



# Asclepius Photo Network

## Order Express

version 2.0.0

## Plugin Development API

version 2.0.0

## Overview:

The Order Express Plugin API allows custom logics to be inserted at various locations of the ordering process. With proper use of the API functions, you can modify default data values, add your business logics and even trigger third party software. The following is the topographic view of the API.

```
InitLibrary()  
  
    [for each order]  
  
    PreOrderTx()  
  
    OnShippingHandling()  
  
        OnBeginDataTx()  
  
            [for each picture uploading]  
  
                OnDataTx() [picture 0]  
                :  
                :  
                OnDataTx() [picture n-1]  
  
            OnEndDataTx()  
  
        PostOrderTx()  
  
ReleaseLibrary()
```

## Building Order Express plugins

1. First you need the header file “PluginAPI.h” for all the headers and definitions.
2. Open the template source file “template.cpp”.
3. Override functions as needed but remember **all function bodies must be present** even the function is not used.
4. **Make sure the plugin API version number is correct.**
5. Compile and build your plugin. Once it’s ready place it under folder “plugin” under the installation folder of Order Express.

## Function prototypes

### PluginAPIVersion

First function to be called by *Order Express* for API verification and further decipher library action as per API version.

```
ASC_API unsigned int PluginAPIVersion()
```

### PluginCaption

Supply the name of your plugin. This name will show up in the plugin list for identification.

```
ASC_API const TCHAR* PluginCaption()
```

### InitLibrary

Called once when the library is loaded. This is the place for parameter block initialization. The parameters that are setup here can be modified by user at run time. If you wish to setup any user data this is the place for it.

Currently the return code from this function is not propagated.

```
ASC_API int InitLibrary(void **pParam, void **pUserData1, void **pUserData2)
```

### ReleaseLibrary

Called once when the library is unloaded. Release any resources allocated in InitLibrary.

Return code is not propagated.

```
ASC_API int ReleaseLibrary(void *pParam, void *pUserData1, void *pUserData2)
```

### **OnShippingHandling**

This function is called to determine cost of shipping and handling for an on-line order. The values are later used during PayPal checkout.

**pCustomer (in)** - Points to 1024 char long buffer that contains customer Email address.

**pOrderNumber (in)** - Points to 1024 char long buffer that contains the order number. The default is the string form of iOrderSequenceNumber. If it is modified, the new value will be propagated through the rest this order transfer.

**pCountry (in)** - Points to 1024 char long buffer that contains name of country where the order is made.

**pRegion (in)** - Points to 1024 char long buffer that contains name of region where the order is made.

**pCity (in)** - Points to 1024 char long buffer that contains name of city where the order is made.

**pParam (in/out)** - Points to the parameter list defined in the InitLibrary() function.

**pUserData1 (in/out)** - Points to pUserData1 memory block defined in the InitLibrary() function.

**pUserData2 (in/out)** - Points to pUserData2 memory block defined in the InitLibrary() function.

**pShipping (out)** - Pointer to floating point variable that carries the shipping cost back to its caller.

**pHandling (out)** - Pointer to floating point variable that carries the handling cost back to its caller.

Return code is not propagated.

```
ASC_API int OnShippingHandling(TCHAR *pCustomer, TCHAR *pOrderNumber,
TCHAR *pCountry, TCHAR *pRegion, TCHAR *pCity, void *pParam, void
*pUserData1, void *pUserData2, float *pShipping, float *pHandling)
```

### **PreOrderTx**

This function is called when an order is received but before the actual data transfer. There are several string parameters that can be controlled by this function.

**iOrderSequenceNumber (in)** - The sequential order number for this order.

pRootFolder (in) - Points to 1024 char long buffer that contains root folder information.

pOrderNumber (in/out) - Points to 1024 char long buffer that contains the order number. The default is the string form of iOrderSequenceNumber. If it is modified, the new value will be propagated through the rest of this order transfer.

pDestination (in/out) - Points to 1024 char long buffer that contains the destination folder where the pictures will be stored. If it is modified, the new value will be propagated through the rest of this order transfer.

pParam (in/out) - Points to the parameter list defined in the InitLibrary() function.

pUserData1 (in/out) - Points to pUserData1 memory block defined in the InitLibrary() function.

pUserData2 (in/out) - Points to pUserData2 memory block defined in the InitLibrary() function.

Return code is not propagated.

```
ASC_API int PreOrderTx(int iOrderSequenceNumber, TCHAR *pRootFolder,
TCHAR *pOrderNumber, TCHAR *pDestination, void *pParam, void
*pUserData1, void *pUserData2)
```

### **PostOrderTx**

This function is called after the order has been transferred. Modify Order Number and/or Destination if needed.

pOrderNumber (in/out) - Points to 1024 char long buffer that contains the order number. The default is the string form of iOrderSequenceNumber. If it is modified, the new value will be propagated through the rest this order transfer.

pDestination (in/out) - Points to 1024 char long buffer that contains the destination folder where the pictures will be stored. If it is modified, the new value will be propagated through the rest of this order transfer.

pParam (in/out) - Points to the parameter list defined in the InitLibrary() function.

pUserData1 (in/out) - Points to pUserData1 memory block defined in the InitLibrary() function.

pUserData2 (in/out) - Points to pUserData2 memory block defined in the InitLibrary() function.

Return code is not propagated.

```
ASC_API int PostOrderTx(TCHAR *pOrderNumber, TCHAR *pDestination, void *pParam, void *pUserData1, void *pUserData2)
```

### **OnBeginDataTx**

This function is called when the picture transfer begins. At this stage the order details is available. Order details include file names, product code, paper size and code.

pCustomer (in/out) - Points to 1024 char long buffer that contains customer Email address.

pOrderNumber (in/out) - Points to 1024 char long buffer that contains the order number. The default is the string form of iOrderSequenceNumber. If it is modified, the new value will be propagated through the rest this order transfer.

pDestination (in/out) - Points to 1024 char long buffer that contains the destination folder where the pictures will be stored. If it is modified, the new value will be propagated through the rest of this order transfer.

pSelection (in) - Pointer to array of SELECTION structure. Each SELECTION denotes order details made by the customer. They can be accessed as pSelection[0], pSelection[1], ... , pSelection[iTotalSelection-1]

iTotalSelection (in) - Integer. Number of items in the SELECTION array.

pParam (in/out) - Points to the parameter list defined in the InitLibrary() function.

pUserData1 (in/out) - Points to pUserData1 memory block defined in the InitLibrary() function.

pUserData2 (in/out) - Points to pUserData2 memory block defined in the InitLibrary() function.

Return code is not propagated.

```
ASC_API int OnBeginDataTx(TCHAR *pCustomer, TCHAR *pOrderNumber, TCHAR *pDestination, SELECTION *pSelection, int iTotalselection, void *pParam, void *pUserData1, void *pUserData2)
```

### **OnDataTx**

This function is called at the end of each picture transfer.

iFileSequenceNumber (in) - Zero base sequence number of the picture that has just been uploaded.

pOrderNumber (in/out) - Points to 1024 char long buffer that contains the order number. The default is the string form of iOrderSequenceNumber. If it is modified, the new value will be propagated through the rest this order transfer.

pDestination (in/out) - Points to 1024 char long buffer that contains the destination folder where the pictures will be stored. If it is modified, the new value will be propagated through the rest of this order transfer.

pFileName (in/out) - Points to 1024 char long buffer that contains the file name of the picture just uploaded.

pSelection (in) - Pointer to the SELECTION structure that contains order information for the picture that just been uploaded.

pParam (in/out) - Points to the parameter list defined in the InitLibrary() function.

pUserData1 (in/out) - Points to pUserData1 memory block defined in the InitLibrary() function.

pUserData2 (in/out) - Points to pUserData2 memory block defined in the InitLibrary() function.

Return code is not propagated.

```
ASC_API int OnDataTx(int iFileSequenceNumber, TCHAR *pOrderNumber,
TCHAR *pDestination, TCHAR *pFileName, SELECTION *pSelection, void
*pParam, void *pUserData1, void *pUserData2)
```

### **OnEndDataTx**

This function is called when all the pictures have successfully been uploaded. It is called after the last OnDataTx() function is called.

pCustomer (in/out) - Points to 1024 char long buffer that contains customer Email address.

pOrderNumber (in/out) - Points to 1024 char long buffer that contains the order number. The default is the string form of iOrderSequenceNumber. If it is modified, the new value will be propagated through the rest this order transfer.

pDestination (in/out) - Points to 1024 char long buffer that contains the destination folder where the pictures will be stored. If it is modified, the new value will be propagated through the rest of this order transfer.

pSelection (in) - Pointer to array of SELECTION structure. Each SELECTION denotes order details made by the customer. They can be accessed as

pSelection[0], pSelection[1], ... , pSelection[iTotalSelection-1]

iTotalSelection (in) - Integer. Number of items in the SELECTION array.

pParam (in/out) - Points to the parameter list defined in the InitLibrary() function.

pUserData1 (in/out) - Points to pUserData1 memory block defined in the InitLibrary() function.

pUserData2 (in/out) - Points to pUserData2 memory block defined in the InitLibrary() function.

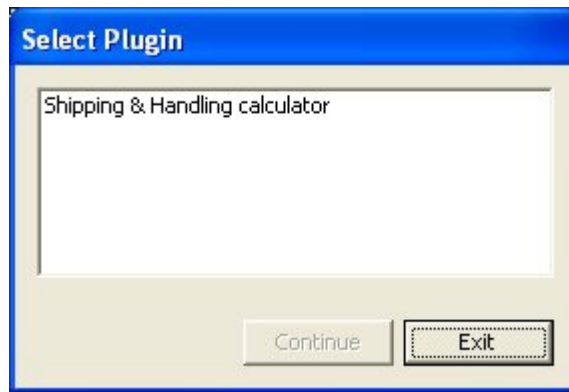
Return code is not propagated.

```
ASC_API int OnEndDataTx(TCHAR *pCustomer, TCHAR *pOrderNumber, TCHAR
*pDestination, SELECTION *pSelection, int iTotalselection, void
*pParam, void *pUserData1, void *pUserData2)
{
    return 1;
}
```

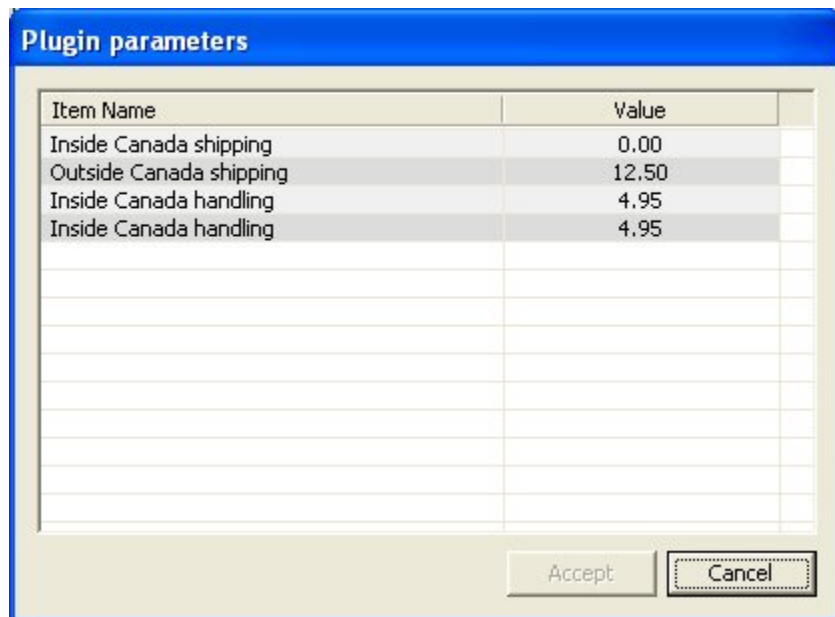
## Example:

The following example demonstrates the use of API functions to build an Order Express plugin. This plugin determines the shipping and handling charge base on the country code of the customer. In this example, the shipping and handling charge differs between Canadian and non-Canadian customers.

Once the dll is build, place it in folder "Plugin" under the Order Express installation folder. If the plugin is build correctly, when you start Order Express again, you will notice a new menu item "Plugin Parameters..." appears in the "System" menu. Click on this item and the new plugin "Shipping & handling calculator" will show up in the plugin list.



Select this new plugin and press "Continue" to access its parameters.



These values will be used when your customers make payment through PayPal for print orders. You can try changing the values to get a feeling of how easy it is to setup parameters for your own plugins. Notice that the parameter values are not persistent in this example. If you would like to make these values persistent, you will need to add code to either serialize the values to a data file, database or the registry. The serialization code should be implemented in `InitLibrary()` and `ReleaseLibrary()`.

### Example source code:

```
#include "stdafx.h"
#include "pluginexample.h"
#include "PluginAPI.h"

TCHAR *StringAlloc(const TCHAR *p)
{
    TCHAR *s = (TCHAR*)malloc((wcslen(p) + 1) * sizeof(TCHAR));
    wcscpy(s, p);
    return s;
}

void ReleaseParamBlock(PLUGINPARAMBLOCK *pBlk)
{
    int i = 0;
    while ( pBlk[i].eType != PLUGINPARAM_NONE )
    {
        free(pBlk[i].pItemName);
        switch(pBlk[i].eType)
        {
            case PLUGINPARAM_BOOL:
                break;
            case PLUGINPARAM_INT:
                break;
            case PLUGINPARAM_FLOAT:
                break;
            case PLUGINPARAM_STRING:
                free(pBlk[i].Value.String.pString);
                break;
            case PLUGINPARAM_MULTI:
                free(pBlk[i].Value.Multi.pItems);
                break;
        }
        i++;
    }
    free(pBlk);
}

ASC_API unsigned int PluginAPIVersion()
{
    return 0x2000000;
}

ASC_API const TCHAR* PluginCaption()
{
    return _T("Shipping & Handling calculator");
}
```

```

}

ASC_API int InitLibrary(void **pParam, void **pUserData1, void
**pUserData2)
{
    PLUGINPARAMBLOCK *pBlk =
(PLUGINPARAMBLOCK*)calloc(sizeof(PLUGINPARAMBLOCK), 5);
    //
    // This is an example of a series of floating point items.
    //
    pBlk[0].pItemName = StringAlloc(_T("Inside Canada shipping"));
    pBlk[0].eType = PLUGINPARAM_FLOAT;
    pBlk[0].Value.fValue = 0.0F;

    pBlk[1].pItemName = StringAlloc(_T("Outside Canada shipping"));
    pBlk[1].eType = PLUGINPARAM_FLOAT;
    pBlk[1].Value.fValue = 12.50F;

    pBlk[2].pItemName = StringAlloc(_T("Inside Canada handling"));
    pBlk[2].eType = PLUGINPARAM_FLOAT;
    pBlk[2].Value.fValue = 4.95F;

    pBlk[3].pItemName = StringAlloc(_T("Inside Canada handling"));
    pBlk[3].eType = PLUGINPARAM_FLOAT;
    pBlk[3].Value.fValue = 4.95F;

    pBlk[4].eType = PLUGINPARAM_NONE;

    *pParam = (void*)pBlk;
    return 1;
}

ASC_API int ReleaseLibrary(void *pParam, void *pUserData1, void
*pUserData2)
{
    PLUGINPARAMBLOCK *p = (PLUGINPARAMBLOCK*)pParam;
    ReleaseParamBlock(p);
    return 1;
}

ASC_API int OnShippingHandling(TCHAR *pCustomer, TCHAR *pOrderNumber,
TCHAR *pCountry, TCHAR *pRegion, TCHAR *pCity, void *pParam, void
*pUserData1, void *pUserData2, float *pShipping, float *pHandling)
{
    PLUGINPARAMBLOCK *pBlk = (PLUGINPARAMBLOCK*)pParam;
    if ( wcscmp(pCountry, _T("CANADA")) == 0 )
    {
        *pShipping = pBlk[0].Value.fValue;
        *pHandling = pBlk[2].Value.fValue;
    }
    else
    {
        *pShipping = pBlk[1].Value.fValue;
        *pHandling = pBlk[3].Value.fValue;
    }
    return 1;
}

```

```
}

ASC_API int PreOrderTx(int iOrderSequenceNumber, TCHAR *pRootFolder,
TCHAR *pOrderNumber, TCHAR *pDestination, void *pParam, void
*pUserData1, void *pUserData2)
{
    return 1;
}

ASC_API int PostOrderTx(TCHAR *pOrderNumber, TCHAR *pDestination, void
*pParam, void *pUserData1, void *pUserData2)
{
    return 1;
}

ASC_API int OnBeginDataTx(TCHAR *pCustomer, TCHAR *pOrderNumber, TCHAR
*pDestination, SELECTION *pSelection, int iTotalselection, void
*pParam, void *pUserData1, void *pUserData2)
{
    return 1;
}

ASC_API int OnDataTx(int iFileSequenceNumber, TCHAR *pOrderNumber,
TCHAR *pDestination, TCHAR *pFileName, SELECTION *pSelection, void
*pParam, void *pUserData1, void *pUserData2)
{
    return 1;
}

ASC_API int OnEndDataTx(TCHAR *pCustomer, TCHAR *pOrderNumber, TCHAR
*pDestination, SELECTION *pSelection, int iTotalselection, void
*pParam, void *pUserData1, void *pUserData2)
{
    return 1;
}
```