EPSON

111-56-URM-010

TrueOrder KDS

API Specification

About this Specification

This specification presents the TrueOrder KDS API (Application Programming Interface), which is served over HTTP for customers looking for deeper integration with the KDS. The endpoint accepts HTTP POST requests to provide details on active orders; the client can either poll order status or register for callbacks from the KDS for actions such as item and order bumps and new order entry.

Table of Contents

1. Introduction

The Epson TrueOrder KDS traditionally parses incoming “print jobs” from a Point of Sale System (POS) to an Epson POS printer. Each POS system that the Epson KDS supports requires a “parser” module to understand the format of the print job. This parser module is configured using the KDS Configuration Utility and the format must be followed from then on.

This method has advantages in that it is very easy to have a working KDS system out of the box for POS systems that are already supported. There is no integration with the POS required. For a POS that is not supported it is not difficult for Epson to create a new parser module – typically within 1-2 weeks.

Some disadvantages are that the POS may not have the flexibility to deliver all types of orders the Epson KDS is capable of handling -. i.e. Void orders or order Appends. Also, as the data is a print job there is no mechanism to provide feedback to the POS about the status of any given order.

This document outlines a specification for an “API” or Application Programming Interface to the Epson KDS system. The API provides an alternative direct way of communicating with the Epson KDS that not only allows data to be sent to the Epson KDS but also for KDS data to be returned. This allows for better control and feedback from the Epson KDS system.

2. API Interface

Data is exchanged between the POS and KDS using JSON format, which is described later in this document. The communication between POS and KDS is done using HTTP port 80 (and port 443 if SSL is enabled):

http(s)://192.168.192.168/cgi-bin/kdsapi/service.cgi

HTTP POST (*MAIN DELIVERY OPTION*) – JSON data will be the body of HTTP POST request and response using the following Content-Type:

```
Content-Type: application/json.
```

3. API and POS Parsers

Traditionally, TrueOrder KDS allows a parser to be configured for the KDS to understand the POS data. A POS parser can still be selected and used in conjunction with the KDS API to provide flexibility for customers who wish to have multiple inputs to the POS system.

The POS parser would still listen to the traditional inputs -> TCP port 9100 (or ePOS XML). The KDS API service is listening on TCP port 80 (and port 443 if SSL is enabled).

A POS parser needs to be selected when configuring the KDS. If a traditional POS parser is not required, one can simply select the default “Epson” parser to allow the system to be configured.

4. Typical Order Receipt

A sample of a typical order receipt or kitchen chit (printout):

```
Server William
Table 7     CK# 0012
--------------------
1  Steak Burger
1  Fries
    No salt
1  Coke
    Regular
    No ice
```

It can be broken down into the following basic fields:

- Server Name
- Table Number
- Check Number
- Item Qty
- Item Description
- Modifier(s)

TrueOrder KDS can show other miscellaneous information within the order such as course info, seat info, order types, customer name, customer phone number, customer email etc. This is described elsewhere within this specification.

4.1. JSON Schema

The interface to the KDS API is based on the following JSON schema:

```
{
   "seq" : number,
   "type" : "string",
   "server" : "string",
   "table" : "string",
   "check" : "string",
   "cust_name" : "string",
   "cust_phone" : "string",
   "cust_email" : "string",
   "itemlist" :
   [
      {
         "itemid" : number,
         "item" : "string",
         "header" : "string",
         "label" : "string",
         "qty" : number,
         "modifierlist" :
         [
            {
               "modifier" : "string",
               "colour" : "string"
            }
         ]
      }
   ]
}
```

4.1.1. JSON Schema: Detailed Description

- `"seq"` – JSON payload sequence number. Should be incremented by 1 for every API call to the Epson KDS. Range 0 – 1048575. Once the upper limit is reached it should roll back to 0.
- `"type"` – Order type. Can be `"new"`, `"void"`, `"append"`, `"status"`, `"callback"`. Each order type is described later.

- "server" – Server name as it would be displayed in the KDS order header. It could be replaced with other text such as "Pickup", or "Delivery". While 0-35 ASCII characters are accepted the server name should be kept small so as to be displayed fully in the order header.
- "table" – Table number as displayed in the KDS order header. 0-20 ASCII characters are accepted however the table number should be kept small to be displayed fully in the order header.
- "check" – Check number for the order as displayed in the KDS order header. This check number should be unique for every new order entered into the system. 0-20 ASCII characters are accepted however the check number should be kept small to be displayed fully in the order header.
- "cust_name" – Optional field indicating customer name. 0-35 ASCII characters are accepted.
- "cust_phone" – Optional field indicating customer phone number. Must be in one of the following formats:

  (555)555-5555

  555-555-5555

  5555555555
- "cust_email" – Optional field indicating customer e-mail address. 0-35 ASCII characters are accepted.

The "cust_name", "cust_phone", "cust_email" are all optional and can be included or omitted in any combination.

Only 1 of the following 3 fields (item, header, label) should be sent:

- "item" – Description of the item as displayed on the Epson KDS. E.g. "Hamburger". 0-35 ASCII characters.
- "header" – Any information that should be displayed at the top of the order (below the KDS Order Header). Eg. "Dine-In, Take-Out. There can be multiple header fields but each should correspond to its own itemid. 0-35 ASCII characters.
- "label" – Any miscellaneous information to be displayed within the order. E.g. "Course 1", "Seat 2", or "Bar Seat 11". 0-35 ASCII characters.

- "itemid" – The unique identifier assigned to an item by the POS. The itemid needs to be unique within an order and is used by the KDS to easily associate to an item. The best practice is for the first item in the list to have an id of 1 and then increase for every item. Numbers 1-99 are supported.
- "modifier" – Description of the modifier as displayed on the Epson KDS. 0-35 ASCII characters.
- "colour" – can be "normal" (default) or "alert" to show the text in the alert color.

The qty (quantity) following field is optional:

- "qty" – Quantity of the item as displayed on the KDS. Numbers 1-99 are supported.

4.2. Sample JSON Payload

```
{
   “seq” : 1,
   “type” : “new”,
   “server” : “William”,
   “table” : “7”,
   “check” : “12”,
   “cust_name” : “Robert”,
   “cust_phone” : “5555555555”,
   “cust_email” : “Robert@Robert.com”,
   “itemlist” :
   [ {
         “itemid” : 1,
         “qty” : 1,
         “item” : “Steak Burger”
      },
      {
         “itemid” : 2,
         “qty” : 1,
         “item” : “Fries”,
         “modifierlist” :
         [ {
               “modifier” : “No salt”,
               “colour” : “alert”
            }
         ]
      },
      {
         “itemid” : 3,
         “qty” : 1,
         “item” : “Coke”,
         “modifierlist” :
         [ {
               “modifier” : “Regular”
            },
            {
               “modifier” : “No ice”
            }
         ]
      }
   ]
}
```

4.3. Response Error Codes

The KDS will send back a response indicating success or failure. Upon success the following is returned:

```
{
   "errorcode" : 0,
   "description" : "success"
}
```

Upon failure, an appropriate "errorcode" from the list below is returned:

- 1 – json syntax error
- 2 – callback entry does not exist
- 3 – no callbacks are registered
- 4 – order does not exist
- 5 – invalid 'callbackid'
- 6 – invalid callback 'trigger'
- 7 – invalid callback 'url'
- 8 – reached maximum order number
- 9 – invalid json parameter
- 10 – missing json parameter
- 101 – data format error
- 104 – timeout, no response from kds

5. Order Types

5.1. “new” Order

5.1.1. Request to KDS

Request for a brand-new order to be sent to the KDS:

```
{
    “seq”  : 1,
    “type” : “new”,
    “server” : “William”,
    “table” : “7”,
    “check” : “12”,
    “cust_name” : “Robert”,
    “cust_phone” : “5555555555”,
    “cust_email” : “Robert@Robert.com”,
    “itemlist” :
    [
        {
            “itemid” : 1,                 Unique positive integer number within the order context
            “header” : “Dine-In”          Will be displayed only once on screen (“header” type)
        },
        {
            “itemid” : 2,
            “header” : “Address: 185 Renfrew Dr.”
        },
        {
            “itemid” : 3,
            “label” : “--- Course 1 ---”          Will be displayed as a special not-
                                                  selectable item on the screen
                                                  multiple times (“label” type)
        },
        {
            “itemid” : 4,
            “qty” : 1,
            “item” : “Steak Burger”
        },
        {
            “itemid” : 5,
            “qty” : 1,
            “item” : “Fries”,
            “modifierlist” :
            [
                { “modifier” : “No salt”, “colour” : “alert” }
            ]
        },
        {
            “itemid” : 6,
            “label” : “--- Course 2 ---”
        },
        {
            “itemid” : 7,
            “qty” : 1,
            “item” : “Coke”,
            “modifierlist” :
            [
                { “modifier” : “Regular” },
            ]
        }
    ]
}
```

5.1.2. Response from KDS

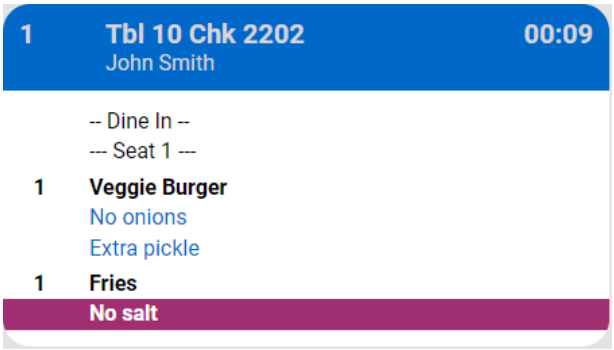
The KDS sends back a response indicating success or failure. Response error codes are listed in Section 4.3.

5.1.3. Example

JSON

```
{
    "seq": 1,
    "type": "new",
    "server": "John Smith",
    "table": "Tbl 10",
    "check": "Chk 2022",
    "itemlist":
    [
        {
            "itemid": 1,
            "header": "-- Dine In --"
        },
        {
            "itemid": 2,
            "label": "--- Seat 1 ---"
        },
        {
            "itemid": 3,
            "qty": 1,
            "item": "Veggie Burger",
            "modifierlist":
            [
                {
                    "modifier": "No onions"
                },
                {
                    "modifier": "Extra pickle"
                }
            ]
        },
        {
            "itemid": 4,
            "qty": 1,
            "item": "Fries",
            "modifierlist":
            [
                {
                    "modifier": "No salt",
                    "colour": "alert"
                }
            ]
        }
    ]
}
```

On-Screen Order



5.2. “append” Order

5.2.1. Request to KDS

Request for an addition to be sent to an already existing order. The check number should match a previous order:

```
{
   “seq” : 2,
   “type” : “append”,
   “server” : “William”,
   “table” : “7”,
   “check” : “12”
   “itemlist” :
   [
      {
         “itemid” : 8,      Unique positive number within an order
         “qty” : 1,
         “item” : “Classic Burger”,
         “modifierlist” :
         [
            { “modifier” : “No pickle” }
         ]
      }
   ]
}
```

The difference between “new” and “append” order is that appended orders will show the quantity column in a different background color on the KDS display to indicate the order was appended.

5.2.2. Response from KDS

The KDS sends back a response indicating success or failure. Response error codes are listed in Section 4.3.

5.2.3. Example

Here we append to the same order sent in Example 5.1.3.

JSON

```
{
    "seq": 2,
    "type": "append",
    "server": "John Smith",
    "table": "Tbl 10",
    "check": "Chk 2202",
    "itemlist": [
        {
            "itemid": 4,
            "qty": 1,
            "item": "Classic Hamburger",
            "modifierlist": [
                {
                    "modifier": "No Pickle"
                }
            ]
        }
    ]
}
```

On-Screen Order

5.3. “void” Order

5.3.1. Request to KDS

Voiding the order will be facilitated with ‘itemid’ field to un-ambiguously find the exact item within the order.

When voiding “Fries” and “Coke” from above order, POS will send

```
{
   “seq” : 3,
   “type” : “void”,
   “server” : “William”,
   “table” : “7”,
   “check” : “12”,
   “itemlist” :
   [
      { “itemid” : 5 },      Fries
      { “itemid” : 6 }       Coke
   ]
}
```

To void the whole order, POS will send request without item list.

```
{
   “seq” : 3,
   “type” : “void”,
   “server” : “William”,
   “table” : “7”,
   “check” : “12”
}
```

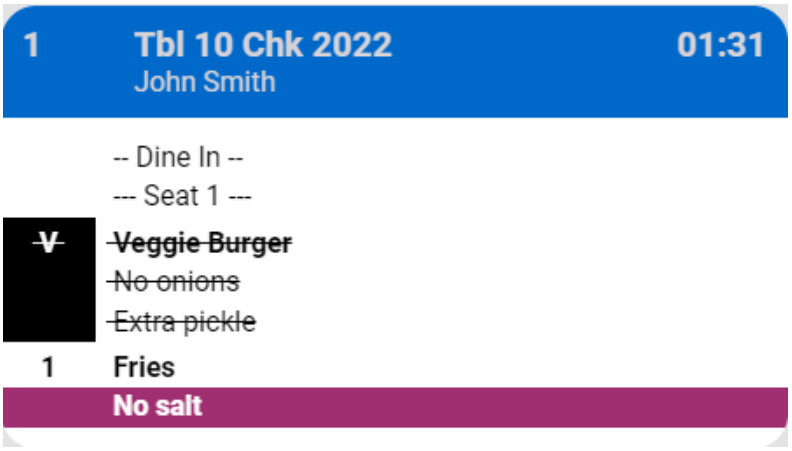
5.3.2. Response from KDS

The KDS sends back a response indicating success or failure. Response error codes are listed in Section 4.3.

5.3.3. Example

Here we void a single item, and then void the entire order from Section 5.1.3.

JSON	On-Screen Order

Single item void:
{
"seq": 1,
"type": "void",
"server": "John Smith",
"table": "Tbl 10",
"check": "Chk 2022",
"itemlist": [
{
"itemid": 3
}
]
}
	

Full order void:
{
"seq": 1,
"type": "void",
"server": "John Smith",
"table": "Tbl 10",
"check": "Chk 2022"
}
	

5.4. “status” of Order(s)

5.4.1. Requests to KDS

5.4.1.1. Single Order Status Request

For a single order,

```
{
   “seq” : 4,
   “type” : “status”,
   “statusof” : “ordersonly”,
   “orderlist” :
   [
      {
         “table” : “7”,
          “check” : “12”
      }
   ]
}
```

5.4.1.2. Multiple Orders Status Request

For multiple orders, `orderlist` will contain multiple `check numbers`:

```
{
   “seq” : 5,
   “type” : “status”,
   “statusof” : “ordersonly”,
   “orderlist” :
   [
      {
         “table” : “Table 7”,
          “check” : “12”
      },
      {
         “table” : “Table 8”,
          “check” : “14”
      },
      {
         “table” : “Table 9”,
          “check” : “15”
      }
   ]
}
```

If `orderslist` is missing from the request to the KDS, then it implies all active orders, in which case all active orders will indeed be returned.

Furthermore, the parameter `statusof` can be:

- `"ordersonly"` – for the status of orders, and the counts of active, bumped, voided items.
- `"ordersanditems"` – same as `"ordersonly"` (i.e. order status and item counts), plus the status of all their items.

5.4.2. "status" Response from KDS

There are some extra fields in the response in addition to "errorcode":

`state` of order can be:

- `"active"` – still cooking. This means, at least one item is `"active"`.
- `"bumped"` – the order has been completely bumped.
- `"voided"` – the order has been completely voided

Similarly, the `state` of item can be:

- `"active"` – still cooking
- `"bumped"` – the item has been bumped
- `"voided"` – the item has been voided

For `"active"` orders only, `timeout` of order will be one of:

- `"normal"` – normal timer
- `"priority"` – the timer reached Priority Time
- `"rush"` – the timer reached Rush Time

If the order is `"active"`, then `elapsedtime` is the number of second since the order was created, and it may have different values when the next POS request is made. If the order is `"bumped"` or `"voided"`, then the `elapsedtime` is the number of second it took for the order to be bumped or voided, and it would remain the same for subsequent POS requests.

There are 4 distinct item counts in the response:

- "activeitemcount" – the number of active items
- "bumpeditemcount" – the number of bumped items
- "voideditemcount" – the number of voided items
- "totalitemcount" – the number of items in total, which is sum of above counts.

5.4.2.1. "ordersonly" Response Type

If the request was "ordersonly", then KDS will return order status and item counts.

```
{
   "errorcode" : 0,
   "description" : "success",
   "orderlist" :
   [
      {
         "check" : "15",
         "table" : "Table 9",
         "server" : "William",
         "cust_name" : "Robert",
         "cust_phone" : "5555555555",
         "cust_email" : "Robert@Robert.com",
         "state" : "active",
         "elapsedtime" : 120,
         "timeout" : "normal",
         "activeitemcount" : 2,
         "bumpeditemcount" : 1,
         "voideditemcount" : 1,
         "totalitemcount" : 4
      }
   ]
}
```

5.4.2.2. “ordersanditems” Response Type

If the request was “ordersanditems”, then KDS will return order status, item counts, and item status in further detail.

```
{
    “errorcode” : 0,
    “description” : “success”,
    “orderlist” :
    [
        {
            “check” : “15”,
            “table” : “Table 9”,
            “server” : “William”,
            “cust_name” : “Robert”,
            “cust_phone” : “5555555555”,
            “cust_email” : “Robert@Robert.com”,
            “state” : “active”,
            “elapsedtime” : 120,
            “timeout” : “normal”,

            “activeitemcount” : 2,
            “bumpeditemcount” : 1,
            “voideditemcount” : 1,
            “totalitemcount” : 4

            “itemlist” :
            [
                {
                   “itemid” : 2,
                   “state” : “active”,
                  “qty” : 1,
                  “item” : “Classic Burger”,
                  “modifierlist” :
                  [
                      { “modifier” : “No pickle” }
                   ]
                },
                {
                   “itemid” : 3,
                   “state” : “bumped”
                  “qty” : 1,
                  “item” : “Fries”,
                },
                {
                   “itemid” : 4,
                   “state” : “voided”
                  “qty” : 1,
                  “item” : “Coke”,
                },
                {
                   “itemid” : 5,
                   “state” : “active”
                  “qty” : 1,
                  “item” : “Classic Burger”,
                }
            ]
        }
    ]
}
```

5.4.2.3. “status” Error Response

If there is an error responding to a “status” request on the KDS side, the response will contain the appropriate error code as per Section 4.3.

6. Callback Registration

Callbacks can be registered with a KDS device for the customer's own server to receive information for various events, such as new order entry, or an order or item bump.

6.1. "callback" Request to KDS

```
{
   "seq" : 11,
   "type" : "callback",
   "callbackid" : 1,
   "action" : "set",
   "url" :
   "http://192.168.1.100:8080/kdsapi/callback_listener.cgi",
   "trigger" : "onbump"
}
```

Where:

- `url` can be standard HTTP POST, like
  - `"http://192.168.1.100:8080/kdsapi/callback_listener.cgi"`
- `callbackid` is unique identifier assigned to callback entry by POS. This identifier will be used to delete the entry later. Should start at 1 and increment by 1 for each new call back registration.
- `action` can be
  - `"set"` – to create or overwrite a callback entry,
  - `"clear"` – to delete the entry,
- `trigger` is the condition that causes KDS to call back. It can be
  - `"onenter"` – KDS will return both `check, table` and `itemid(s)` when an order is entered.
  - `"onbump"` – KDS will return both `check, table` and `itemid(s)` when a bump takes place.
  - `"onpriority"` – when the timer of order reaches Priority Time, KDS will return `check, table.`
  - `"onrush"` – when the timer of order reaches Rush Time, KDS will return `check, table.`

- "onunbump" – KDS will return both `check, table` and `itemid` when an item is unbumped.
- "onrecall" – KDS will return both `check, table` when an order recall takes place.

6.2. "callback" Response from KDS

The KDS sends back a response indicating success or failure. Response error codes are listed in Section 4.3.

7. Callback Notifications from KDS

7.1. “onenter” Callback

The “new”, “append”, “void” requests will all generate “onenter” callback, with type “new”, “append”, or “void” separately. If the trigger is “onenter” then the KDS will send the original print job to the registered callback URL in the following JSON format. Optional “cust_name”, “cust_phone” and “cust_name” will be included if these fields were included in “new” order request by the POS.

- **Callback for “new”**

```
{
    "type" : "new",
    "check" : "0025",
    "table" : "Table 15",
    "server" : "Mark",
    "cust_name" : "Robert",
    "cust_phone" : "5555555555",
    "cust_email" : "Robert@Robert.com",
    "itemlist" :
    [  {
            "itemid" : 1,
            "qty" : 1,
            "item" : "Meat Lover",
            "modifierlist" :
            [  {
                    "modifier" : "> Olives",
                    "color" : "alert"
                },
                {
                    "modifier" : "> NO Mushrooms"
                }
            ]
        },
        {
            "itemid" : 2,
            "label" : "- Also sent to:Station 1"
        },
        {
            "itemid" : 3,
            "label" : "- Also sent to:Pizza Oven"
        }
    ]
}
```

- **Callback for “append”**

```
{
   "type" : "append",
   "check" : "0025",
   "table" : "Table 15",
   "server" : "Mark",”
   "cust_name" : "Robert",
   "cust_phone" : "5555555555",
   "cust_email" : "Robert@Robert.com",
   “itemlist” :
   [
      {
         “itemid” : 8,
         “qty” : 1,
         “item” : “Classic Burger”,
         “modifierlist” :
         [
            { “modifier” : “No pickle” }
         ]
      }
   ]
}
```

- **Callback for “void”**

Callback generated by void item 1 of above example will be:

```
{
   "type" : "void",
   "check" : "0025",
   "table" : "Table 15",
   "server" : "Mark",”
   "cust_name" : "Robert",
   "cust_phone" : "5555555555",
   "cust_email" : "Robert@Robert.com",
   “itemlist” :
   [
         {
             "itemid" : 1,
             "qty" : 1,
             "item" : "Meat Lover",
             "modifierlist" :
             [
                 {
                     "modifier" : "> Olives",
                     "color" : "alert"
                 },
                 {
                     "modifier" : "> NO Mushrooms"
                 }
             ]
         }
   ]
}
```

Note that a callback generated by full order void will not contain any item:

```
{
   "type" : "void",
   "check" : "0025",
   "table" : "Table 15",
   "server" : "Mark",”
   "cust_name" : "Robert",
   "cust_phone" : "5555555555",
   "cust_email" : Robert@Robert.com
}
```

7.2. “onbump” Callback

If the trigger is “onbump” and an order is bumped, then KDS will send check, table of the bumped order. Optional “cust_name”, “cust_phone” and “cust_name” will be included if these fields were included in “new” order request.

```
{
   “type” : “callback”,
   “callbackid” : 1,
   “trigger” : “onbump”,
   “check” : “12”,
   “table” : “Table 7”
   “cust_name” : “Robert”,
   “cust_phone” : “5555555555”,
   “cust_email” : “Robert@Robert.com”
}
```

If the trigger is “onbump” and an item is bumped (on SCROLL mode), then KDS will send the same content as order bump, together with itemid, itemname and modifiers for this item.

```
{
   “type” : “callback”,
   “callbackid” : 1,
   “trigger” : “onbump”,
   “check” : “12”,
   “table” : “Table 7”,
   “cust_name” : “Robert”,
   “cust_phone” : “5555555555”,
   “cust_email” : “Robert@Robert.com”,
   “itemid” : 8
   “item” : “Classic Burger”,
   “modifierlist” :
   [
      { “modifier” : “No pickle” }
   ]
}
```

7.3. “onpriority” Callback

If the trigger is “onpriority” or “onrush”, and an order becomes Priority or Rush, respectively, then KDS will send check, table of that order.

```
{
   “type” : “callback”,
   “callbackid” : 2,
   “trigger” : “onpriority”,
   “check” : “12”,
   “table” : “Table 7”
}
```

7.4. “onrush” Callback

```
{
   “type” : “callback”,
   “callbackid” : 3,
   “trigger” : “onrush”,
   “check” : “12”,
   “table” : “Table 7”
}
```

7.5. “onunbump” Callback

If the trigger is “onunbump” and an order item is unbumped (on SCROLL mode or directly unbump from touch screen), then KDS will send check, table of the bumped order and the bumped itemid, itemname and modifierlist for the item.

```
{
   “type” : “callback”,
   “callbackid” : 1,
   “trigger” : “onunbump”,
   “check” : “12”,
   “table” : “Table 7”,
   “cust_name” : “Robert”,
   “cust_phone” : “5555555555”,
   “cust_email” : “Robert@Robert.com”,
   “itemid” : 8,
   “item” : “Classic Burger”,
   “modifierlist” :
   [
      { “modifier” : “No pickle” }
   ]
}
```

7.6. “onrecall” Callback

If the trigger is “onrecall” and an order is recalled, then KDS will send `check, table` of the bumped order.

```
{
   “type” : “callback”,
   “callbackid” : 1,
   “trigger” : “onrecall”,
   “check” : “12”,
   “table” : “Table 7”,
   “cust_name” : “Robert”,
   “cust_phone” : “5555555555”,
   “cust_email” : “Robert@Robert.com”
}
```

7.7. POS/Server Response

The Epson KDS ignores any response from the POS or customer server.

8. “status” of Callback

8.1. Request to KDS

```
{
   “type” : “status”,
   “seg” : 12,
   “statusof” : “callback”,
   “callbacklist” :
   [
      { “callbackid” : 1 },
      { “callbackid” : 2 },
      { “callbackid” : 3 }
   ]
}
```

If `callbacklist` is missing or empty, then it implies all existing callbacks.

8.2. Response from KDS

```
{
   “errorcode” : 0,
   “description” : “success”,
   “callbacklist” :
   [
      {
         “callbackid” : 1,
         “url” :
         “http://192.168.1.100:8080/kdsapi/callback_listener.cgi”,
         “trigger” : “onbump”
      },
      {
         “callbackid” : 2,
         “url” :
         “http://192.168.1.100:8080/kdsapi/callback_listener1.cgi”,
         “trigger” : “onpriority”
      }
      {
         “callbackid” : 3,
         “url” : “”,
         “trigger” : “”
   }]
}
```

If there is an error, then `errorcode` will be non-zero with appropriate `description`. List of error conditions are in Section 4.3.

9. Callback Behavior in Multi-Station Mode

In single station mode the behavior is simple: when a certain action occurs, and if the corresponding callback is registered, then the KDS station sends the callback data to the registered server URL (endpoint).

For multi-station mode, each KDS device will generate callbacks for actions related to its own display. However, the Master device (the configured “POS Connected” device) can generate callbacks for itself, as well as certain callbacks on behalf of other displays too. This behavior is described in the table below:

Table 9.1: Multi-Station Callback Behaviour

	Display Device (Non-Master)	Master Device (POS Connected Device)	
		POS Routing	KDS Routing
onenter	- New order: Trigger “new” onenter callback on the device if this order is displayed on it - Append: Trigger “append” onenter callback on the device if this append occurs on it - Void: Trigger “void” onenter callback on the device if this void occurs on it	Same as Display Device	- New order: All new order coming to the whole system trigger master “new” onenter callback - Append: All append action coming to the whole system trigger “append” onenter callback - Void: All void action coming to the whole system trigger “void” onenter callback
onpriority	Triggered by orders displayed on its screen hitting priority time	Same as Display Device	Same as Display Device
onrush	Triggered by orders displayed on its screen hitting rush time	Same as Display Device	Same as Display Device
onbump (order)	Triggered by order bumped from itself	Same as Display Device	Triggered by all order bump from all devices
onbump (item)	Triggered by item bumped from itself	Same as Display Device	Triggered by all item bump from all devices
onunbump (item)	Triggered by item unbumped from itself	Same as Display Device	Triggered by all item unbump from all devices
onrecall (order)	Triggered by item recalled from itself	Same as Display Device	Same as Display Device