Component Intercommunication Specification

Tommy Lamb

February 3, 2018

Contents

1 Server API

1.1 General Comments

The server scripts follow standard HTTP status codes. In the case of a bad request (eg missing arguments) it will return status code 400. Where the server encounters an error, 500 is returned. These include errors in retrieving data from the database, which *may* be due to invalid argument values being supplied. Certain errors (of either type) will return an error message in the response text. More error reporting will be added in future. This should be used for debugging purposes only, and not relied upon for the final product.

As yet, the server does not check how many results are returned from the database. This means it may return invalid, empty JSON objects where the supplied arguments don't correspond to a database entry. Where more than one result is returned by the database, no guarantees are made as to which result is ultimately returned.

In accessing the server scripts, it is necessary that the requesting code also be running on the MACS departmental server. In practice this means the various components can only be tested on the server, rather than on local machines. Copying all files to another PHP-compatible server instance may negate this requirement. Failing to do so will generate a JavaScript error on the browser console which mentions the Access-Control-Allow-Origin response header. It may be possible to remove this requirement; investigation ongoing.

When updating an Order, it is not necessary to list all Items in the order, only those which have been modified. Indeed the JSON specification makes no allowance for unmodified items to be listed alongside modified ones.

1.2 Example Request

```
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200){
                console.log(this.responseText);
        } else if (this.readyState==4 &&
         ↪  (this.status==500||this.status==400)){
                //Error Handling here
        }
};
        xhttp.open("POST", "GetCustomerOrder.php" , true);
```

```
xhttp.setRequestHeader("Content-type",
↪  "application/x-www-form-urlencoded");
xhttp.send("OrderName=Pond&OrderNumber=12");
```

For "application/json;charset=UTF-8" encoded data, use the following:

```
xhttp.setRequestHeader("Content-type",
↪  "application/json;charset=UTF-8");
xhttp.send(JSON.stringify(Object));
```

Where Object is the JavaScript object representing the order.

1.3 Customer Component

1.3.1 GetCustomerOrder.php

Request Type:	POST
Encoding Type:	application/x-www-form-urlencoded
Argument List:	OrderName
	OrderNumber
Returns:	JSON: Order object
URL:	

`www2.macs.hw.ac.uk/~til1/ThirdYear/GroupProject/Customer/GetCustomerOrder.php`

1.4 Waiting Component

1.4.1 GetMenu.php

Request Type:	POST
Encoding Type:	application/x-www-form-urlencoded
Argument List:	MenuID
Returns:	JSON: Menu object
URL:	

`www2.macs.hw.ac.uk/~til1/ThirdYear/GroupProject/Waiting/GetMenu.php`

1.4.2 AddCustomerOrder.php

Request Type:	POST
Encoding Type:	application/json;charset=UTF-8
Argument List:	JSON Order string
Returns:	nothing
URL:	

`www2.macs.hw.ac.uk/~til1/ThirdYear/GroupProject/Waiting/AddCustomerOrder.php`

1.4.3 UpdateCustomerOrder.php

Request Type:	POST
Encoding Type:	application/json;charset=UTF-8
Argument List:	JSON Order string with *Modified* flags included
Returns:	nothing
URL:	

`www2.macs.hw.ac.uk/~til1/ThirdYear/GroupProject/Waiting/UpdateCustomerOrder.php`

2 JSON Specification

2.1 Menu

- The Menu object primarily consists of a tree of two other objects: Menu Item and Menu Section
- MenuID is the globally unique identifier for this menu object
- DisplayName is the string to be used if representing the object to a user
- Sections is a list of the top-level MenuSection objects that comprises this menu object. Examples may include MenuSection objects representing Starters, Mains, Deserts, or Drinks.

```
Menu : {
        "MenuID" : int,
        "DisplayName" : string,
        "Sections" : [MenuSection, MenuSection, ...]
}
```

2.1.1 Menu Item

- Represents any single 'thing' that can be ordered by a customer.
- The ItemID is a globablly unique identifier for the specific item.
- ParentSectionID is the globally unique identifier for the menu section under which this item is listed.
- DisplayName is the string to be used if representing the object to a user

- Description is a textual description of the object, eg *Decadent chocolate Bombe glacée drizzled with toffee sauce and served with black forest fruit compote.*
- Price is the decimal value representing the price of the item to the customer ordering.

```
MenuItem : {
        "ItemID" : int,
        "ParentSectionID" : int,
        "DisplayName" : string,
        "Description" : string,
        "Price" : decimal
}
```

2.1.2 Menu Section

- Represents a section of the menu, which may contain either a number of subsections, or a number of menu items.
- ParentSectionID is the globally unique identifier for the menu section under which this section is listed.
- ParentSectionID may be null if the section is a top-level element. That is, child only to the Menu object.
- GroupColour is a string containing a colour hex code, eg "#FFFFFF"
- HasItems represents whether the section has items (true) or subsections (false).
- Behaviour where both Items and Subsections are specified is as yet undefined.

```
MenuSection : {
        "SectionID" : int,
        "ParentSectionID" : int,
        "DisplayName" : string,
        "GroupColour" : string,
        "HasItems" : Boolean,
        ["Items" : [MenuItem, MenuItem, ...]],
        ["Subsections" : [MenuSection, MenuSection, ...]]
}
```

2.2 Order

- Represents an order as placed by a customer (by any means).
- OrderNumber is a number that can be used to geographically locate the customer and/or to group orders entered separately. For example, a table number. For grouping separate orders, this allows orders to be placed separately from a group of people, but have those orders conceptually grouped in the system to allow better customer service (eg have all orders prepared at the same time).
- Items is the list of OrderItem objects that this order consists of.
- ETA is either the string HH:MM representing the estimated time of completion for the order, or null if no such estimate exists.

```
Order : {
        "OrderNumber" : int,
        "OrderName" : string,
        "ETA" :  string | null,
        "Items" : [OrderItem, OrderItem, ...],
}
```

2.2.1 Order Item

- Represents a single type of MenuItem ordered by a customer.
- Amount is the non-zero integer representing how many of the MenuItem the customer has ordered.
- Request is a potentially empty string which represents any specific requests or choices made by the customer (as under F-UR-1.4 and related Non-Functional requirements)
- MenuItem is the JSON object specified above representing the thing being ordered.
- ETA is either the string HH:MM representing the estimated time of completion for this item, or null if no such estimate exists.
- GroupNumber is an integer used to group items within an order. This allows the system to handle starters, mains, and deserts being specified within one order but handled individually (IE served separately).
- The Modified flag must be specified for all items if the Order object is being used in the context of changing the details of an existing order. The value of the flag is determined as follows:

  0 : Item has been added to order

  1 : Item has been removed from the order

  2 : Item ETA has been changed (including to/from null)

  3 : Item details have changed (one or both of: Amount, GroupNumber)

  When changing an order, the *MenuItem* field must remain unchanged within the *OrderItem* object. To change an item in the order, the original *OrderItem* object should be included with the *Modified* flag set to 1 (removed), and the new OrderItem object listed with *Modified* flag set to 0 (added).

  A similar procedure must be followed for modifying the *request* field. A new *OrderItem* object with the new request should be used with the *Modified* flag set to 0 (added). The original *OrderItem* should be included with either the *Modified* flag set to 1 (removed), or set to 3 (details changed) with the *Amount* field reduced by the appropriate amount. The latter case is when two or more of the same menu items have been ordered, but only some of them have their request amended:

for example, if an order with three identical beverages with no request is updated to have 2 with no request and 1 with the request ”no ice”.

```
OrderItem : {
        "Amount" : int,
        "Request" : string,
        "MenuItem" : MenuItem,
        "ETA" : string | null,
        "GroupNumber" : int,
        ["Modified" : int]
}
```