **paymentOnFile**	*Identifies if the owner has already payment on file or not.*
string **suiteOrSeat**	

This holds the list with value(s) 'SUITE'|'SEAT'|'BOTH' *that apply to owner.*
string **isGuestAccount** *Identifies if the owner account is classified as a guest account or not.*
boolean **canOrder** *Identifies if owner can order or not.*
string **userDefined1** *1st free alphanumeric user defined entry field.*
string **userDefined2** *2nd free alphanumeric user defined entry field.*
string **userDefined3** *3rd free alphanumeric user defined entry field.*
ARRAY(Address) **addresses** *List of a* **Address** *structure that is associated with the owner record.*

XML Instance Representation

```
<...>
<lsm:id> xs:string (length <= 20) </lsm:id> [1]
<lsm:title> xs:string (length <= 50) </lsm:title> [1]
<lsm:prefix> xs:string (length <= 50) </lsm:prefix> [1]
<lsm:firstName> xs:string (length <= 50) </lsm:firstName> [1]
<lsm:lastName> xs:string (length <= 50) </lsm:lastName> [1]
<lsm:description> xs:string (length <= 100) </lsm:description> [1]
<lsm:company> xs:string (length <= 50) </lsm:company> [1]
<lsm:accountNumber> xs:string (length <= 50) </lsm:accountNumber> [1]
<lsm:defaultDeliveryTimeId> xs:long </lsm:defaultDeliveryTimeId> [1]
<lsm:defaultMenuPriceLevelId> xs:long </lsm:defaultMenuPriceLevelId> [1]
<lsm:locationId> xs:long </lsm:locationId> [1]
<lsm:email> xs:string </lsm:email> [1]
<lsm:paymentOnFile> xs:boolean </lsm:paymentOnFile> [1]
<lsm:suiteOrSeat> xs:string (value comes from list: {'SUITE'|'SEAT'|'BOTH'})
</lsm:suiteOrSeat> [1]
<lsm:isGuestAccount> ... </lsm:isGuestAccount> [1]
<lsm:canOrder> xs:boolean </lsm:canOrder> [1]
<lsm:userdefined1> xs:string (length <= 100) </lsm:userdefined1> [1]
<lsm:userdefined2> xs:string (length <= 100) </lsm:userdefined2> [1]
<lsm:userdefined3> xs:string (length <= 100) </lsm:userdefined3> [1]
<lsm:addresses> [0..1]
<lsm:address> lsm:Address </lsm:address> [1..*]
</lsm:addresses>
</...>
```

Address

This structure is to encapsulate the details of an Address record related to a suite Owner.

Public Attributes

string **addressLine**	*Alphanumeric value for address line (e.i. 10974 Neptune Dr.).*
string **city**	*Textual value for a city name.*
string **stateProv**	*Textual value for state or province.*
string **postalCode**	*Alphanumeric value for postal code.*
string **country**	*Textual value for a country (e.i. Canada, Mexico, U.S.A.)*

XML Instance Representation

```
<...>
<lsm:addressLine> xs:string (length <= 50) </lsm:addressLine> [1..4]
<lsm:city> xs:string (length <= 50) </lsm:city> [1]
<lsm:stateProv> xs:string (length <= 20) </lsm:stateProv> [1]
<lsm:postalCode> xs:string (length <= 20) </lsm:postalCode> [1]
<lsm:country> xs:string (length <= 20) </lsm:country> [1]
</...>
```

Question

This structure is to encapsulate the details of a Question record.

Public Attributes

long **id**	***id** is a unique numeric value that identifies a question record in the LSM database. The **id** is auto-generated by the system.*
long **number**	***number** is a numeric value assigned to a question record. The number is entered by the Suites Administrator.*
string **name**	*Question **name** is a descriptive alphanumeric value that provides meaningful information about a particular question.*
string **text**	*Represents the **text** of a question (or question text it's the question itself).*
boolean **printOnConfirmation**	*Identifies if a question should print on confirmation or not.*

long **locationId** ***locationId*** *is a unique numeric value that identifies the location for which the question was created.*
ARRAY**(**TextQuestionType **text,** ListQuestionType **list)** **type** *List of* **TextQuestionType** *and* **ListQuestionType** *linked to the question record.*

XML Instance Representation

```
<...>
<lsm:id> xs:long </lsm:id> [1]
<lsm:number> xs:long </lsm:number> [1]
<lsm:name> xs:string (length <= 100) </lsm:name> [1]
<lsm:text> xs:string (length <= 100) </lsm:text> [1]
<lsm:printOnConfirmation> xs:boolean </lsm:printOnConfirmation> [1]
<lsm:locationId> xs:long </lsm:locationId> [1]
<lsm:type> [1]
Start Choice [1]
<lsm:text> lsm:TextQuestionType </lsm:text> [1]
<lsm:list> lsm:ListQuestionType </lsm:list> [1]
End Choice
</lsm:type>
</...>
```

TextQuestionType

This structure is to encapsulate the details of a Text Question Type record.

Public Attributes

string **defaultAnswer** *Indicates a textual value up to 200 characters in length as default answer for the Text Question Type.*

XML Instance Representation

```
<...>
<lsm:defaultAnswer> xs:string (length <= 200) </lsm:defaultAnswer> [1]
</...>
```

ListQuestionType

This structure is to encapsulate the details of a List Question Type record.

Public Attributes

long **id** ***id*** *is a unique numeric value that identifies a list question type record in the DB. The* ***id*** *is auto-generated by the system*

string **name** ***name*** *is a descriptive alphanumeric value that provides meaningful information for a list question type.*
string **description** ***description*** *is a textual value that provides meaningful information for about the list question type.*
ARRAY(long **id**, string **name**) **validValues** *List of* **id** *and* **name** *linked to the list question type record.*

XML Instance Representation

```
<...>
<lsm:id> xs:long </lsm:id> [1]
<lsm:name> xs:string (length <= 50) </lsm:name> [1]
<lsm:description> xs:string (length <= 200) </lsm:description> [1]
<lsm:validValues> [1]
<lsm:id> xs:long </lsm:id> [1]
<lsm:name> xs:string (length <= 50) </lsm:name> [1]
</lsm:validValues>
</...>
```

AllDataRequest

This structure is to encapsulate the request of get All Data operation.

Public Attributes

none

AllDataResponse

This structure is to encapsulate the result of get All Data operation.

Public Attributes

Location **locations** *Indicates a list of Location configuration data type.*
Event **events** *Indicates a list of Event (structure) configuration data type.*
EventType **eventTypes** *Indicates a list of EventType configuration data type.*
DeliveryTime **deliveryTimes** *Indicates a list of DeliveryTime (structure) configuration data type.*
Suite **suites** *Indicates a list of Suite configuration data type.*
Section **sections** *Indicates a list of Section (structure) configuration data type.*

MajorGroup **majorGroups**	*Indicates a list of MajorGroup configuration data type.*
FamilyGroup **familyGroups**	*Indicates a list of FamilyGroup (structure) configuration data type.*
MenuItemCategory **menuItemCategories**	*Indicates a list of MenuItemCategory configuration data type.*
MenuPriceLevel **menuPriceLevels**	*Indicates a list of MenuPriceLevel (structure) configuration data type.*
MenuItem **menuItems**	*Indicates a list of MenuItem structure (configuration data type).*
Owner **owners**	*Indicates a list of Owner structure (configuration data type).*
BarParLevel **barParLevels**	*Indicates a list of BarParLevel structure (configuration data type).*
Question **questions**	*Indicates a list of Question configuration data type.*

XML Instance Representation

```
<...>
<lsm:locations> [1]
<lsm:location> lsm:Location </lsm:location> [0..*]
</lsm:locations>
<lsm:events> [1]
<lsm:event> lsm:Event </lsm:event> [0..*]
</lsm:events>
<lsm:eventTypes> [1]
<lsm:eventType> lsm:EventType </lsm:eventType> [0..*]
</lsm:eventTypes>
<lsm:deliveryTimes> [1]
<lsm:deliveryTime> lsm:DeliveryTime </lsm:deliveryTime> [0..*]
</lsm:deliveryTimes>
<lsm:suites> [1]
<lsm:suite> lsm:Suite </lsm:suite> [0..*]
</lsm:suites>
<lsm:sections> [1]
<lsm:section> lsm:Section </lsm:section> [0..*]
</lsm:sections>
<lsm:majorGroups> [1]
<lsm:majorGroup> lsm:MajorGroup </lsm:majorGroup> [0..*]
</lsm:majorGroups>
<lsm:familyGroups> [1]
<lsm:familyGroup> lsm:FamilyGroup </lsm:familyGroup> [0..*]
</lsm:familyGroups>
```

```
<lsm:menuItemCategories> [1]
<lsm:menuItemCategory> lsm:MenuItemCategory </lsm:menuItemCategory> [0..*]
</lsm:menuItemCategories>
<lsm:menuPriceLevels> [1]
<lsm:menuPriceLevel> lsm:MenuPriceLevel </lsm:menuPriceLevel> [0..*]
</lsm:menuPriceLevels>
<lsm:menuItems> [1]
<lsm:menuItem> lsm:MenuItem </lsm:menuItem> [0..*]
</lsm:menuItems>
<lsm:owners> [1]
<lsm:owner> lsm:Owner </lsm:owner> [0..*]
</lsm:owners>
<lsm:barParLevels> [1]
<lsm:barParLevel> lsm:BarParLevel </lsm:barParLevel> [0..*]
</lsm:barParLevels>
<lsm:questions> [1]
<lsm:question> lsm:Question </lsm:question> [0..*]
</lsm:questions>
</...>
```

LocationDataRequest

This structure is to encapsulate the request of get just Location Data operation.

Public Attributes

long **locationId**
locationId *is a unique numeric value that identifies the location for which the request for data was made.*

LocationDataResponse

This structure is to encapsulate the result of get just Location Data operation.

Public Attributes

Location **location**
Indicates a single Location configuration data type, based on request parameter value.
Event **events**
Indicates a list of Event configuration data type, based on request parameter value.
EventType **eventTypes**
Indicates a list of EventType configuration data type, based on request parameter.
DeliveryTime **deliveryTimes**
Indicates a list of DeliveryTime configuration data type, based on request parameter.
Suite **suites**
Indicates a list of Suite configuration data type, based on request parameter.
Section **sections**
Indicates a list of Section configuration data type, based on request parameter.

MajorGroup **majorGroups** *Indicates a list of MajorGroup configuration data type, based on request parameter.*
FamilyGroup **familyGroups** *Indicates a list of FamilyGroup configuration data type, based on request parameter.*
MenuItemCategory **menuItemCategories** *Indicates a list of MenuItemCategory configuration data type, based on request parameter.*
MenuPriceLevel **menuPriceLevels** *Indicates a list of MenuPriceLevel configuration data type, based on request parameter.*
MenuItem **menuItems** *Indicates a list of MenuItem configuration data type, based on request parameter.*
Owner **owners** *Indicates a list of Owner configuration data type, based on request parameter.*
BarParLevel **barParLevels** *Indicates a list of BarParLevel configuration data type, based on request parameter.*
Question **questions** *Indicates a list of Question configuration data type, based on request parameter.*

XML Instance Representation

```
<...>
<lsm:location> lsm:Location </lsm:location>
<lsm:events> [1]
<lsm:event> lsm:Event </lsm:event> [0..*]
</lsm:events>
<lsm:eventTypes> [1]
<lsm:eventType> lsm:EventType </lsm:eventType> [0..*]
</lsm:eventTypes>
<lsm:deliveryTimes> [1]
<lsm:deliveryTime> lsm:DeliveryTime </lsm:deliveryTime> [0..*]
</lsm:deliveryTimes>
<lsm:suites> [1]
<lsm:suite> lsm:Suite </lsm:suite> [0..*]
</lsm:suites>
<lsm:sections> [1]
<lsm:section> lsm:Section </lsm:section> [0..*]
</lsm:sections>
<lsm:majorGroups> [1]
<lsm:majorGroup> lsm:MajorGroup </lsm:majorGroup> [0..*]
</lsm:majorGroups>
<lsm:familyGroups> [1]
<lsm:familyGroup> lsm:FamilyGroup </lsm:familyGroup> [0..*]
</lsm:familyGroups>
<lsm:menuItemCategories> [1]
<lsm:menuItemCategory> lsm:MenuItemCategory </lsm:menuItemCategory> [0..*]
```

</lsm:menuItemCategories>
<lsm:menuPriceLevels> [1]
<lsm:**menuPriceLevel**> lsm:MenuPriceLevel </lsm:**menuPriceLevel**> [0..*]
</lsm:menuPriceLevels>
<lsm:menuItems> [1]
<lsm:**menuItem**> lsm:MenuItem </lsm:**menuItem**> [0..*]
</lsm:menuItems>
<lsm:owners> [1]
<lsm:**owner**> lsm:Owner </lsm:**owner**> [0..*]
</lsm:owners>
<lsm:barParLevels> [1]
<lsm:**barParLevel**> lsm:BarParLevel </lsm:**barParLevel**> [0..*]
</lsm:barParLevels>
<lsm:questions> [1]
<lsm:**question**> lsm:Question </lsm:**question**> [0..*]
</lsm:questions>
</...>

AllMenuItemsRequest

This structure is to encapsulate the request of get All Menu Items operation.

Public Attributes

none

AllMenuItemsResponse

This structure is to encapsulate the result of get All Menu Items operation.

Public Attributes

MenuItem **menuItems**

List of all the **MenuItem** *structures that are present in the LSM database.*

XML Instance Representation

<...>
<lsm:majorGroups> [1]
<lsm:**majorGroup**> lsm:MajorGroup </lsm:**majorGroup**> [0..*]
</lsm:majorGroups>
<lsm:familyGroups> [1]
<lsm:**familyGroup**> lsm:FamilyGroup </lsm:**familyGroup**> [0..*]
</lsm:familyGroups>
<lsm:menuItemCategories> [1]
<lsm:**menuItemCategory**> lsm:MenuItemCategory </lsm:**menuItemCategory**> [0..*]
</lsm:menuItemCategories>

<lsm:menuPriceLevels> [1]
<lsm:**menuPriceLevel**> lsm:MenuPriceLevel </lsm:**menuPriceLevel**> [0..*]
</lsm:menuPriceLevels>
<lsm:menuItems> [1]
<lsm:**menuItem**> lsm:MenuItem </lsm:**menuItem**> [0..*]
</lsm:menuItems>
</...>

MenuItemsByLocationRequest

This structure is to encapsulate the request of get just a location Menu Items operation.

Public Attributes

long **locationId**

__locationId__ is a unique numeric value that identifies the location for which the request for menu items data was made.

MenuItemsByLocationResponse

This structure is to encapsulate the result of get just a location Menu Items operation.

Public Attributes

MenuItem **menuItems**

List of **MenuItem** *structures that are related to the location specified in the request parameter.*

XML Instance Representation

<...>
<lsm:majorGroups> [1]
<lsm:**majorGroup**> lsm:MajorGroup </lsm:**majorGroup**> [0..*]
</lsm:majorGroups>
<lsm:familyGroups> [1]
<lsm:**familyGroup**> lsm:FamilyGroup </lsm:**familyGroup**> [0..*]
</lsm:familyGroups>
<lsm:menuItemCategories> [1]
<lsm:**menuItemCategory**> lsm:MenuItemCategory </lsm:**menuItemCategory**> [0..*]
</lsm:menuItemCategories>
<lsm:menuPriceLevels> [1]
<lsm:**menuPriceLevel**> lsm:MenuPriceLevel </lsm:**menuPriceLevel**> [0..*]
</lsm:menuPriceLevels>
<lsm:menuItems> [1]
<lsm:**menuItem**> lsm:MenuItem </lsm:**menuItem**> [0..*]
</lsm:menuItems>
</...>

AllOwnersRequest

This structure is to encapsulate the request of get All Owners operation.

Public Attributes

None

AllOwnersResponse

This structure is to encapsulate the result of get All Owners operation.

Public Attributes

Owner **owners**

List of all the **Owner** *structures that are present in the LSM database.*

XML Instance Representation

```
<...>
<lsm:owners> [1]
<lsm:owner> lsm:Owner </lsm:owner> [0..*]
</lsm:owners>
</...>
```

OwnersByLocationRequest

This structure is to encapsulate the request of get just a location Owners operation.

Public Attributes

long **locationId**

locationId *is a unique numeric value that identifies the location for which the request for owners data was made.*

OwnersByLocationResponse

This structure is to encapsulate the result of get just a location Owners operation

Public Attributes

Owner **owners**

List of **Owner** *structures that are related to the location specified in the request parameter.*

XML Instance Representation

```
<...>
<lsm:owners> [1]
<lsm:owner> lsm:Owner </lsm:owner> [0..*]
</lsm:owners>
</...>
```

Order Processing Data Structures

CreditCard

This structure holds the two pieces of information for a credit card record.

Public Attributes

string	**number**
	*Indicates Owner's credit card **number** attached to the standing order.*
string	**expirationDate**
	Indicates expiration date (mmyy) for Owner's credit card number attached to the standing order.

XML Instance Representation

```
<...>
<lsm:number> xs:string (length <= 20) </lsm:number> [1]
<lsm:expirationDate> xs:string (length = 4) </lsm:expirationDate> [1]
</...>
```

OrderContactInfo

This structure is holding the contact information for a person related to a single order.

Public Attributes

string	**name**
	*Indicates the **name** of contact person related to the (event/standing) order.*
string	**phone**
	*Holds the **phone** number of contact person related to this order.*
string	**extension**
	*Holds the **extension** of the phone number.*
string	**fax**
	*Holds the **fax** number of contact person related to the (event/standing) order.*

XML Instance Representation

```
<...>
<lsm:name> xs:string (length <= 50) </lsm:name> [1]
<lsm:phone> xs:string (length <= 20) </lsm:phone> [1]
<lsm:extension> xs:string (length <= 20) </lsm:extension> [1]
<lsm:fax> xs:string (length <= 20) </lsm:fax> [1]
</...>
```

Answer

This structure is holding answer to question attached to a single order.

Public Attributes

long **questionId** *Indicates the id of the question for which the answer is provided.*
string **answerText** *Indicates the textual value of an answer to a question related to order.*
long **answerValue** *Indicates the numeric value of an answer to a question.*

XML Instance Representation

```
<...>
<lsm:questionId> xs:long </lsm:questionId> [1]
<lsm:answerText> xs:string (length <= 200) </lsm:answerText> [1]
<lsm:answerValue> xs:long </lsm:answerValue> [1]
</...>
```

EventOrderHeader

This structure is to hold the event order summary information.

Public Attributes

string **orderId** *Holds the order id of an event order.*
string **ownerId** *Holds the owner id of a suite owner for which the event order was created.*
string **suiteId** *Indicates the suite id of a suite where the order ultimately needs to be delivered.*
string **eventId** *Indicates the event id of an existing event for which the order was created.*
long **seatId** *Indicates the seat id in a suite/section/row where the order ultimately needs to be delivered*
decimal **subtotal** *Holds the subtotal amount of an event order.*
OrderContactInfo **contactInfo** *Indicates the contact information for a person (most likely suite owner) related to this order.*
string **paymentType** *Indicates the payment type that will be used by the owner to pay for the order.*
CreditCard **creditCard** *Holds the credit crad information of a suite owner related to this event order.*
string **comments**

Indicates the instructional textual value related to this event order.
DateTime **createdDate** *Holds the date and time when the event order got created.*
DateTime **modifiedDate** *Holds the date and time when the event order got modified last time.*

XML Instance Representation

```
<...>
<lsm:orderId> xs:string (length <= 20) </lsm:orderId> [1]
<lsm:ownerId> xs:string (length <= 20) </lsm:ownerId> [1]
<lsm:suiteId> xs:string (length <= 20) </lsm:suiteId> [1]
<lsm:eventId> xs:string (length <= 20) </lsm:eventId> [1]
<lsm:seatId> xs:long </lsm:seatId> [1]
<lsm:subtotal> xs:decimal </lsm:subtotal> [1]
<lsm:contactInfo> lsm:OrderContactInfo </lsm:contactInfo> [1]
<lsm:paymentType> xs:string (value comes from list: {'CREDIT'|'OTHER'})
</lsm:paymentType> [1]
<lsm:creditCard> lsm:CreditCard </lsm:creditCard> [1]
<lsm:comments> xs:string (length <= 1000) </lsm:comments> [1]
<lsm:createdDate> xs:dateTime </lsm:createdDate> [1]
<lsm:modifiedDate> xs:dateTime </lsm:modifiedDate> [1]
</...>
```

EventOrderDetail

This structure is to encapsulate the event order detail information.

Public Attributes

long **id** *Holds the header id that represents a link to the event order summary record.*
long **menuItemId** *Indicates the link to the menu item record used in the event order.*
int **quantity** *Indicates the quantitavive usage of a particular menu item used in the event order.*
long **priceLevel** *Indicates the price level of a particular menu item used in the event order.*
long **deliveryTimeId** *Indicates the link to the delivery time id record that applies to this event order.*
boolean **isModifier** *Indicates if the menu item represents a modifier or not.*

boolean **isDeleted**	*Indicates whether or not the event order detail is deleted.*
DateTime **createdDate**	*Holds the date and time when the event order detail record got created.*
DateTime **modifiedDate**	*Holds the date and time when the event order detail record got modified last time.*

XML Instance Representation

```
<...>
<lsm:id> xs:long </lsm:id> [1]
<lsm:menuItemId> xs:long </lsm:menuItemId> [1]
<lsm:quantity> xs:integer (1 <= value <= 999) </lsm:quantity> [1]
<lsm:priceLevel> xs:long </lsm:priceLevel> [1]
<lsm:deliveryTimeId> xs:long </lsm:deliveryTimeId> [1]
<lsm:isModifier> xs:boolean </lsm:isModifier> [1]
<lsm:isDeleted> xs:boolean </lsm:isDeleted> [1]
<lsm:createdDate> xs:dateTime </lsm:createdDate> [1]
<lsm:modifiedDate> xs:dateTime </lsm:modifiedDate> [1]
</...>
```

EventOrder

This structure represents a single event order.

Public Attributes

EventOrderHeader **header**	*Indicates a structure containing information related to an event order header.*
EventOrderDetail **item**	*Indicates a structure containing information related to one or multiple event order details.*
Answer **answer**	*Indicates a structure containing answers related to one or multiple event order questions.*

XML Instance Representation

```
<...>
<lsm:header> lsm:EventOrderHeader </lsm:header> [1]
<lsm:items> [1]
<lsm:item> lsm:EventOrderDetail </lsm:item> [0..*]
</lsm:items>
<lsm:answers> [1]
<lsm:answer> lsm:Answer </lsm:answer> [0..*]
</lsm:answers>
</...>
```

StandingOrderHeader

This structure is to hold the standing order summary information.

Public Attributes

string **orderId**	*Holds the order id of a standing order.*
string **ownerId**	*Holds the owner id of a suite owner for which the standing order was created.*
long **eventTypeId**	*Indicates the event type id od an existing event type for which the standing order was created.*
string **comments**	*Indicates the instructional textual value related to this standing order.*
DateTime **createdDate**	*Holds the date and time when the standing order got created.*
DateTime **modifiedDate**	*Holds the date and time when the standing order got modified last time.*

XML Instance Representation

```
<...>
<lsm:orderId> xs:string (length <= 20) </lsm:orderId> [1]
<lsm:ownerId> xs:string (length <= 20) </lsm:ownerId> [1]
<lsm:eventTypeId> xs:long </lsm:eventTypeId> [1]
<lsm:comments> xs:string (length <= 1000) </lsm:comments> [1]
<lsm:createdDate> xs:dateTime </lsm:createdDate> [1]
<lsm:modifiedDate> xs:dateTime </lsm:modifiedDate> [1]
</...>
```

StandingOrderDetail

This structure is to encapsulate the standing order detail information.

Public Attributes

long **id**	*Holds the header id that represents a link to the standing order summary record.*
long **menuItemId**	*Indicates the link to the menu item record used in the standing order.*
int **quantity**	*Indicates the quantitavive usage of a particular menu item used in the standing order.*

long **priceLevel** *Indicates the price level of a particular menu item used in the standing order.*
long **deliveryTimeId** *Indicates the link to the delivery time id record that applies to this standing order.*
boolean **isModifier** *Indicates if the menu item represents a modifier or not.*
boolean **isDeleted** *Indicates whether or not the standing order detail is deleted.*
DateTime **createdDate** *Holds the date and time when the standing order detail record got created.*
DateTime **modifiedDate** *Holds the date and time when the standing order detail record got modified last time.*

XML Instance Representation

```
<...>
<lsm:id> xs:long </lsm:id> [1]
<lsm:menuItemId> xs:long </lsm:menuItemId> [1]
<lsm:quantity> xs:integer (1 <= value <= 999) </lsm:quantity> [1]
<lsm:priceLevel> xs:long </lsm:priceLevel> [1]
<lsm:deliveryTimeId> xs:long </lsm:deliveryTimeId> [1]
<lsm:isModifier> xs:boolean </lsm:isModifier> [1]
<lsm:isDeleted> xs:boolean </lsm:isDeleted> [1]
<lsm:createdDate> xs:dateTime </lsm:createdDate> [1]
<lsm:modifiedDate> xs:dateTime </lsm:modifiedDate> [1]
</...>
```

StandingOrder

This structure represents a single standing order.

Public Attributes

StandingOrderHeader **header** *Indicates a structure containing information related to a standing order header.*
StandingOrderDetail **item** *Indicates a structure containing information related to one or multiple standing order details.*
Answer **answer** *Indicates a structure containing answers related to one or multiple standing order questions.*

XML Instance Representation

```
<...>
<lsm:header> lsm:StandingOrderHeader </lsm:header> [1]
<lsm:items> [1]
<lsm:item> lsm:StandingOrderDetail </lsm:item> [0..*]
</lsm:items>
<lsm:answers> [1]
<lsm:answer> lsm:Answer </lsm:answer> [0..*]
</lsm:answers>
</...>
```

General Result Structure

OperationResult

This structure represents operational result of a request call.

Public Attributes

string **requestId**	*Holds the uniqe request id value (e.i.* 7beaeb7c-a16e-43e1-b563-c74da6b9dd25*).*
decimal **requestProcessingTime**	*Holds a decimal value that represents the time (seconds) that took to process the request (e.i.* 7.09202957*).*
boolean **requestProcessingTimeSpecified**	*Holds a true or false value representing if the request processing time is required as a part of result or not.*
string **errorCode**	*Indicates a textual value that represents the error code.*
string **errorMessage**	*Indicates a textual value that describes the encountered error.*

XML Instance Representation

```
<...>
<lsm:requestId> lsm:string </lsm:requestId> [1]
<lsm:requestProcessingTime> lsm:decimal </lsm:requestProcessingTime>
<lsm:requestProcessingTimeSpecified> lsm:boolean
</lsm:requestProcessingTimeSpecified>
<lsm:Errors>
<lsm:errorCode> lsm:string </lsm:errorCode>
<lsm:errorMessage> lsm:string </lsm:errorMessage>
</lsm:Errors>
</...>
```

6 Sample Code

Suites Configuration Data API

Some of the data that is configured in the Suites Management application can be retrieved using the Suites Configuration Data API.

The client application can retrieve configuration data such as suites locations, events, event types, delivery times, suites, major groups, family groups, menu item categories, menu item price levels, menu items, owners, bar par levels and questions.

Client Authentication

The API client must authenticate in order to be able to consume the API services.

The following code snippet demonstrates how to authenticate using the *ClientAuthenticator* method.

1. Construct a new instance of the ConfigurationDataServiceImpl web service object.
2. Construct a ClientAutheticator request object.
3. Construct a ClientAuthenticatorResponse object to hold the results.
4. Invoke the ClientAuthenticator function of the ConfigurationDataServiceImpl webservice object.

```
ConfigDataService.ConfigurationDataServiceImpl mConfigApi = new
ConfigDataService.ConfigurationDataServiceImpl();

public void InvokeClientAuthenticatorMethod()
{
   ConfigDataService.ClientAuthenticator oClientAuthReq = new
ConfigDataService. ClientAuthenticator();
   oClientAuthReq.Request                    =                    new
ConfigDataService.ClientAuthenticatorRequest();
   oClientAuthReq.Request.ApiPass     =     "54d32072-72bd-4b36-b326-
bff310d9d2d2"; //API key value

   ConfigDataService.ClientAuthenticatorResponse   oClientAuthRsp   =
null;

   // Authenticate using the web service interface
   if ((mConfigApi != null) && (oClientAuthReq != null))
   {
      try
      {
          oClientAuthRsp                                              =
mConfigApi.ClientAuthenticator(oClientAuthReq);

          if (oClientAuthRsp != null)
          {
             if (oClientAuthRsp.OperationResult.Errors == null)
             {
                 // read the result
                 bool bAuth = oClientAuthRsp.IsAuthenticated;
```

```
                  string sLocations = oClientAuthRsp.CSVLocationID;
             }
              else
              {
                  //*client should do something with the error
                  Console.WriteLine(String.Format(
                          "Authentication
operation failed. Error Code: {0}, Error Message: {1}",

oClientAuthRsp.OperationResult.Errors[0].Code,

oClientAuthRsp.OperationResult.Errors[0].Message));
              }
           }
      }
      catch (Exception ex)
      {

          //*client should do something with the exception...
          MessageBox.Show(String.Format("Authentication     operation
failed. Exception Message:
          {0}", ex.Message));
      }
   }
}
```

API Response

The existence of ***OperationResult.Errors*** object of pResponse object indicates whether or not the operation has succeeded. In case the operation succeeded then ***OperationResult.Errors*** is null. In case of failure, ***OperationResult.Errors[0].Code*** property will hold the error code while detailed error message can be found in ***OperationResult.Errors[0].Message*** property.

Get All Data

All Suites Configuration Data can be retrieved in one method call.

The following code snippet demonstrates how to retrieve data using the ***GetAllData*** method.

1. Construct a new instance of the ConfigurationDataServiceImpl web service object.
2. Construct a GetAllDataRequest object and set the filtering parameters in this example no location filtering is specified by setting the parameter to false.
3. Construct a GetAllDataResponse object to hold the results.
4. Invoke the GetAllData function of the ConfigurationDataServiceImpl webservice object.

```
ConfigDataService.ConfigurationDataServiceImpl mConfigApi = new
ConfigDataService.ConfigurationDataServiceImpl();

public void InvokeGetAllDataMethod()
{
   ConfigDataService.GetAllData oGetALLReq = new ConfigDataService.GetAllData();
   oGetALLReq.Request = new ConfigDataService.GetAllDataRequest();
   // If locationIdSpecified is ‘true’ then the existing value for locationId
must be
   // provided in order to get only the data from specified location…
   oGetALLReq.Request.locationId = 0;
   oGetALLReq.Request.locationIdSpecified = false;
   ConfigDataService.GetAllDataResponse oGetALLRsp = null;

   // Get the Configuration data using the web service interface
   if ((mConfigApi != null) && (oGetALLReq != null))
   {
      try
      {
          oGetALLRsp = mConfigApi.GetAllData(oGetALLReq);

          if (oGetALLRsp != null)
          {
             if (oGetAllRsp.OperationResult.Errors == null)
             {
                 // Iterate the locations
                 for (int i = 0; i < oGetALLRsp.Locations.Length; i++)
                 {
                     oGetALLRsp.Locations[i].locationId;
                     oGetALLRsp.Locations[i].name;
                     // Use or Store data
                     .
                     .
                     .
                 }
             }
             else
             {
                 //*client should do something with the error
                 Console.WriteLine(String.Format(
                          "Get All Data
operation failed. Error Code: {0}, Error Message: {1}",
                          oGetALLRsp.OperationResult.Errors[0].Code,
                          oGetALLRsp.OperationResult.Errors[0].Message));
              }
           }
      }
      catch (Exception ex)
      {

            //*client should do something with the exception...
            MessageBox.Show(String.Format("Get All Data operation failed.
Exception Message:
            {0}", ex.Message));
      }
   }
}
```

5. Iterate the data returned in the GetAllDataResponse object. In the above code snippet, only the extraction of the configuration data for locations is shown.

API Response

The existence of ***OperationResult.Errors*** object of pResponse object indicates whether or not the operation has succeeded. In case the operation succeeded then ***OperationResult.Errors*** is null. In case of failure, ***OperationResult.Errors[0].Code*** property will hold the error code while detailed error message can be found in ***OperationResult.Errors[0].Message*** property.

Get Events

Event related data configured in the Suites Management application can be retrieved via GetEvents method call.

The client application can retrieve the Event data such as id, name, event type id, date and time, restock flag, user defined fields (1, 2 and 3) and location id to whom event belongs from Suites Management LSM database using GetEvents method. Depending on its request parameters value this method can return all, some or just a single record of specified Event configuration data type.

The following code snippet demonstrates how to retrieve data using the GetEvents method.

1. Construct a new instance of the ConfigurationDataServiceImpl web service object.
2. Construct a GetEventsRequest object and set the filtering parameters in this example no location filtering is specified by setting the parameter to false.
3. Construct a GetEventsResponse object to hold the results.
4. Invoke the GetEvents function of the ConfigurationDataServiceImpl webservice object.

```
ConfigDataService.ConfigurationDataServiceImpl mConfigApi = new
ConfigDataService.ConfigurationDataServiceImpl();

public void InvokeGetEventsMethod()
{
    ConfigDataService.GetEvents oEventsReq = new ConfigDataService.GetEvents();

    oEventsReq.Request = new ConfigDataService.GetEventsRequest();
   // If locationIdSpecified is ‘true’ then the existing value for locationId
must be
   // provided in order to get only the events from specified location…
   oGetEventsReq.Request.eventId = “”;
   oGetEventsReq.Request.locationId = 0;
   oGetEventsReq.Request.locationIdSpecified = false;

    ConfigDataService.GetEventsResponse oEventsRsp = null;

    // Get the Events data using the web service interface
    if ((mConfigApi != null) && (oEventsReq != null))
    {
        try
        {
            oEventsRSP = mConfigApi.GetEvents(oEventsReq);

            if (oEventsRSP != null)
            {
                if (oEventsRsp.OperationResult.Errors == null)
```

```
                 {
                     //Iterate the events
                     for (int i = 0; i < oGetALLRsp.Events.Length; i++)
                     {
                        oGetALLRsp.Events[i].eventId;
                        oGetALLRsp.Events[i].name;
                        // Use or Store data
                        .
                        .
                        .
                     }
                 }
                 else
                 {
                     // client should do something with the error
                     Console.WriteLine(String.Format(
                                "Get Events
operation failed. Error Code: {0}, Error Message: {1}",
                                oEventsRsp.OperationResult.Errors[0].Code,
                                oEventsRsp.OperationResult.Errors[0].Message));
                 }
             }
         }
         catch (Exception ex)
         {
             //*client should do something with the exception...
             MessageBox.Show(String.Format("Get Events operation failed.
Exception Message:
             {0}", ex.Message));
         }
     }
}
```

Following sections explain parameters that are not explained in Calculate Totals Transaction method in the previous section. Please refer to previous section for all other parameters as they are all already explained in there.

GetEvents request

The request object for GetEvents is simple but also flexible like of all the Suites API requests. Its intent is to request event data from LSM database. The request parameters that usually represent the filter for the required data - in this case are set to *no filtering* – however, they can have powerful filtering capabilities when providing specific values to them.

Event data is the core component of all Suites related transactions. Everything "ordered" or "rung in" for the Suites system must be attached to a Suites Event data.

The code snippet given below demonstrates how to construct input information for Get Events request parameter. This code adds three attributes that are basically disregarding filtering of the response data by event id and location id, which results in receiving the events data for all the locations from the LSM database.

API Response

The response of the method call can be found in ***pResponse*** parameter. The existence of ***OperationResult.Errors*** object of pResponse object indicates whether or not the operation has succeeded. In case the operation succeeded then OperationResult.Errors is null. In case of failure, ***OperationResult.Errors[0].Code*** property will hold the error code while detailed error message can be found in ***OperationResult.Errors[0].Message*** property.

Suites Order Processing API

Event Order Create

EventOrderCreate request

For order creation operation, the client application can use the data available and send the order header and order detail data such as owner id, event id, suites id, seat id, payment type, answer, menu item id, quantity using the EventOrderCreate request. Once the data request is sent, Order_Header_ID and Order_Detail_ID are created in the Suites LSM database.

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

1. Construct a new instance of the SuitesOrderingServiceImpl web service object.
2. Construct an EventOrderCreate request object.
3. Construct EventOrderHeader object and and populate the objects' properties values.
4. Construct EventOrderDetail object and and populate the objects' properties values.
5. Construct an EventOrderCreate response object to hold the results.
6. Invoke the EventOrderCreate function of the SuitesOrderingServiceImpl webservice object.

```
OrderProcessingService.SuitesOrderingServiceImpl m_Ordering = new
OrderProcessingService.SuitesOrderingServiceImpl();

public void InvokeEventOrderCreateMethod()
{
    OrderProcessingService.EventOrder oEventOrder = new
OrderProcessingService.EventOrder();

            OrderProcessingService.EventOrderCreate oOrdCreate = new
OrderProcessingService.EventOrderCreate();

            //create order header object
            OrderProcessingService.EventOrderHeader oOrdHeader = new
OrderProcessingService.EventOrderHeader();

            //populate the order header values:
            oOrdHeader.ownerId = “OW170007”;
            oOrdHeader.suiteId = “S010002”;
            oOrdHeader.eventId = “EV17020011”;
            oOrdHeader.seatId = 1;
            oOrdHeader.isGuestOrder = false;
            if (PaymentTypeComboBox.Text == "CREDIT")
            {
                oOrdHeader.paymentType = 
OrderProcessingService.EventOrderHeaderPaymentType.CREDIT;
                oOrdHeader.CreditCard = new OrderProcessingService.CreditCard();
                oOrdHeader.CreditCard.number = "4444333322221111";
                oOrdHeader.CreditCard.expirationDate = "1020";
            }
            else
            {
                oOrdHeader.paymentType = 
OrderProcessingService.EventOrderHeaderPaymentType.OTHER;
                oOrdHeader.CreditCard = new OrderProcessingService.CreditCard();
                oOrdHeader.CreditCard.number = "";
                oOrdHeader.CreditCard.expirationDate = "";
            }
            oOrdHeader.comments = "Martini should be shaken not stirred…";
            oOrdHeader.ContactInfo = new OrderProcessingService.ContactInfo();
            oOrdHeader.ContactInfo.name = "Varam Bolle";
            oOrdHeader.ContactInfo.phone = "123-345-5678";
            oOrdHeader.ContactInfo.extension = "999";
            oOrdHeader.ContactInfo.fax = "123-345-5679";
            oOrdHeader.createdDate = DateTime.Now;
            oOrdHeader.modifiedDate = DateTime.Now;

            //assign the order header object
            oEventOrder.EventOrderHeader = oOrdHeader;

            oEventOrder.Answers = new 
SuitesAPITest.OrderProcessingService.Answer[1];
            OrderProcessingService.Answer oOrdAnswer = new 
OrderProcessingService.Answer();
            oOrdAnswer.questionId = 1;
            oOrdAnswer.answerText = "Answer text...";
            oOrdAnswer.answerValue = 1;
            oEventOrder.Answers[0] = oOrdAnswer;

            //create the order create request object
            OrderProcessingService.EventOrderCreateRequest oOrdREQ = new 
OrderProcessingService.EventOrderCreateRequest();
            oOrdREQ.EventOrder = oEventOrder;
            oOrdREQ.EventOrder.EventOrderHeader = oOrdHeader;
            int iDetails = 3;  // number of details (e.i. in the list)

            if (iDetails > 0)
            {
                oOrdREQ.EventOrder.EventOrderDetails = new 
OrderProcessingService.EventOrderDetail[iDetails];
                for (int i = 0; i < iDetails; i++)
                {
                    //for each list detail create dynamically the order detail 
object
                    OrderProcessingService.EventOrderDetail oOrdDetail = new 
OrderProcessingService.EventOrderDetail();

                    oOrdDetail.createdDate = DateTime.Now;
                    oOrdDetail.id = 1;
                    oOrdDetail.isDeleted = false;
                    oOrdDetail.isModifier = false;
                    oOrdDetail.modifiedDate = DateTime.Now;
                    oOrdDetail.quantity = “2”;
                    oOrdDetail.menuItemId = 123000;
                    oOrdDetail.priceLevel = 2;

                    oOrdREQ.EventOrder.EventOrderDetails[i] = oOrdDetail;
                }
            }
            oOrdCreate.Request = oOrdREQ;
            //create the order create response object
            OrderProcessingService.EventOrderCreateResponse oOrdRSP = null;

            if ((mainForm.m_Ordering != null) && (oOrdREQ != null))
            {
                try
                {
                    oOrdRSP = mainForm.m_Ordering.EventOrderCreate(oOrdCreate);

                    if (oOrdRSP == null || oOrdRSP.OperationResult.Errors !=
null)
                    {
                        MessageBox.Show("Failed to create Order");
                    }
                    else
                    {
                        MessageBox.Show("Successfully Created Order");
                    }
                }
                catch (Exception ex)
                {
                    // client should do something with the error
                    MessageBox.Show(ex.Message);
                }
            }
}
```

API Response

If operation succeeds then OperationalResult.Errors field will be ‘Null’ and summary of Event Order and Event Order Details will be returned. In case of failure, OperationalResult.Error field will return a ‘non-Null’ value while OperationalResult.ErrorCode will hold error code and OperationalResult.ErrorMessage will hold reason for failure.

Get Events Orders

GetEventOrdersRequest

Event Orders can be retrieved from the LSM database via Order Processing Service API using the GetEventOrders method.

The client application can retrieve the Event Order data such as order id, owner id, event id, suite id, seat id, payment type, subtotal, date and time, contact information to whom event order belongs from Suites Management LSM database using GetEventOrders method. Depending on its request parameters value this method can return all, some or just a single record.

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

1. Construct a new instance of the SuitesOrderingServiceImpl web service object.
2. Construct a GetEventOrders request object.
3. Construct a GetEventOrdersResponse object to hold the results.
4. Invoke the GetEventOrders function of the SuitesOrderingServiceImpl webservice object.
5. Construct EventOrder object.
6. Construct EventOrderHeader object and and populate the objects' properties values with response values
7. Construct EventOrderDetail object and and populate the objects' properties values with response values.

```
OrderProcessingService.SuitesOrderingServiceImpl m_Ordering = new
OrderProcessingService.SuitesOrderingServiceImpl();

public void InvokeGetEventOrdersDataMethod()
{
   OrderProcessingService.GetEventOrders oGetEventOrdReq = new
OrderProcessingService.GetEventOrders();
   oGetEventOrdReq.Request = new OrderProcessingService.GetEventOrdersRequest();
   OrderProcessingService.GetEventOrdersResponse oGetEventOrdResp = null;

   // Get the Configuration data using the web service interface
   if ((m_Ordering != null) && (oGetEventOrdReq != null))
   {
      try
      {
          oGetEventOrdResp = m_Ordering.GetEventOrders(oGetEventOrdReq);

          if (oGetEventOrdResp.EventOrders != null)
          {
                mEventOrdCntnr = new EventOrders();
                //Iterate the event Orders
                for (int i = 0; i < oGetEventOrdResp.EventOrders.Length; i++)
                {
                    EventOrder evOrd = new EventOrder();
                    evOrd.Header = new EventOrderHeader();
                    evOrd.Header.OrderId =
oGetEventOrdResp.EventOrders[i].EventOrderHeader.orderId;
                    evOrd.Header.OwnerId =
oGetEventOrdResp.EventOrders[i].EventOrderHeader.ownerId;
                    // Use or Store data
                    .
                    .
                    .
                    evOrd.Detail = new
EventOrderDetail[oGetEventOrdResp.EventOrders[i].EventOrderDetails.Length];
                    //Iterate the event Order Details
                    for (int j = 0; j <
oGetEventOrdResp.EventOrders[i].EventOrderDetails.Length; j++)
                    {
                        EventOrderDetail odtl = new EventOrderDetail();
                        odtl.Id =
oGetEventOrdResp.EventOrders[i].EventOrderDetails[j].id;
                        odtl.IsDeleted =
oGetEventOrdResp.EventOrders[i].EventOrderDetails[j].isDeleted;
                        // Use or Store data
                        .
                        .
                        .
                    }
                    //Iterate the event Order Answers
                    for (int x = 0; x <
oGetEventOrdResp.EventOrders[i].Answers.Length; x++)
                    {
                        Answer answ = new Answer();
                        evOrd.Answer = new
Answer[oGetEventOrdResp.EventOrders[i].Answers.Length];
                        answ.QuestionId =
oGetEventOrdResp.EventOrders[i].Answers[x].questionId;
                        answ.AnswerText =
oGetEventOrdResp.EventOrders[i].Answers[x].answerText;
                        // Use or Store data
                        .
                        .
                        .
                    }
                    mEventOrdCntnr.eventOrders.Add(evOrd);
                }
            }
        }
        catch (Exception ex)
        {
              // client should do something with the error
              MessageBox.Show(ex.Message;);
        }
    }
}
```

API Response

The response of the method call can be found in ***pResponse*** parameter. The existence of ***OperationResult.Errors*** object of pResponse object indicates whether or not the operation has succeeded. In case the operation succeeded then ***OperationResult.Errors*** is null. In case of failure, ***OperationResult.Errors[0].Code*** property will hold the error code while detailed error message can be found in ***OperationResult.Errors[0].Message*** property.

Parameter Signature

```
GetEventOrdersResponse        pResponse
```

GetEventsResponse signature

```
public class GetEventOrdersResponse
{
    public OperationResult OperationalResult;
    public EventCollection Events;
}
```

Event Order data can be seen in the demo client in the following form using the ‘Get Event Orders’ button. You can also see the xml response in the response tab.

Event Order Update

EventOrderUpdate Request

For order update operation post transaction, the client application can use the event orders available from the GetEventOrders request and update the data such as owner id, event id, suites id, seat id, payment type, answer, menu item id, quantity using the EventOrderUpdate request. Once the data request is sent, Order_Header_ID and Order_Detail_ID records are updated with the new data in the Suites LSM database.

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

1. Construct a new instance of the SuitesOrderingServiceImpl web service object.
2. Construct an EventOrderUpdate request object.
3. Construct EventOrderHeader object and and populate the objects’ properties values.
4. Construct EventOrderDetail object and and populate the objects’ properties values.
5. Construct an EventOrderUpdate response object to hold the results.
6. Invoke the EventOrderUpdate function of the SuitesOrderingServiceImpl webservice object.

```
OrderProcessingService.SuitesOrderingServiceImpl m_Ordering = new
OrderProcessingService.SuitesOrderingServiceImpl();

public void InvokeEventOrderUpdateMethod()
{
    OrderProcessingService.EventOrder oEventOrder = new
OrderProcessingService.EventOrder();

            OrderProcessingService.EventOrderUpdate oOrdUpdate = new
OrderProcessingService.EventOrderUpdate();
            //create order header object
            OrderProcessingService.EventOrderHeader oOrdHeader = new
OrderProcessingService.EventOrderHeader();
            //populate the order header values:
            oOrdHeader.orderId = “OR100203”;
            oOrdHeader.ownerId = “OW170007”;
            oOrdHeader.suiteId = “S010002”;
            oOrdHeader.eventId = “EV17020011”;
            oOrdHeader.seatId  = 1;
            oOrdHeader.isGuestOrder = false;
            if (PaymentTypeComboBox.Text == "CREDIT")
            {
                oOrdHeader.paymentType =
OrderProcessingService.EventOrderHeaderPaymentType.CREDIT;
                oOrdHeader.CreditCard = new OrderProcessingService.CreditCard();
                oOrdHeader.CreditCard.number = “4444333322221111”;
                oOrdHeader.CreditCard.expirationDate = “1020”;
            }
            else
            {
                oOrdHeader.paymentType =
OrderProcessingService.EventOrderHeaderPaymentType.OTHER;
                oOrdHeader.CreditCard = new OrderProcessingService.CreditCard();
                oOrdHeader.CreditCard.number = "";
                oOrdHeader.CreditCard.expirationDate = "";
            }
            oOrdHeader.comments = “Martini should be shaken not stirred…”;
            oOrdHeader.ContactInfo = new OrderProcessingService.ContactInfo();
            oOrdHeader.ContactInfo.name = “Varam Bolle”;
            oOrdHeader.ContactInfo.phone = “123-345-5678”;
            oOrdHeader.ContactInfo.extension = “999”;
            oOrdHeader.ContactInfo.fax = “123-345-5679”;
            oOrdHeader.createdDate = DateTime.Now;
            oOrdHeader.modifiedDate = DateTime.Now;
            //assign the order header object
            oEventOrder.EventOrderHeader = oOrdHeader;

            oEventOrder.Answers = new OrderProcessingService.Answer[1];

            OrderProcessingService.Answer oOrdAnswer = new
OrderProcessingService.Answer();
            oOrdAnswer.questionId = 1;
            oOrdAnswer.answerText = "Answer text...";
            oOrdAnswer.answerValue = 1;
            oEventOrder.Answers[0] = oOrdAnswer;

            //create the order detail object
            OrderProcessingService.EventOrderUpdateRequest oOrdREQ = new
OrderProcessingService.EventOrderUpdateRequest();
            oOrdREQ.EventOrder = oEventOrder;
            oOrdREQ.EventOrder.EventOrderHeader = oOrdHeader;
            int iDetails = 3;  //number of details (e.i. in the list)

            if (iDetails > 0)
            {
                oOrdREQ.EventOrder.EventOrderDetails = new
OrderProcessingService.EventOrderDetail[iDetails];
                for (int i = 0; i < iDetails; i++)
                {
                    OrderProcessingService.EventOrderDetail oOrdDetail = new
OrderProcessingService.EventOrderDetail();

                    oOrdDetail.createdDate = DateTime.Now;
                    oOrdDetail.id = 1;
                    oOrdDetail.isDeleted = false;
                    oOrdDetail.isModifier = false;
                    oOrdDetail.modifiedDate = DateTime.Now;
                    oOrdDetail.quantity = “2”;
                    oOrdDetail.menuItemId = 100101;
                    oOrdDetail.priceLevel = 1;

                    oOrdREQ.EventOrder.EventOrderDetails[i] = oOrdDetail;
                }
            }
            oOrdUpdate.Request = oOrdREQ;

            OrderProcessingService.EventOrderUpdateResponse oOrdRSP = null;

            if ((mainForm.m_Ordering != null) && (oOrdREQ != null))
            {
                try
                {
                    oOrdRSP = mainForm.m_Ordering.EventOrderUpdate(oOrdUpdate);
                    if (oOrdRSP == null)
                    {
                        MessageBox.Show("Failed to update the Event Order.");
                    }
                }
                catch (Exception ex)
                {
                    // client should do something with the error
                    MessageBox.Show(ex.Message);
                }
            }
}
```

API Response

If operation succeeds then ***OperationalResult.Errors*** field will be ‘Null’ and summary of Event Order and Event Order Details will be returned. In case of failure, ***OperationalResult.Error*** field will return a ‘non-Null’ value while ***OperationalResult.ErrorCode*** will hold error code and ***OperationalResult.ErrorMessage*** will hold reason for failure.

7 Demo Client for Suites API

Overview

The Suites API Demo Client is a Windows application that is used to test the features of Suites API. This application sends request to Suites API and displays the response in the UI based upon the input parameters provided by the user.

Application Location

The demo client application can be downloaded from Support.Oracle.com.

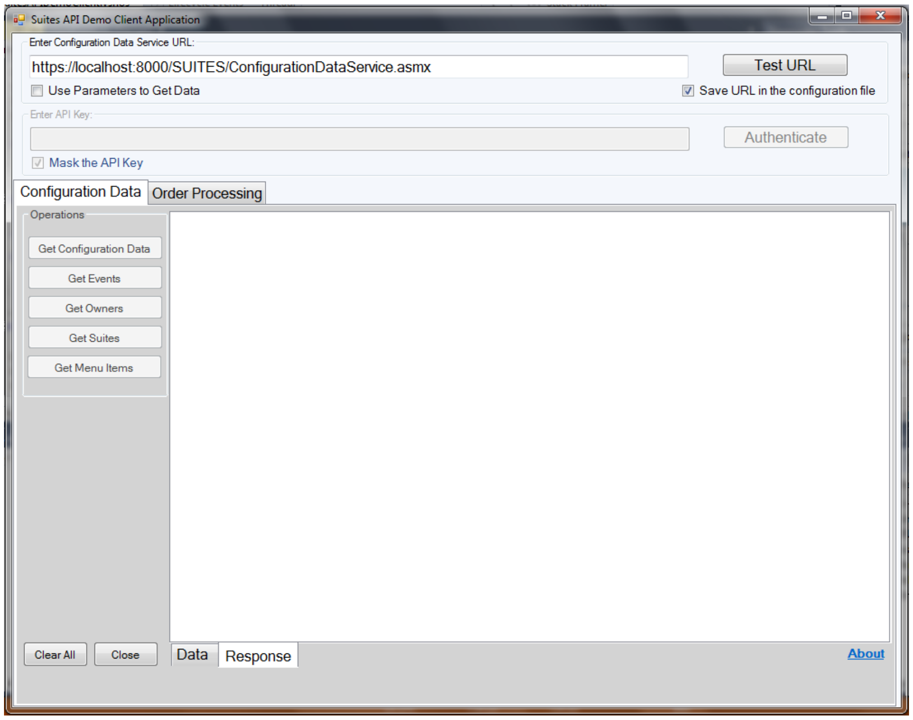
Prerequisites

A machine installed with the Suites Management application which is running IIS to host the Suites API web services. Navigate to the URLs of both Suites API web services using a web browser to validate they are accessible.

Initial Setup

Follow steps given below to configure and run demo client application.

1. Copy the *SuitesAPIDemoClient* to a local folder.
2. Launch the **SuitesAPIDemoClient.exe**.
3. Change the default URL value with the correct ConfigurationDataService URL



Enter Configuration Data Service URL: https://localhost:8000/SUITES/ConfigurationDataService.asmx

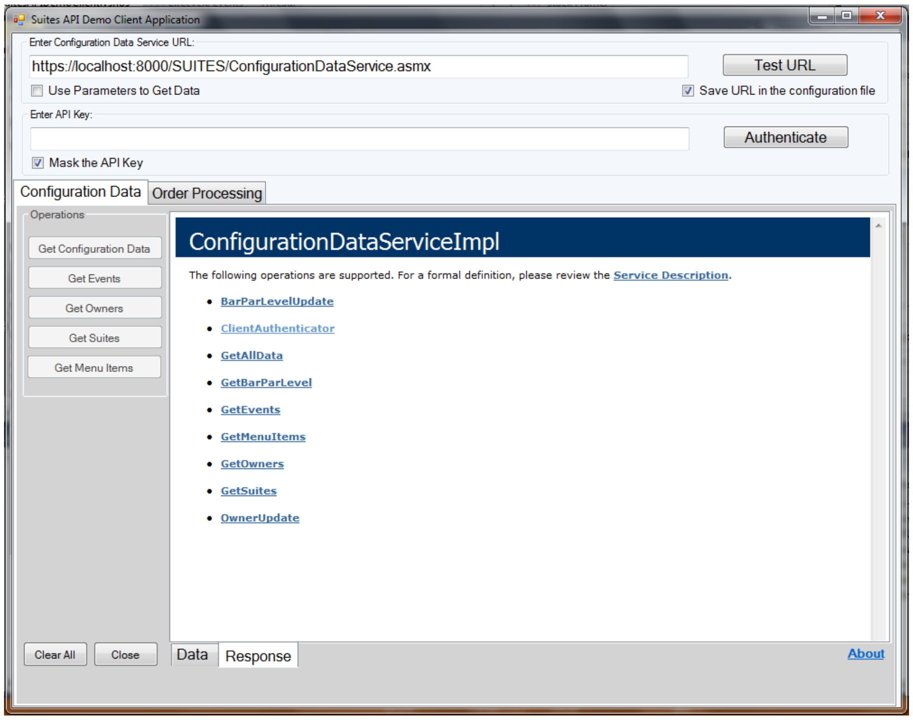
☐ Use Parameters to Get Data

☑ Save URL in the configuration file

Enter API Key:

☑ Mask the API Key

4. Click the **Test URL** button to validate communications with the Suites Configuration Data Service.
5. The successful test will display the Web Service operations in the Response tab and will enable the Authentication group below the URL group



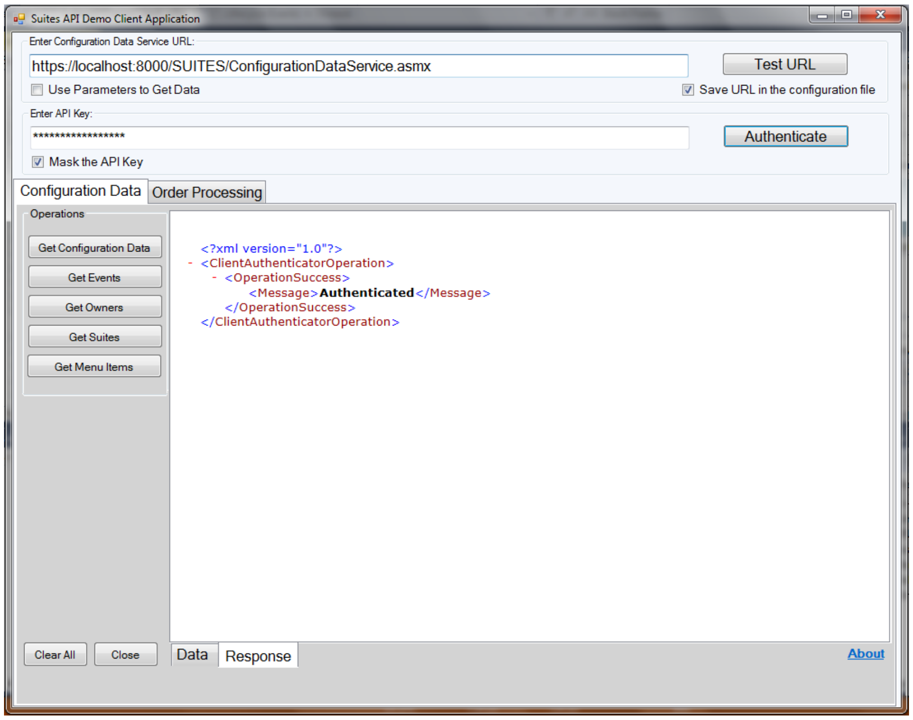
https://localhost:8000/SUITES/ConfigurationDataService.asmx

ConfigurationDataServiceImpl

The following operations are supported. For a formal definition, please review the Service Description.

- BarParLevelUpdate
- ClientAuthenticator
- GetAllData
- GetBarParLevel
- GetEvents
- GetMenuItems
- GetOwners
- GetSuites
- OwnerUpdate

6. Enter the API Key and click the **Authenticate** button.



https://localhost:8000/SUITES/ConfigurationDataService.asmx

****

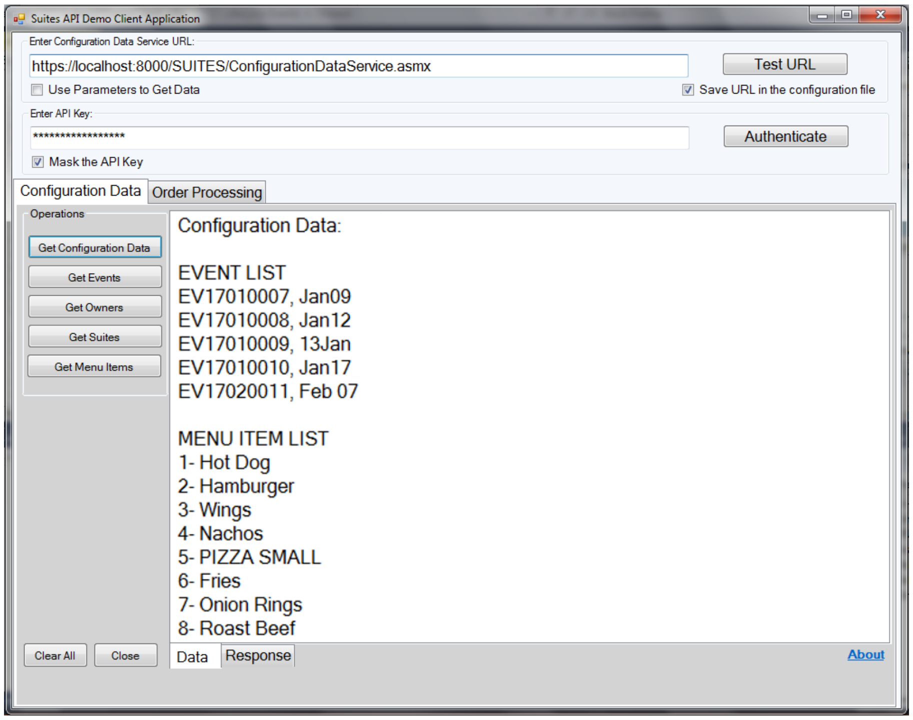
```
  <?xml version="1.0"?>
- <ClientAuthenticatorOperation>
  - <OperationSuccess>
      <Message>Authenticated</Message>
    </OperationSuccess>
  </ClientAuthenticatorOperation>
```

Configuration Data operations

GetAllData operation

The Demo client has a button called “Get Configuration Data” to invoke GetAllData method of Suites API Configuration Data web service. Follow the steps given below to get the data through UI and display in the data/response form. The GetAllData method gets all the data which is used by the OrderProcessing service to create or edit an order.

1. Once the URL is reachable and client authenticated, you can get the data using the **Get Configuration Data** button and see the data in the Data tab and xml response in Response tab.

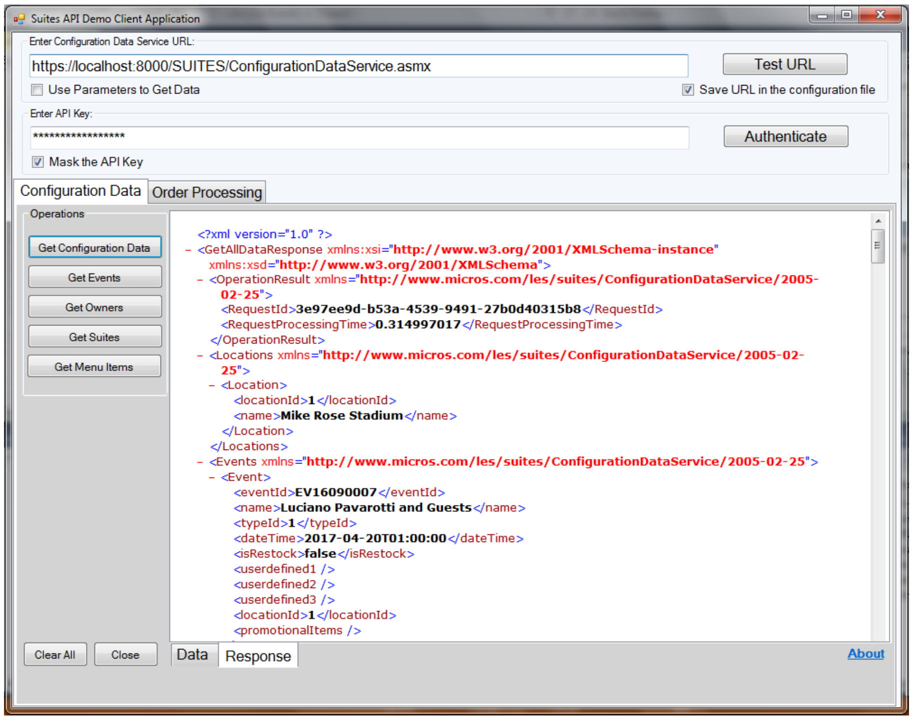


```
Configuration Data:

EVENT LIST
EV17010007, Jan09
EV17010008, Jan12
EV17010009, 13Jan
EV17010010, Jan17
EV17020011, Feb 07

MENU ITEM LIST
1- Hot Dog
2- Hamburger
3- Wings
4- Nachos
5- PIZZA SMALL
6- Fries
7- Onion Rings
8- Roast Beef
```

The user can also see the xml response from the request in the response tab.



Enter Configuration Data Service URL: https://localhost:8000/SUITES/ConfigurationDataService.asmx

Use Parameters to Get Data [ ]   Save URL in the configuration file [✓]

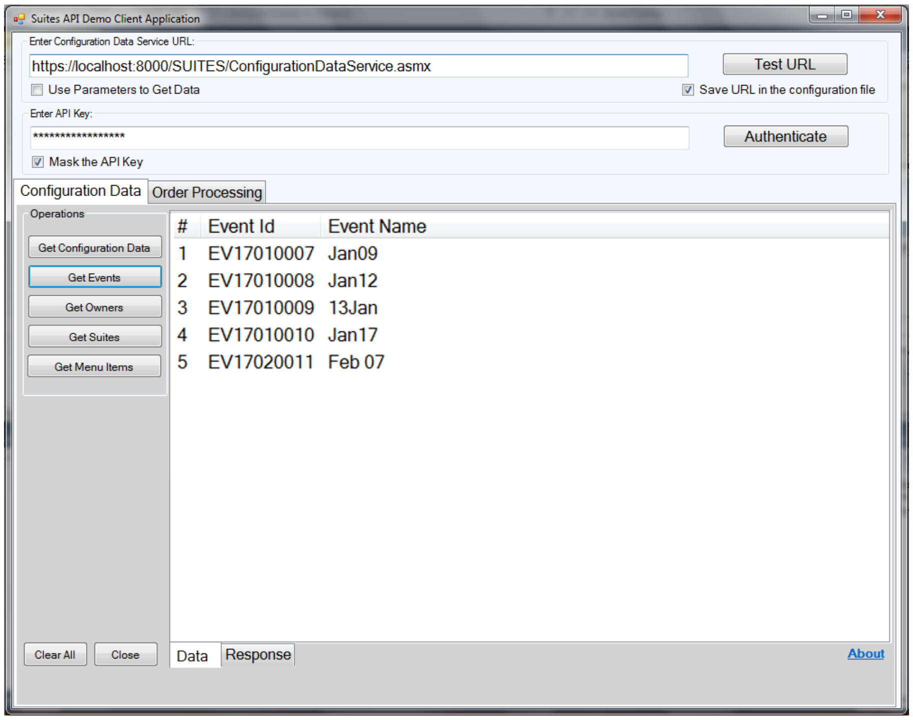
Enter API Key: ****************

Mask the API Key [✓]

```
  <?xml version="1.0" ?>
- <GetAllDataResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  - <OperationResult xmlns="http://www.micros.com/les/suites/ConfigurationDataService/2005-
      02-25">
      <RequestId>3e97ee9d-b53a-4539-9491-27b0d40315b8</RequestId>
      <RequestProcessingTime>0.314997017</RequestProcessingTime>
    </OperationResult>
  - <Locations xmlns="http://www.micros.com/les/suites/ConfigurationDataService/2005-02-
      25">
    - <Location>
        <locationId>1</locationId>
        <name>Mike Rose Stadium</name>
      </Location>
    </Locations>
  - <Events xmlns="http://www.micros.com/les/suites/ConfigurationDataService/2005-02-25">
    - <Event>
        <eventId>EV16090007</eventId>
        <name>Luciano Pavarotti and Guests</name>
        <typeId>1</typeId>
        <dateTime>2017-04-20T01:00:00</dateTime>
        <isRestock>false</isRestock>
        <userdefined1 />
        <userdefined2 />
        <userdefined3 />
        <locationId>1</locationId>
        <promotionalItems />
```

2. The other buttons, “Get Events”, “Get Owners”, “Get Suites” and “Get Menu Items” can be used to see the events, owners, suites, menu items data/response individually.

Events



Enter Configuration Data Service URL: https://localhost:8000/SUITES/ConfigurationDataService.asmx

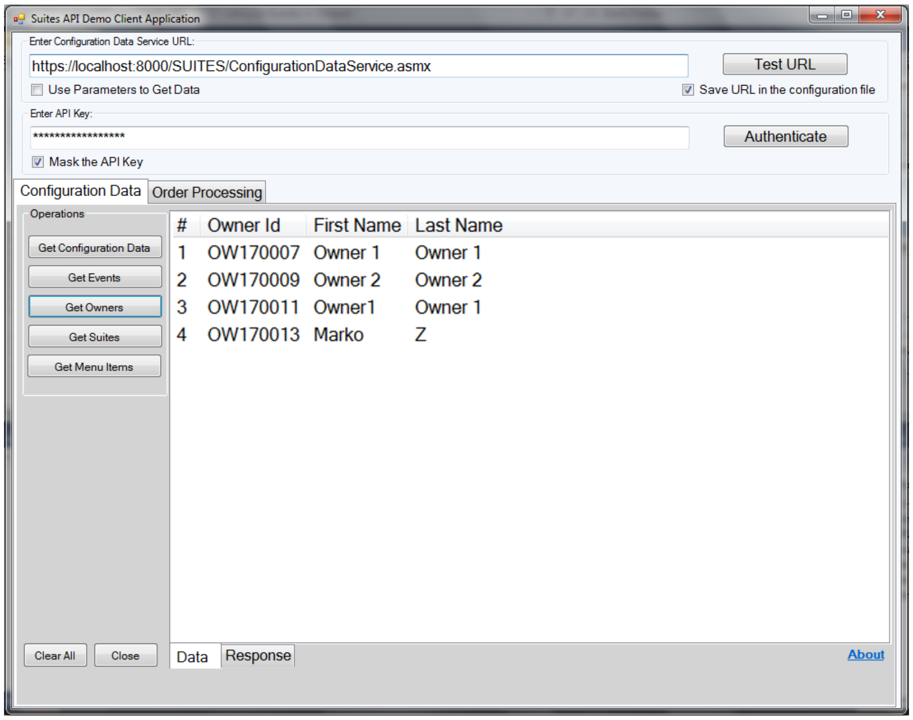
Use Parameters to Get Data [ ]   Save URL in the configuration file [✓]

Enter API Key: ****************

Mask the API Key [✓]

#	Event Id	Event Name
1	EV17010007	Jan09
2	EV17010008	Jan12
3	EV17010009	13Jan
4	EV17010010	Jan17
5	EV17020011	Feb 07

Owners



https://localhost:8000/SUITES/ConfigurationDataService.asmx

☐ Use Parameters to Get Data

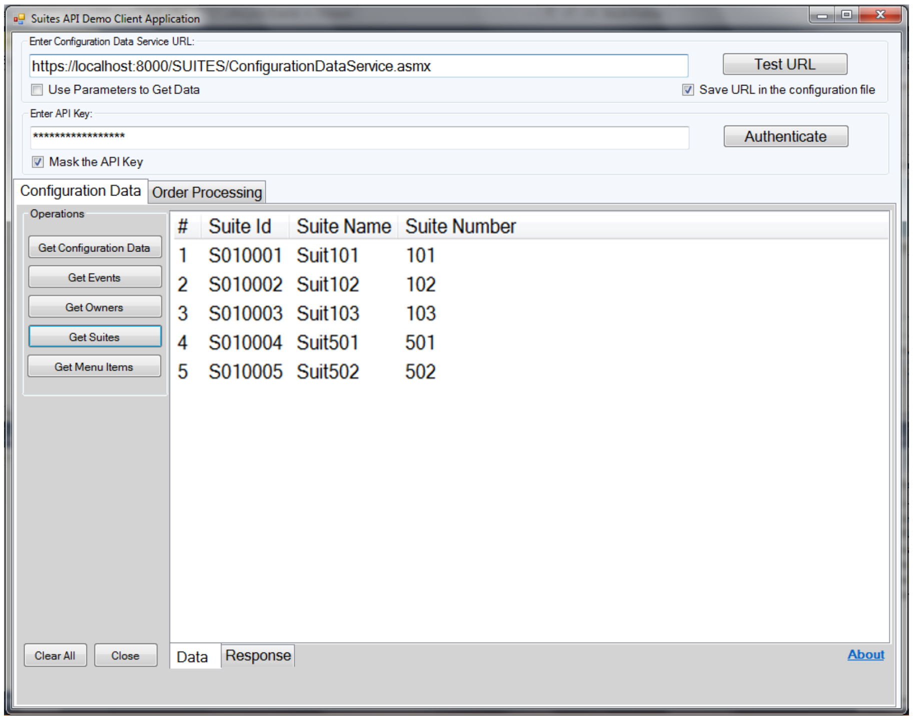
☑ Save URL in the configuration file

****

☑ Mask the API Key

#	Owner Id	First Name	Last Name
1	OW170007	Owner 1	Owner 1
2	OW170009	Owner 2	Owner 2
3	OW170011	Owner1	Owner 1
4	OW170013	Marko	Z

Suites



https://localhost:8000/SUITES/ConfigurationDataService.asmx

☐ Use Parameters to Get Data

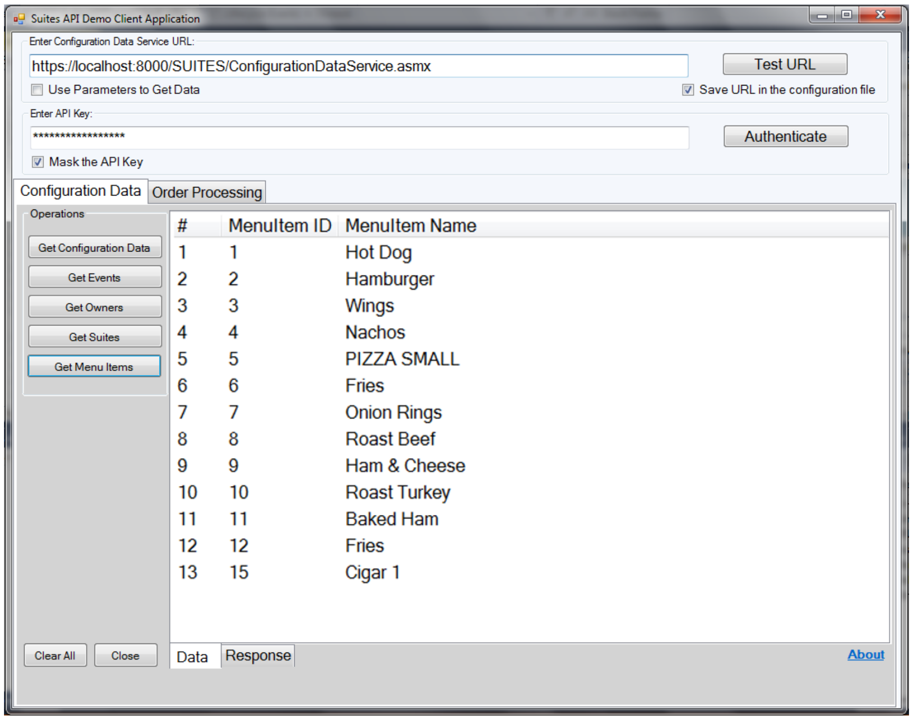
☑ Save URL in the configuration file

****

☑ Mask the API Key

#	Suite Id	Suite Name	Suite Number
1	S010001	Suit101	101
2	S010002	Suit102	102
3	S010003	Suit103	103
4	S010004	Suit501	501
5	S010005	Suit502	502

Menu Items



https://localhost:8000/SUITES/ConfigurationDataService.asmx

****

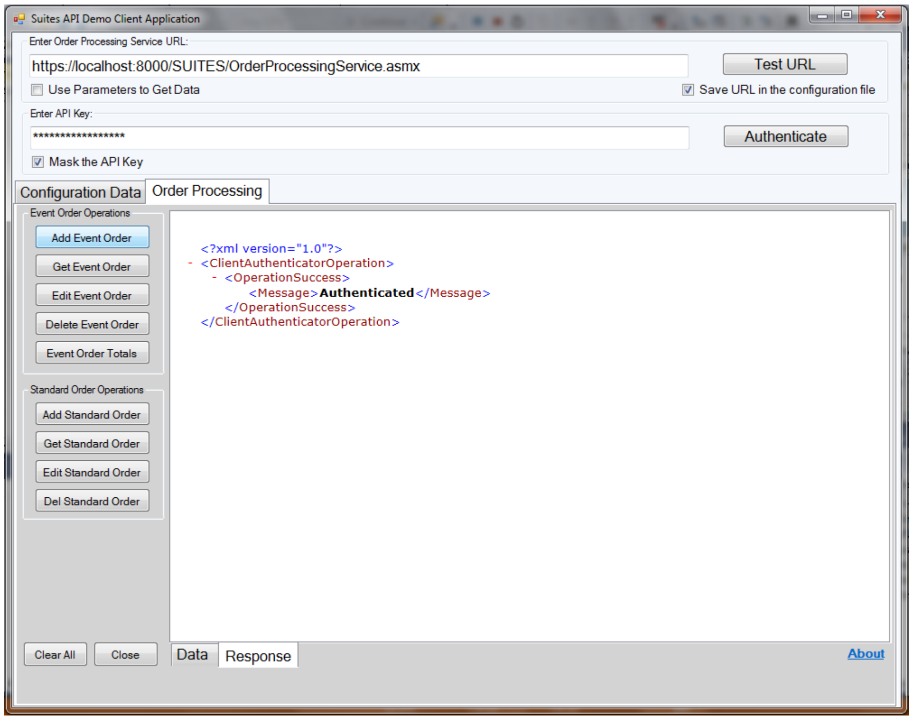
#	MenuItem ID	MenuItem Name
1	1	Hot Dog
2	2	Hamburger
3	3	Wings
4	4	Nachos
5	5	PIZZA SMALL
6	6	Fries
7	7	Onion Rings
8	8	Roast Beef
9	9	Ham & Cheese
10	10	Roast Turkey
11	11	Baked Ham
12	12	Fries
13	15	Cigar 1

Suites API Order Processing Operations

Create Event Order

Add Event Orders button can be used to send a request to the Suites API to create a new Event Order. The Suites data needs to be loaded into memory prior to creating the order by using the "Get Configuration Data" button on the Configuration Data tab.

1. Click on **Add Event Order** button from the **Event Order Processing** tab.



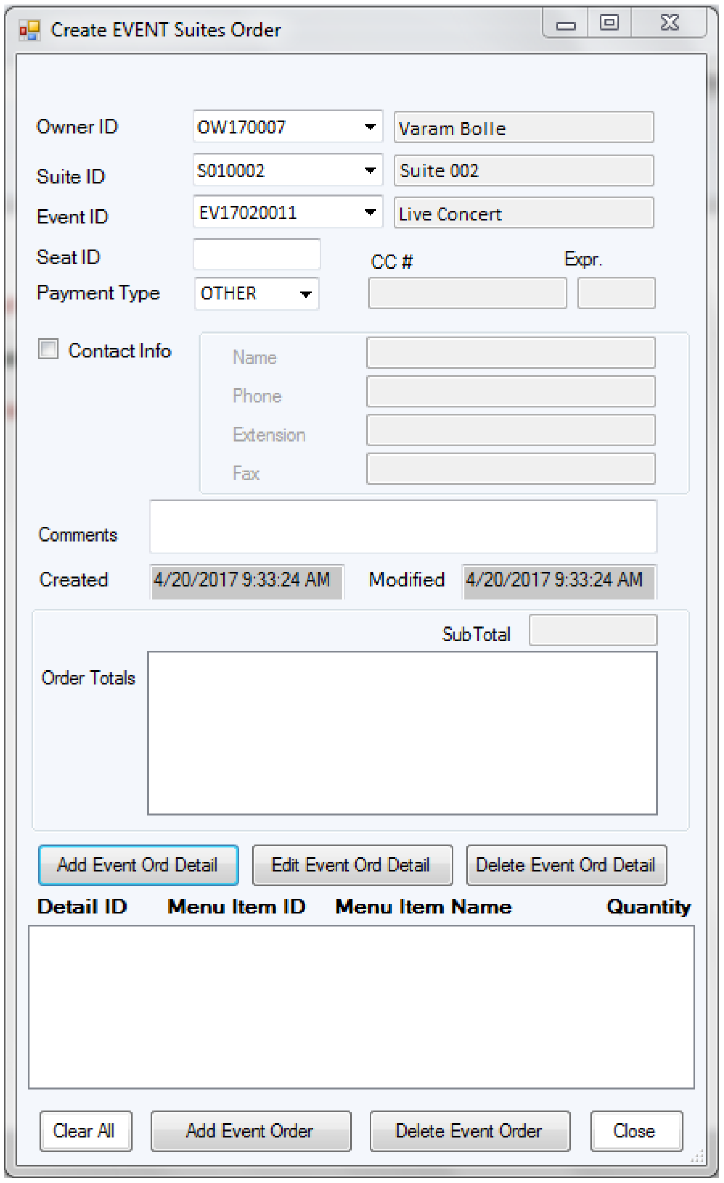
https://localhost:8000/SUITES/OrderProcessingService.asmx

****

```
<?xml version="1.0"?>
- <ClientAuthenticatorOperation>
  - <OperationSuccess>
      <Message>Authenticated</Message>
    </OperationSuccess>
  </ClientAuthenticatorOperation>
```

2. **Add OrderForm** will be displayed to enter the order information
3. Select the **OwnerID** from the Owner ID dropdown.
4. Select the **Suite ID** from the Suite ID dropdown box.
5. Select the **Event ID** from the Event ID dropdown box.

6. Click **Add Event Ord Detail** to add menu items.


Create EVENT Suites Order

Owner ID	OW170007	Varam Bolle
Suite ID	S010002	Suite 002
Event ID	EV17020011	Live Concert
Seat ID		
Payment Type	OTHER	

CC #

Expr.

Contact Info

Name

Phone

Extension

Fax

Comments

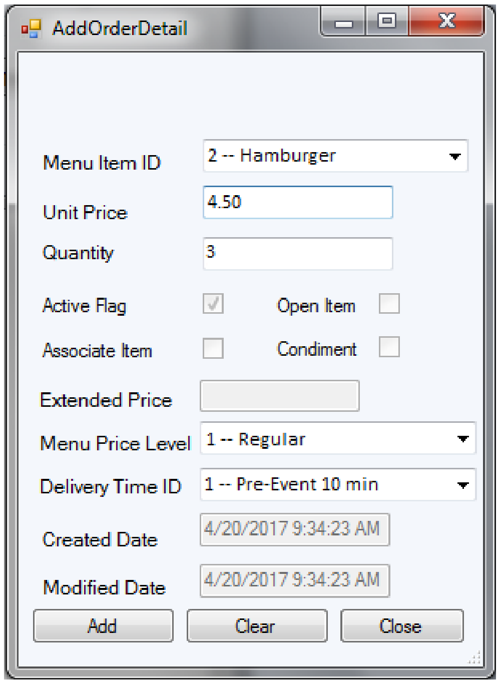
Created 4/20/2017 9:33:24 AM Modified 4/20/2017 9:33:24 AM

SubTotal

Order Totals

Detail ID	Menu Item ID	Menu Item Name	Quantity

7. Select the menu item and enter the quantity.



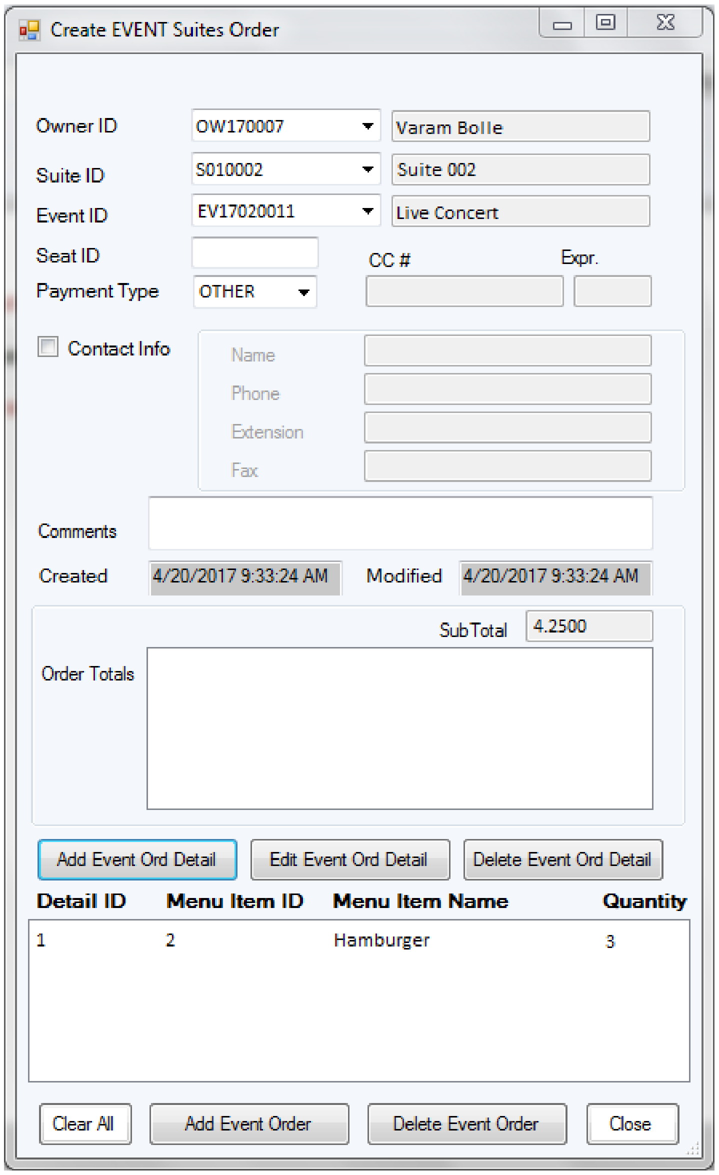
AddOrderDetail

Menu Item ID	2 -- Hamburger		
Unit Price	4.50		
Quantity	3		
Active Flag	✓	Open Item	
Associate Item		Condiment	
Extended Price			
Menu Price Level	1 -- Regular		
Delivery Time ID	1 -- Pre-Event 10 min		
Created Date	4/20/2017 9:34:23 AM		
Modified Date	4/20/2017 9:34:23 AM		

8. Click the **Add** button on the Add Order Detail form.

9. The selected menu item is populated in the List Box on the Add Order Form.

10. Once all the menu items are added, click the **Add Event Order** button to send the request to add the new Event Order.



Owner ID	OW170007	Varam Bolle
Suite ID	S010002	Suite 002
Event ID	EV17020011	Live Concert
Seat ID		
Payment Type	OTHER	

CC # Expr.

Contact Info: Name, Phone, Extension, Fax

Comments

Created 4/20/2017 9:33:24 AM Modified 4/20/2017 9:33:24 AM

SubTotal 4.2500

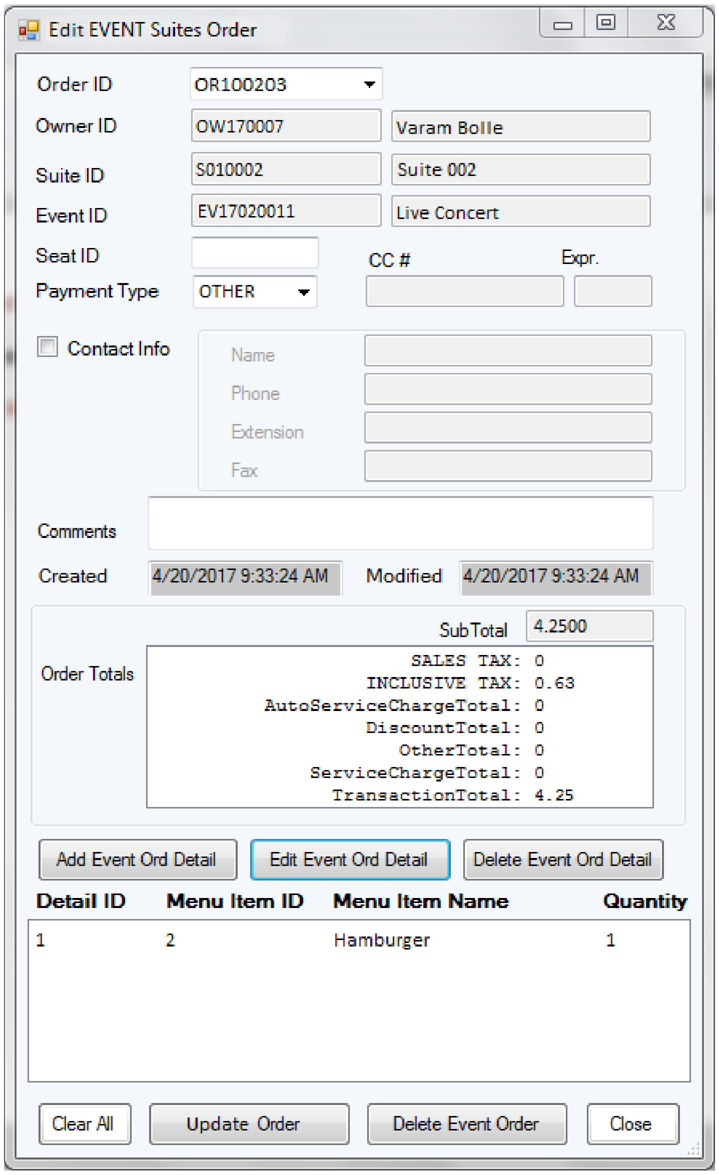
Order Totals

Detail ID	Menu Item ID	Menu Item Name	Quantity
1	2	Hamburger	3

Update Event Order

Get Event Order button can be used to send a request to Suites API to retrieve full detail of all the Event Orders from the LSM database. This button calls GetEventOrders method of Suites API. Using these event orders, user can edit an event order.

1. Click the **Edit Event Order** button from the Order Processing tab to open the following Add Order Form to edit an event order detail.



Order ID	OR100203		
Owner ID	OW170007	Varam Bolle	
Suite ID	S010002	Suite 002	
Event ID	EV17020011	Live Concert	
Seat ID		CC #	Expr.
Payment Type	OTHER		
Comments			
Created	4/20/2017 9:33:24 AM	Modified	4/20/2017 9:33:24 AM
SubTotal	4.2500		

```
           SALES TAX: 0
       INCLUSIVE TAX: 0.63
AutoServiceChargeTotal: 0
       DiscountTotal: 0
          OtherTotal: 0
  ServiceChargeTotal: 0
    TransactionTotal: 4.25
```

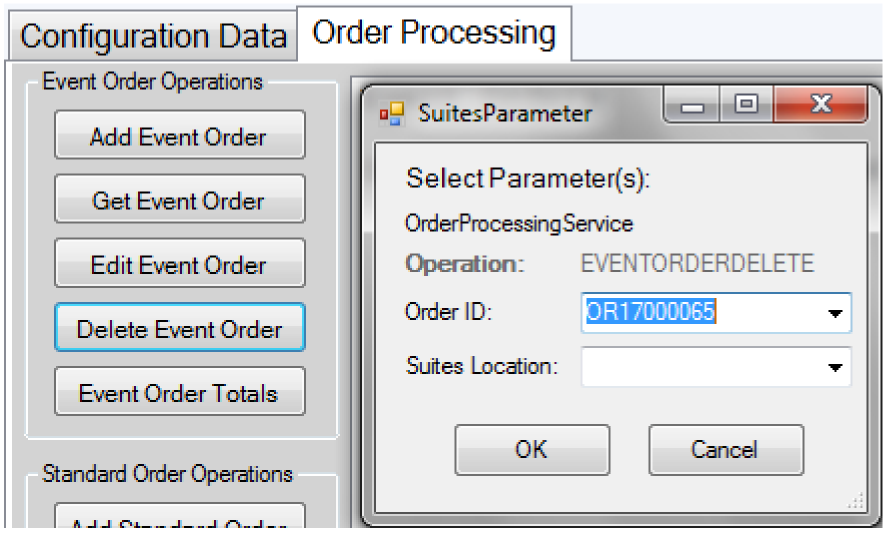
Detail ID	Menu Item ID	Menu Item Name	Quantity
1	2	Hamburger	1

2. Click on **Edit Detail** button to edit the details (Item Price Level, Quantity) in the Add Detail form
3. Click on **Update Order** button to send the update request to the Suites API
4. The user can go to the Suites web site and validate the Order for the Event has been updated.

Delete Event Order

Delete Event Order button is used to delete an event order using the Suites API EventOrderDelete method from the Suites LSM database.

1. Click the **Delete Event Order** button on the Order Processing tab
2. Select **Order ID** from the Order ID dropdown box of SuitesParameters dialog.
3. Click **OK** button to delete the Event Order selected in previous step.



Select Parameter(s):
OrderProcessingService
Operation: EVENTORDERDELETE
Order ID: OR17000065
Suites Location:

8 Error Logging

Suites API writes all errors and other informational messages to a flat file under following folder for diagnostics purpose.

<ROOT_INSTALL_DRIVE>\MICROS\LES\Suites\Logs\

Example path: **C:\MICROS\LES\Suites\Logs\ConfigurationDataService.log**

Example of log content:

```
2017-01-31 14:35:02,689 INFO ConfigurationDataService - SOAP REQUEST: ID:
d7ed73c0-6832-447c-9d1e-949258826132

2017-01-31 14:35:02,690 INFO ConfigurationDataService -            URL:
https://machine-name:444/SUITES/ConfigurationDataService.asmx

2017-01-31 14:35:02,691 INFO ConfigurationDataService -         ACTION:
http://www.micros.com/les/suites/ConfigurationDataService/2005-02-25/GetEvents

2017-01-31 14:35:02,692 INFO ConfigurationDataService -        ONE WAY: False

2017-01-31 14:35:02,692 INFO ConfigurationDataService -        MESSAGE: <?xml
version="1.0" encoding="utf-8"?><soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"><soap:Body><GetEvents
xmlns="http://www.micros.com/les/suites/ConfigurationDataService/2005-02-
25"><Request /></GetEvents></soap:Body></soap:Envelope>

2017-01-31 14:35:02,813 INFO ConfigurationDataService - SOAP RESPONSE: ID:
d7ed73c0-6832-447c-9d1e-949258826132

2017-01-31 14:35:02,815 INFO ConfigurationDataService -             URL:
https://machine-name:444/SUITES/ConfigurationDataService.asmx

2017-01-31 14:35:02,817 INFO ConfigurationDataService -          ACTION:
http://www.micros.com/les/suites/ConfigurationDataService/2005-02-25/GetEvents

2017-01-31 14:35:02,818 INFO ConfigurationDataService -       EXCEPTION: False

2017-01-31 14:35:02,820 INFO ConfigurationDataService -         MESSAGE: <?xml
version="1.0" encoding="utf-8"?><soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"><soap:Body><GetEventsResponse
xmlns="http://www.micros.com/les/suites/ConfigurationDataService/2005-02-
25"><OperationResult><RequestId>d7ed73c0-6832-447c-9d1e-
949258826132</RequestId><RequestProcessingTime>0.00405650353</RequestProcessingTim
e></OperationResult><Events><Event><eventId>EV16090007</eventId><name>Luciano
Pavarotti and Guests</name><typeId>1</typeId><dateTime>2016-09-
27T00:00:00</dateTime><isRestock>false</isRestock><userdefined1 /><userdefined2
/><userdefined3 /><locationId>1</locationId><promotionalItems
/></Event><Event><eventId>EV17010008</eventId><name>Andrea Bocelli Concert
2017</name><typeId>1</typeId><dateTime>2017-01-
20T00:00:00</dateTime><isRestock>false</isRestock><userdefined1 /><userdefined2
/><userdefined3 /><locationId>1</locationId><promotionalItems
/></Event></Events></GetEventsResponse></soap:Body></soap:Envelope>
```

9 Secure Development Guide

The Suites API secure development guidelines are provided to help developers build secure solutions that integrate with Oracle Hospitality Suites Management.

1. Communications with Suites API should be done over a secure connection using TLS 1.2 or higher.
2. Suites API requires an API authentication key and authorization token to execute Suites methods.
3. Applications consuming the Suites API should protect any sensitive customer data stored at rest.
    - A minimum AES256 encryption should be used
    - Encryption keys should not be stored with encrypted data
4. Applications should not write sensitive data (customer information, credit card numbers, email addresses etc.) into application log files.
5. Do not embed hard coded usernames, passwords, encryption keys, etc. into the application.
6. Where a login is required, applications should implement a strong password policy.
    - Minimum password length
    - Combination of letters and numbers
    - No dictionary words
7. When using a SQL database, limit the user privildges to the minimum required by the application.
8. Applications integrating with the Suites API should be built with the latest components to ensure that the latest security patches are included.
9. Applications consuming the Suites API should follow the OWASP standards `https://www.owasp.org`
10. Third Party applications should be developed and tested using security testing tools to prevent vulnerabilities in the application.