

# EASU XML/SOAP API

## Programmers Guide

EASU V1.0.0.1

Easify Ltd

By Richard Moore, Ross Tyler, John Blakeman



# EASU XML/SOAP API Programmers Reference

## Overview

The Easify Advanced Server Upgrade (EASU) provides Ecommerce connectivity to external applications/middleware/websites via a suite of XML/SOAP Web services that are installed on your Easify database server when you install EASU.

The use of XML/SOAP web services allows for a variety of platforms to be programmed to work with Easify including Microsoft .Net and PHP based web sites. It is also possible to access the EASU API from non web based systems such as network server applications, database servers or desktop applications.

The EASU XML/SOAP API gives you the following functionality:

- *Insert customer records from your website into Easify*
- *Insert orders from your website into Easify*
- *Insert order details (products) from your website into Easify*
- *Insert order payment details from your website into Easify*
- *Upload product information including stock levels from Easify to your website*
- *Retrieve Easify lookup table information from your website*

## Architecture

For Ecommerce the intended means of communication with Easify (via the EASU API) is either directly from the website to the EASU web services, or by creating custom middleware that sits between the website and Easify.

The architecture that you choose to adopt will depend on the requirements of the system in which you are working. If you have full control over the website you would probably choose to code the website to talk directly to the EASU web services. If you do not have full control over the website, but have access to an Ecommerce API on the web server you would probably choose to implement custom middleware that would communicate between the API of the website and the EASU API.

In this guide we will assume that the EASU API is being used with a website, and will refer to 'website' throughout. If you are programming middleware or other kind of external application the meaning of the document will still apply but you will have to make the mental translation.

## Web Service Connectivity

The EASU API consists of a suite of web services which are automatically installed on your Easify database server when you install EASU. If you have multiple companies running in Easify, a separate set of web services will be installed for each company that you install EASU for. This way you can link each separate company in Easify to a separate Ecommerce system.

The EASU web services are installed in IIS on your database server, and are accessible on the server via the URLs in the following table:

EASU Web Service URL (local URL as on database server)	Description
<a href="http://localhost/Easify/inbound/easifyinboundcustomers.asmx">http://localhost/Easify/inbound/easifyinboundcustomers.asmx</a>	Add customers to Easify from website
<a href="http://localhost/Easify/inbound/easifyinboundorders.asmx">http://localhost/Easify/inbound/easifyinboundorders.asmx</a>	Add orders to Easify from website
<a href="http://localhost/Easify/inbound/easifyinboundpayments.asmx">http://localhost/Easify/inbound/easifyinboundpayments.asmx</a>	Add order payment records to Easify from website
<a href="http://localhost/Easify/inbound/easifyinboundproducts.asmx">http://localhost/Easify/inbound/easifyinboundproducts.asmx</a>	Returns Easify Product Information from Easify to website

The URLs given in the above table are the URLs as they would exist for a company named 'Easify' when accessed from the Easify database server (localhost). The web service URL is constructed in the following format:

[https://<SERVER\\_NAME>/<COMPANY\\_NAME>/<PATH\\_TO\\_WEB\\_SERVICE>](https://<SERVER_NAME>/<COMPANY_NAME>/<PATH_TO_WEB_SERVICE>)

In the above example if your server name was [server.easifyelectricals.com](https://server.easifyelectricals.com) and your company name (with white space removed) was EasifyElectricalsLtd then the URL for your web services would be:

<https://server.easifyelectricals.com/EasifyElectricalsLtd/> followed by the path to the web service you require.

If you installed EASU for a second company named Easify Plumbing Ltd on the same server, then the URL would be:

<https://server.easifyelectricals.com/EasifyPlumbingLtd/> followed by the path to the web service you require.

For clarity in the remainder of this guide we will omit the server name and company name from web service URLs.

In addition to the web services mentioned above, EASU also provides a means for product updates to flow from Easify via EASU to your website. This is achieved by you implementing a simple web service on your web server, and pointing Easify to direct product updates to the web service. Source code for this is provided at the end of this guide in VB.Net and PHP.

### **Publishing the EASU Web Service Server**

In order for your EASU web services to be accessible by your web server, you will need to publish the web services to a public IP address. The way you do this will depend on what type of firewall you are using for your Easify database server.

In the first URL example we gave you would need to ensure that traffic to the URL

<https://server.easifyelectricals.com> was forwarded to the public IP address of your Easify database server. Or in the (preferred) case that your Easify database server has a private internal IP address and uses NAT, you would forward incoming HTTPS (TCP port 443) traffic to the Easify database server via rules on your firewall.

## Web Service Security

### SSL Encryption

In the above examples we have specified that the URL should be **HTTPS** so that the web service transactions are encrypted during transmission. We recommend that you always run web services over an SSL secure encrypted link; this will prevent 3<sup>rd</sup> parties eavesdropping on the contents of web service transmissions.

You can either use a self-signed certificate for this purpose, or purchase a 3<sup>rd</sup> party SSL certificate.

### IP Address Lockdown

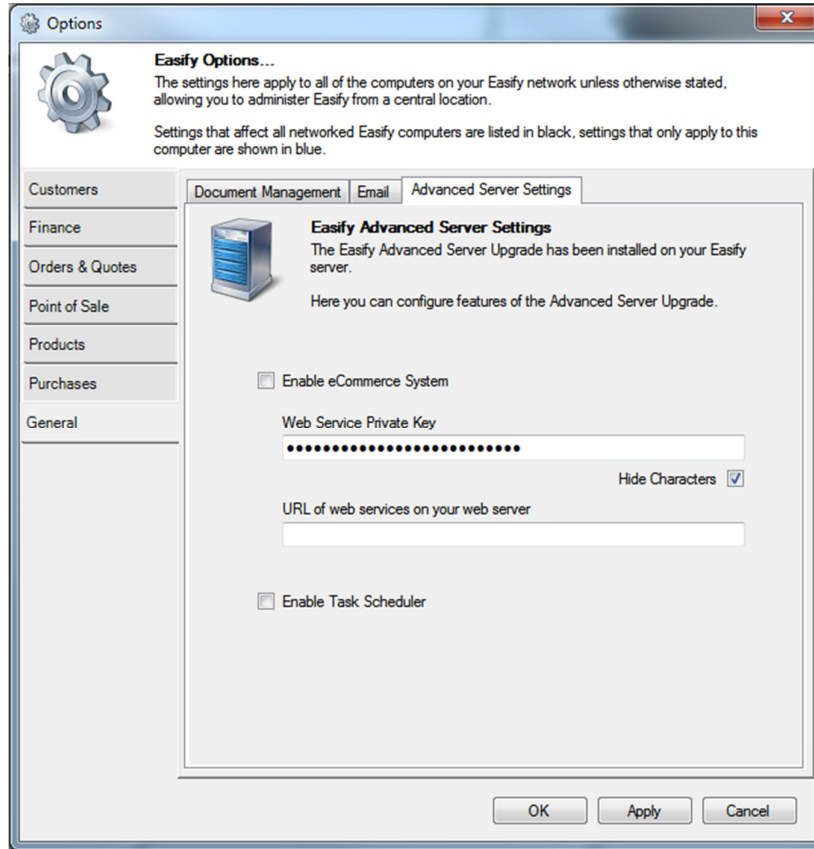
It is recommended that you lock down the communications between web server and EASU web services to only the IP addresses of both servers in your firewalls.

In other words if your web server IP address is **12 . 23 . 34 . 45** then lock down your EASU server to only accept incoming web service requests from the IP address **12 . 23 . 34 . 45**. Equally if you are using the EASU web services to upload product information to your web server then consider only accepting such traffic from the IP address of your EASU server.

### Private Key

As an added layer of security, all EASU API calls must include a private key (PK) as a parameter. The private key is stored in the Easify database for each company and can be configured from one of your Easify clients, in Easify click the **Tools->Options** menu, and in the Options window navigate to the **General->Advanced Server** Settings tab where you can enter a private key.

We recommend that you choose a private key that is unique to you, and that you make it longer than 16 characters, containing a mixture of upper and lowercase characters as well as numbers and non-alpha-numeric characters. The private key effectively acts as a password ensuring that even if an attacker does discover the URL of your Easify web services, spoofs your server IP address and manages to connect, they will not be able to invoke any web methods unless they know your private key.



## Getting Started

### Inserting orders from a web site into Easify

Probably the main use for the EASU XML/SOAP API is in automatically inserting orders that have been raised in an online shop on a web server into an Easify backend database.

In other words, an order that a customer places on a web site is automatically copied from the web-site to Easify.

Once EASU has been installed on Easify, and has been correctly published so that it is accessible from the web server that is hosting the web site, it is a relatively simple matter to write the code for the web server to copy new orders into Easify.

The majority of this guide assumes that you are using a Microsoft .Net web server and that your website has been created using ASP.Net and coded in the language VB.Net. We have also provided some code examples in PHP to illustrate how you can connect to Easify if you have a PHP based web server. A typical example of accessing an EASU web service using PHP is given on page 25.

### Fundamentals of inserting a new order from a website into Easify

For the purpose of this example we will assume that you have completed a sale on your website, and are just at the point of copying the sale over to Easify.

We will assume that you have access to the customer details, their order details, the items contained within the order, and their payment details.

Note that depending on your website PCI-DSS compliance requirements, you should not store credit card transaction details such as card number, CV2 codes, expiry dates etc... Also for the same reasons sensitive credit card information should not be recorded in Easify.

We will also assume that for your customer, order and each product in the order, you have a unique ID number for each that has been generated by your website. For instance for the customer you should have a website generated unique Customer Number and the order should have a unique website generated Order Number.

The basic process of inserting an order from your website is as follows:

1. Insert customer details from website into Easify
2. Insert order details from website into Easify
3. Add order details (products) from website to Easify
4. Insert order payment details from website into Easify

The remainder of this programmers guide provides a reference to the various web services and their methods that you will use in order to achieve the above.

## EasifyInboundCustomers Web Service

The EasifyInboundCustomers web service provides functionality to allow you to insert customer records from your web site into Easify.

It also includes helper methods to allow you to access lookup values that will be required when inserting customer records.

The following methods are supported by EasifyInboundCustomers:

Method	Description
InsertCustomer	Insert a new customer record into Easify
GetPaymentTerms	Get a list of payment terms as configured in Easify
GetRelationships	Get a list of customer relationships as configured in Easify
GetTypes	Get a list of customer types as configured in Easify

EasifyInboundCustomers surfaces the following classes:

Types
CustomerDetails
CustomerRelationships
CustomerTypes
PaymentTerms

EasifyInboundCustomers provides the following enumerations:

Enums	Description
InsertCustomerExceptions	Enumerated list of possible exceptions specific to customer insert
GeneralExceptions	Enumerated list of general exceptions

## EasifyInboundCustomers.InsertCustomer Method

Inserts a customer record from the web server into Easify.

To use this method, first instantiate a CustomerDetails class that will be passed as the first parameter. The ExtCustomerId value should be set to the Customer Id value of the customer record on the web server, this ensures that Easify can associate the new customer record in Easify with the customer record on the web server.

Where a customer record already exists in Easify (identified by the ExtCustomerId parameter which corresponds to the Easify customer no) the customer record in Easify will be updated with the new customer details.

There is no return value from this web method. Any exception encountered from the call will indicate that that insert was unsuccessful.

### Web Service URL

</Inbound/EasifyInboundCustomers.asmx?op=InsertCustomer>

### Parameters

Name	Type	Description	Required
Customer	CustomerDetails	Customer details class populated with customer details	Yes
PK	String	Easify Private Key	Yes

### VB.Net Example

```
' Assume we have imported the InboundCustomers web service
Dim EasifyInboundCustomersWS As New EasifyInboundCustomers
Dim Customer As New CustomerDetails

Try
    Customer.ExtCustomerId = "100000"
    Customer.CompanyName = "Some Test Co"
    Customer.Title = "Mr"
    Customer.FirstName = "Test"
    Customer.Surname = "Bloke"
    Customer.JobTitle = "Test Manager"
    Customer.Address1 = "23 Test Street"
    Customer.Town = "Sometown"
    Customer.County = "Countyshire"
    Customer.Postcode = "AA77 88JJ"
    Customer.Country = "United Kingdom"

    Customer.WorkTel = "01305 303040"
    Customer.Mobile = "N/A"
    Customer.Email = "info@sometestco.co.uk"
    Customer.Fax = "N/A"

    Customer.SubscribeToNewsletter = False
    Customer.TradeAccount = False
    Customer.CreditLimit = 0
    Customer.CustomerTypeId = 1
    Customer.PaymentTermsId = 1
    Customer.RelationshipId = 1

    ' Insert the customer into Easify
    EasifyInboundCustomersWS.InsertCustomer(Customer, "ChangeThisToYourPrivateKey")
Catch ex As SoapException
    ' See exception handling section at the end of this guide.
End Try
```

## CustomerDetails Class Details

Property	Type	Notes	Required
ExtCustomerId	String	Customer number as generated by the website. External to Easify.	Yes
Title	String		No
FirstName	String		No
Surname	String		No
JobTitle	String		No
Department	String		No
CompanyName	String		No
Address1	String		No
Address2	String		No
Address3	String		No
Town	String		No
County	String		No
Postcode	String		No
Country	String		No
WorkTel	String		No
HomeTel	String		No
Mobile	String		No
OtherTel	String		No
Fax	String		No
Email	String		No
WebAddress	String	Customers website address	No
Notes	String	Notes about the customer	No
SubscribeToNewsletter	Boolean	True if the customer wants to subscribe to newsletter	No
TradeAccount	Boolean	Whether the account is a trade account	No
CreditLimit	Decimal	The credit limit for the customer	No
DeliveryTitle	String		No
DeliveryFirstName	String		No
DeliverySurname	String		No
DeliveryJobTitle	String		No
DeliveryDepartment	String		No
DeliveryCompanyName	String		No
DeliveryAddress1	String		No
DeliveryAddress2	String		No
DeliveryAddress3	String		No
DeliveryTown	String		No
DeliveryCounty	String		No
DeliveryPostcode	String		No
DeliveryCountry	String		No
DeliveryTel	String		No
DeliveryEmail	String		No
PaymentTermsId	Integer	Reference to Easify PaymentTerm	Yes
RelationshipId	Integer	Reference to Easify CustomerRelationship	Yes
CustomerTypeId	Integer	Reference to Easify CustomerType	Yes

## EasifyInboundCustomers.GetPaymentTerms Method

Returns an array of Easify PaymentTerms objects.

You can call this method to obtain a list of all the payment terms that have been configured in Easify.

### Web Service URL

</Inbound/EasifyInboundCustomers.asmx?op=GetPaymentTerms>

### Parameters

Name	Type	Description	Required
PK	String	Easify Private Key	Yes

### VB.Net Example

```
Dim EasifyInboundCustomersWS As New EasifyInboundCustomers
Dim PaymentTermsArray() As PaymentTerms
Try
    PaymentTermsArray = EasifyInboundCustomersWS.GetPaymentTerms("ChangeThisToYourPrivateKey")
    For Each PaymentTerm In PaymentTermsArray
        Debug.Print(PaymentTerm.Description)
    Next
Catch ex As Exception
    ' See exception handling section at the end of this guide.
End Try
```

### PaymentTerms Class Details

Properties	Type	Description
Id	Integer	Unique Payment Terms Id
Description	String	Payment Terms description

## EasifyInboundCustomers.GetRelationships Method

Returns an array of Easify Customer Relationship objects.

You can call this method to obtain a list of all the customer relationships that have been configured in Easify. Customer relationships can be Lead, Prospect, Active etc... they define the customer's relation to your company.

### Web Service URL

</Inbound/EasifyInboundCustomers.asmx?op=GetRelationships>

### Parameters

Name	Type	Description	Required
PK	String	Easify Private Key	Yes

### VB.Net Example

```
' Assume we have imported the InboundCustomers web service
Dim EasifyInboundCustomersWS As New EasifyInboundCustomers
Dim RelationshipArray() As Relationship
Try
    RelationshipArray = EasifyInboundCustomersWS.GetRelationships("ChangeThisToYourPrivateKey")
    For Each Relationship In RelationshipArray
        Debug.Print(Relationship.Description)
    Next
Catch ex As Exception
    ' See exception handling section at the end of this guide.
End Try
```

### Relationships Class Details

Properties	Type	Description
Id	Integer	Unique Customer Relationship Id
Description	String	Customer Relationship description

## EasifyInboundCustomers.GetTypes Method

Returns an array of Easify CustomerTypes objects.

Call this method to retrieve a list of all the customer types that have been configured in Easify. Customer types are used to describe what type a customer is i.e. Commercial, Domestic, Government etc... The list of customer types can be customised in Easify.

### Web Service URL

</Inbound/EasifyInboundCustomers.asmx?op=GetTypes>

### Parameters

Name	Type	Description	Required
PK	String	Easify Private Key	Yes

### VB.Net Example

```
' Assume we have imported the InboundCustomers web service
Dim EasifyCustomer As New EasifyInboundCustomers
Dim CustomerTypesArray() As CustomerTypes

Try
    CustomerTypesArray = EasifyCustomer.GetTypes("ChangeThisToYourPrivateKey")
    For Each CustomerType In CustomerTypesArray
        Debug.Print(CustomerType.Description)
    Next
Catch ex As Exception
    ' See exception handling section at the end of this guide.
End Try
```

### CustomerTypes Class Details

Properties	Type	Description
Id	Integer	Unique Customer Type Id
Description	String	Customer Type description

## EasifyInboundCustomers Exception Enumerations

### InsertCustomerExceptions Enumeration

Value
ExtCustomerId_must_be_assigned_a_value
PaymentTermsId_must_be_assigned_a_value
RelationshipId_must_be_assigned_a_value
CustomerTypeId_must_be_assigned_a_value
EASIFY_INVALID_PRIVATE_KEY
EASIFY_UNCAUGHT_EXCEPTION
EASU_LICENSE_IS_NOT_VALID_OR_EXPIRED

### GeneralExceptions Enumeration

Value
EASIFY_INVALID_PRIVATE_KEY
EASIFY_UNCAUGHT_EXCEPTION
EASU_LICENSE_IS_NOT_VALID_OR_EXPIRED

## EasifyInboundOrders Web Service

The EasifyInboundOrders web service provides a way of inserting orders and related information from a website or middleware into Easify.

The following methods are supported by EasifyInboundOrders:

Methods	Description
AddInternalNote	Add an internal note to an existing Easify order
InsertOrder	Insert a new order into Easify
InsertOrderItem	Add a new item to an existing Easify order
GetItemStatuses	Retrieves a list of item statuses from Easify
GetStatuses	Retrieves a list of order statuses from Easify
GetTaxRates	Retrieves a list of supported tax rates from Easify
GetTypes	Retrieves a list of order types from Easify

EasifyInboundOrders surfaces the following classes:

Types
Order
OrderDetails
ItemStatuses
Statuses
OrderTypes
TaxRates

EasifyInboundOrders provides the following enumerations:

Enums	Description
AddInternalNoteExceptions	Enumeration of exceptions that may be thrown by the AddInternalNote method
InsertOrderExceptions	Enumeration of exceptions that may be thrown by the InsertOrder method
InsertItemExceptions	Enumeration of exceptions that may be thrown by the InsertItem method
GeneralExceptions	Enumeration of general exceptions that may be thrown by any method

## EasifyInboundOrders.AddInternalNote Method

Adds an internal note to an existing Easify order.

The order must already exist in Easify, and be referenced by the ExtOrderNo which is the order id that was originally obtained from the web server when the order was first inserted into Easify.

There is no return value from this web method. Any exception encountered from the call will indicate that that insert was unsuccessful.

### Web Service URL

</Inbound/EasifyInboundOrders.asmx?op=AddInternalNote>

### Parameters

Name	Type	Description	Required
ExtOrderNo	String	Reference to an existing order, this must be the Order Id from the web server.	Yes
Note	String	Note to be added	Yes
PK	String	Easify Private Key	Yes

### VB.Net Example

```
' Assume we have imported the InboundOrders web service
Dim EasifyOrderWS As New EasifyInboundOrders
Try
    ' The ExtOrderNo field must be the order no field you supplied when creating the order
    EasifyOrderWS.AddInternalNote("100010", "Just testing.", "ChangeThisToYourPrivateKey")
Catch ex As Exception
    ' See exception handling section at the end of this guide.
End Try
```

## EasifyInboundOrders.InsertOrder Method

Inserts a new order into Easify.

Note: A customer record associated with the order must exist before you call this method. When inserting a new order, first insert the new customer into Easify, and then insert the order into Easify using the web server's customer no for the ExtCustomerId property. The ExtOrderNo must be the order number obtained from the web server.

There is no return value from this web method. Any exception encountered from the call will indicate that that insert was unsuccessful.

### Web Service URL

</Inbound/EasifyInboundOrders.asmx?op=InsertOrder>

### Parameters

Name	Type	Description	Required
Order	Order	Class containing new order information	Yes
PK	String	Easify Private Key	Yes

### VB.Net Example

```
' Assume we have imported the InboundOrders web service
Dim wsEasifyOrder As New wsOrders.EasifyInboundOrders
Dim Ord As New Order

Try
    Ord.DateOrdered = Now()
    Ord.DatePlaced = Now()
    Ord.ExtCustomerId = "10000" ' ExtId of customer record already inserted
    Ord.ExtOrderNo = "100" ' Order Id obtained from web server

    Ord.DateInvoiced = Nothing
    Ord.DatePaid = Nothing
    Ord.Invoiced = True
    Ord.OrderType = 1
    Ord.Paid = True
    Ord.PaymentTermsId = 1
    Ord.StatusId = 1

    Ord.Comments = "New order placed via website."

    wsEasifyOrder.InsertOrder(Ord, "ChangeThisToYourPrivateKey")
Catch ex As Exception
    ' See exception handling section at the end of this guide.
End Try
```

### Order Class Details

Property	Type	Notes	Required
ExtOrderNo	String	Must be set to the order number of the order on the web server	Yes
ExtCustomerId	String	Reference to an existing Easify Customer record. This must be the customer id as on the web server	Yes
DatePlaced	Date	The date the order was created	Yes
StatusId	Integer	Reference to an Easify OrderStatus	Yes
OrderType	Integer	Reference to an Easify OrderType	Yes
PaymentTermsId	Integer	Reference to an Easify PaymentTerm	Yes
CustomerRef	String	<i>Reserved</i>	No

Comments	String	Order comments	Yes
UseTradeMargins	Boolean	Set true if order should use trade margins	No
Paid	Boolean	Set true if order has been paid in full	No
DatePaid	Date	Date the order was paid in full	No
Invoiced	Boolean	Set true if the order has been invoiced	No
DateInvoiced	Date	Date the order was invoiced	No
DateOrdered	Date	Date the order was placed. <b>Not required but you should set this as it is the date that the sales reports will pick up on.</b>	No
Scheduled	Boolean	Set true if order is to be scheduled in Easify	No
Priority	Integer	<i>Reserved</i>	No
Recurring	Boolean	<i>Reserved</i>	No
RecurTimePeriod	Integer	<i>Reserved</i>	No
DueDate	Date	Date the order should be scheduled for	No
DueTime	Date	Time that the order should be scheduled for	No
Duration	Decimal	Scheduled duration (hrs) for the order	No
DueDate2	Date	Date the order should be scheduled for (if 2 <sup>nd</sup> schedule enabled in Easify)	No
DueTime2	Date	Time that the order should be scheduled for (if 2 <sup>nd</sup> schedule enabled in Easify)	No
DueDuration2	Decimal	Scheduled duration (hrs) for the order (if 2 <sup>nd</sup> schedule enabled in Easify)	No

## EasifyInboundOrders.InsertOrderItem Method

Adds a product to an existing order in Easify.

This method should be called for every item that is to be added to the order in Easify. The order must already exist in Easify, so you should insert the order into Easify first and then proceed to add each item to the order in turn. The ExtOrderNo of each item should be set to the order number of the inserted order as obtained from the web server.

There is no return value from this web method. Any exception encountered from the call will indicate that that insert was unsuccessful.

### Web Service URL

</Inbound/EasifyInboundOrders.asmx?op=InsertOrderItem>

### Parameters

Name	Type	Description	Required
OrderDetails	OrderDetails	Details of the product being added to the order	Yes
AllowZeroPrice	Boolean	Set true if a zero price is allowed for this item	Yes
AllowZeroQty	Boolean	Set true if a zero quantity is allowed for this item	Yes
PK	String	Easify Private Key	Yes

### VB.Net Example

```
' Assume we have imported the InboundOrders web service
Dim EasifyOrderWS As New EasifyInboundOrders
Dim Product As New OrderDetails

Try
    Product.Sku = 100001
    Product.Price = 5.55
    Product.Qty = 1
    Product.ExtParentId = Nothing ' Reserved
    Product.ExtOrderNo = 100 ' The order number obtained from web server
    Product.ExtOrderDetailsId = 1 ' Order details id obtained from web server
    Product.AutoAllocateStock = True
    Product.TaxId = 1
    Product.TaxRate = 20

    EasifyOrderWS.InsertOrderItem(Product, False, False, "ChangeThisToYourPrivateKey")
Catch ex As Exception
    ' See exception handling section at the end of this guide.
End Try
```

### OrderDetails Class Details

Property	Type	Description	Required
Skus	String	Reference to an Easify Product SKU	Yes
Qty	Single	Quantity for the Order Item	Yes
Price	Decimal	Unit Price (net of VAT) for the Order Item	Yes
Comments	String	Comments associated with the Order Item	No
TaxRate	Single	Actual Tax Rate used for the Order	Yes
TaxId	Integer	Reference to an Easify Tax Rate	Yes
Spare	String	User definable field, typically used for serial numbers when enabled in Easify options	No
ExtParentId	Integer	<i>Reserved</i>	No
ExtOrderDetailsId	Integer	Reference for the order item, this must be the	Yes

		order details id from the web server.	
ExtOrderNo	Integer	Must be set to the order number of the order on the web server	Yes
AutoAllocateStock	Boolean	Set true to use Easify stock allocation system if enabled in Easify	Yes

## EasifyInboundOrders.GetItemStatuses Method

Returns an array of Easify OrderItemStatuses objects.

You can call this method to obtain a list of all possible order item statuses in Easify.

### Web Service URL

</Inbound/EasifyInboundOrders.asmx?op=GetItemStatuses>

### Parameters

Name	Type	Description	Required
PK	String	Easify Private Key	Yes

### VB.Net Example

```
' Assume we have imported the InboundOrders web service
Dim EasifyOrderWS As New EasifyInboundOrders
Dim StatusesArray() As wsOrders.OrderItemStatuses

Try
    StatusesArray = EasifyOrderWS.GetItemStatuses("ChangeThisToYourPrivateKey")
    For Each OrderItemStatus In StatusesArray
        Debug.Print(OrderItemStatus.Description)
    Next
Catch ex As Exception
    ' See exception handling section at the end of this guide.
End Try
```

### OrderItemStatuses Class Details

Property	Type	Description
Description	String	Order Item Status description

## EasifyInboundOrders.GetStatuses Method

Returns an array of Easify OrderStatuses objects.

You can use this method to obtain a list of the possible order statuses that an order can have in Easify.

### Web Service URL

</Inbound/EasifyInboundOrders.asmx?op=GetStatuses>

### Parameters

String Value	Type	Description	Required
PK	String	Easify Private Key	Yes

### VB.Net Example

```
' Assume we have imported the InboundOrders web service
Dim EasifyOrderWS As New wsOrders.EasifyInboundOrders
Dim OrderStatusesArray() As wsOrders.OrderStatuses

Try
    OrderStatusesArray = EasifyOrderWS.GetStatuses("ChangeThisToYourPrivateKey")
    For Each OrderStatus In OrderStatusesArray
        Debug.Print(OrderStatus.Description)
    Next
Catch ex As Exception
    ' See exception handling section at the end of this guide.
End Try
```

### OrderStatuses Class Details

Property	Type	Description
Id	Integer	Unique Order Status Id
Description	String	Order Status description
System	Boolean	Indicates whether the Order Status is a system status
DefaultType	Boolean	Indicates whether the Order status is the default Item status setting for Orders

## EasifyInboundOrders.GetTaxRates Method

Returns an array of Easify TaxRates objects.

You can use this method to obtain a list of tax rates that are used in Easify.

### Web Service URL

</inbound/easifyinboundorders.asmx?op=GetTaxRates>

### Parameters

Name	Type	Description	Required
Key	String	Easify Private Key	Yes

### VB.Net Example

```
' Assume we have imported the InboundOrders web service
Dim EasifyOrderWS As New wsOrders.EasifyInboundOrders
Dim TaxRatesArray() As wsOrders.TaxRates

Try
    TaxRatesArray = EasifyOrderWS.GetTaxRates("ChangeThisToYourPrivateKey")
    For Each TaxRate In TaxRatesArray
        Debug.Print(TaxRate.Description)
    Next
Catch ex As Exception
    ' See exception handling section at the end of this guide.
End Try
```

### TaxRates Class Details

Property	Type	Description
Id	Integer	Unique Tax Rate Id
Description	String	Tax Rate description
TaxCode	String	Tax Code
TaxRate	Double	Tax Rate
IsDefault	Boolean	Indicates whether this is the default Tax Rate applied to Order Items

## EasifyInboundOrders.GetTypes Method

Returns an array of Easify OrderTypes objects.

You can use this method to obtain a list of order types that are used in Easify.

### Web Service URL

</Inbound/EasifyInboundOrders.asmx?op=GetTypes>

### Parameters

Name	Type	Description	Required
PK	String	Easify Private Key	Yes

### VB.Net Example

```
' Assume we have imported the InboundOrders web service
Dim EasifyOrderWS As New EasifyInboundOrders
Dim OrderTypesArray() As OrderTypes

Try
    OrderTypesArray = EasifyOrderWS.GetTypes("ChangeThisToYourPrivateKey")
    For Each OrderType In OrderTypesArray
        Debug.Print(OrderType.Description)
    Next
Catch ex As Exception
    ' See exception handling section at the end of this guide.
End Try
```

### OrderTypes Class Details

Property	Type	Description
Id	Integer	Unique Order Type Id
Description	String	Order Type description

## EasifyInboundOrders Exception Enumerations

### AddInternalNoteExceptions

Value
ExtOrderNo_must_be_assigned_a_value
Order_does_not_exist_based_on_ExtOrderNo
Note_must_be_assigned_a_value
EASIFY_INVALID_PRIVATE_KEY
EASIFY_UNCAUGHT_EXCEPTION
EASU_LICENSE_IS_NOT_VALID_OR_EXPIRED

### InsertOrderExceptions

Value
ExtOrderNo_supplied_has_already_been_downloaded
ExtOrderNo_must_be_assigned_a_value
OrderType_must_be_assigned_a_value
StatusId_must_be_assigned_a_value
PaymentTermsId_must_be_assigned_a_value
ExtCustomerId_must_be_assigned_a_value
Comments_must_be_assigned_a_value
DatePlaced_must_be_assigned_a_value
Customer_does_not_exist_based_on_ExtCustomerId
EASIFY_INVALID_PRIVATE_KEY
EASIFY_UNCAUGHT_EXCEPTION
EASU_LICENSE_IS_NOT_VALID_OR_EXPIRED

### InsertOrderItemExceptions

Value
ExtOrderNo_must_be_assigned_a_value
Quantity_must_be_greater_than_0
Unit_Price_must_be_greater_than_0
Buy_Price_must_be_greater_than_0
ExtOrderDetailsId_must_be_assigned_a_value
SKU_must_be_assigned_a_value
TaxId_must_be_assigned_a_value
TaxRate_for_TaxId_is_different
Order_not_found_for_ExtOrderNo
Status_not_set_based_upon_STOCK_CODE_QUANTITY_ExtOrderNo
SKU_must_be_numeric
EASIFY_INVALID_PRIVATE_KEY
EASIFY_UNCAUGHT_EXCEPTION
EASU_LICENSE_IS_NOT_VALID_OR_EXPIRED

### GeneralExceptions

Value
EASIFY_INVALID_PRIVATE_KEY
EASIFY_UNCAUGHT_EXCEPTION
EASU_LICENSE_IS_NOT_VALID_OR_EXPIRED

## EasifyInboundPayments Web Service

The EasifyInboundPayments web service allows you to insert payment records related to an order from your website into Easify.

The following methods are supported by EasifyInboundPayments:

Methods	Description
InsertPayment	Adds a new payment record into Easify
GetPaymentMethods	Retrieves a list of supported payment types from Easify
GetPaymentAccounts	Retrieves a list of payment accounts from Easify

EasifyInboundPayments surfaces the following classes:

Types
PaymentDetails
PaymentAccounts
PaymentMethods

EasifyInboundPayments provides the following enumerations:

Enums	Description
InsertPaymentExceptions	Enumeration of exceptions that may be thrown by the InsertPayment method
GeneralExceptions	Enumeration of general exceptions that may be thrown by any method

## EasifyInboundPayments.InsertPayment Method

Inserts a payment record from the website into Easify.

An order must already exist in Easify, and be referenced by the ExtOrderNo which is the order id that was originally obtained from the web server when the order was first inserted into Easify.

There is no return value from this web method. Any exception encountered from the call will indicate that that insert was unsuccessful.

### Web Service URL

</Inbound/EasifyInboundPayments.asmx?op=InsertPayment>

### Parameters

Name	Type	Description	Required
Payment	PaymentDetails	Payment Details	Yes
PK	String	Easify Private Key	Yes

### VB.Net Example

```
' Assume we have imported the InboundPayments web service
Dim EasifyPaymentWS As New EasifyInboundPayments
Dim Payment As New wsPayments.PaymentDetails

Try
    Payment.ExtOrderNo = "100"
    Payment.Amount = 500
    Payment.PaymentAccountId = 1 ' Set which Easify account id you want payment into
    Payment.PaymentMethodId = 2 ' Set to relevant payment method id
    Payment.PaymentTypeId = 3 ' Set to relevant payment type id
    Payment.PaymentDate = Now()
    Payment.TransactionRef = "12345"
    Payment.Comments = "Payment via website"

    EasifyPaymentWS.InsertPayment(Payment, "ChangeThisToYourPrivateKey")
Catch ex As Exception
    ' See exception handling section at the end of this guide.
End Try
```

### PHP Example

```
<?php
/* Example EASU Inbound Payments Web Service */
$client = new
SoapClient("http://<COMPANY_DOMAIN>/<COMPANY_NAME>/Inbound/EasifyInboundPayments.asmx?wsdl");
$PrivateKey = "ChangeThisToYourPrivateKey";
$client->InsertPayment(array("Payment" => array("PaymentDate" => "2011-10-03T09:00:00",
    "PaymentAccountId" => 1, // Set which Easify account id you want payment into
    "TransactionRef" => "12345",
    "PaymentMethodId" => 2, // Set to relevant payment method id
    "PaymentTypeId" => 3, // Set to relevant payment type id
    "ExtOrderNo" => 100,
    "Comments" => "Payment via website",
    "Amount" => 500),
    "PK" => $PrivateKey));
?>
```

### PaymentDetails Class Details

Property	Type	Description	Required
PaymentDate	Date	Date the Payment was made	Yes
PaymentAccountId	Integer	Reference to a valid Easify PaymentAccount Id	Yes
TransactionRef	String	A transaction reference for the Payment	No
PaymentMethodId	Integer	Reference to a valid Easify PaymentMethod Id	Yes

PaymentTypeId	Integer	Reference to a valid Easify PaymentTypes Id	Yes
ExtOrderNo	String	Must be set to the order number of the order on the web server	Yes
Comments	String	Any comments associated with the Payment	No
Amount	Decimal	Payment amount	Yes

## EasifyInboundPayments.GetPaymentAccounts Method

Retrieves a list of payment accounts objects from Easify.

### Web Service URL

</Inbound/EasifyInboundPayments.asmx?op=GetPaymentAccounts>

### Parameters

Name	Type	Description	Required
PK	String	Easify Private Key	Yes

### VB.Net Example

```
' Assume we have imported the InboundPayments web service
Dim EasifyPaymentWS As New EasifyInboundPayments
Dim PaymentAccountsArray() As PaymentAccounts

Try
    PaymentAccountsArray = EasifyPaymentWS.GetPaymentAccounts("ChangeThisToYourPrivateKey")
    For Each PaymentAccount In PaymentAccountsArray
        Debug.Print(PaymentAccount.Description)
    Next
Catch ex As Exception
    ' See exception handling section at the end of this guide.
End Try
```

### PaymentAccounts Class Details

Property	Type	Description
Id	Integer	Unique Payment Account Id
Description	String	Account description
Active	Boolean	Indicates whether the account is active

## EasifyInboundPayments.GetPaymentMethods Method

Retrieves a list of payment method objects from Easify.

### Web Service URL

</Inbound/EasifyInboundPayments.asmx?op=GetPaymentMethods>

### Parameters

Name	Type	Description	Required
PK	String	Easify Private Key	Yes

### VB.Net Example

```
' Assume we have imported the InboundPayments web service
Dim EasifyPaymentWS As New EasifyInboundPayments
Dim PaymentMethodsArray() As PaymentMethods

Try
    PaymentMethodsArray = EasifyPaymentWS.GetPaymentMethods("ChangeThisToYourPrivateKey")
    For Each PaymentMethod In PaymentMethodsArray
        Debug.Print(PaymentMethod.Description)
    Next
Catch ex As Exception
    ' See exception handling section at the end of this guide.
End Try
```

### PaymentMethods Class Details

Property	Type	Description
Id	Integer	Unique Payment Method Id
Description	String	Payment method description
Active	Boolean	Indicates whether the Payment Method is active
ShowInPos	Boolean	Indicates whether the Payment Method should be shown in Easify Point Of Sale, not relevant to web based transactions.

## EasifyInboundPayments Exception Enumerations

### InsertPaymentExceptions

Value
ExtOrderNo_must_be_assigned_a_value
PaymentAccountId_must_be_assigned_a_value
PaymentMethodId_must_be_assigned_a_value
PaymentTypeId_must_be_assigned_a_value
PaymentDate_must_be_assigned_a_value
EASIFY_INVALID_PRIVATE_KEY
EASIFY_UNCAUGHT_EXCEPTION
EASU_LICENSE_IS_NOT_VALID_OR_EXPIRED

### GeneralExceptions

Value
EASIFY_INVALID_PRIVATE_KEY
EASIFY_UNCAUGHT_EXCEPTION
EASU_LICENSE_IS_NOT_VALID_OR_EXPIRED

## EasifyInboundProducts Web Service

The EasifyInboundProducts Web Service provides you with a means to retrieve information about products and related product information such as suppliers and manufacturers from Easify.

The following methods are supported by EasifyInboundProducts:

Methods	Description
GetCategories	Retrieve the list of product categories from Easify
GetSubCategories	Retrieve the list of product subcategories from Easify
GetSubCategoriesByCategory	Retrieve a list of product subcategories from Easify that come under a given category
GetManufacturers	Retrieve the list of manufacturers from Easify
GetSuppliers	Retrieve the list of suppliers from Easify
GetTaxRates	Retrieve the list of supported tax rates from Easify

EasifyInboundProducts surfaces the following classes:

Types
Categories
SubCategories
Manufacturers
Suppliers
TaxRates

EasifyInboundProducts provides the following enumerations:

Enums	Description
GeneralExceptions	Enumeration of general exceptions that may be thrown by any method

## EasifyInboundProducts.GetCategories Method

Returns an array of Easify Product Categories.

### Web Service URL

</Inbound/EasifyInboundProducts.asmx?op=GetCategories>

### Parameters

Name	Type	Description	Required
PK	String	Easify Private Key	Yes

### VB.Net Example

```
' Assume we have imported the InboundProducts web service
Dim EasifyProductWS As New EasifyInboundProducts
Dim CategoriesArray() As Categories

Try
    CategoriesArray = EasifyProductWS.GetCategories("ChangeThisToYourPrivateKey")
    For Each Category In CategoriesArray
        Debug.Print(Category.Description)
    Next
Catch ex As Exception
    ' See exception handling section at the end of this guide.
End Try
```

### Categories Class Details

Property	Type	Description
Id	Integer	Unique Product Category Id
Description	String	Product Category description

## EasifyInboundProducts.GetSubCategories Method

Returns an array of Easify Product SubCategories.

### Web Service URL

</Inbound/EasifyInboundProducts.asmx?op=GetSubCategories>

### Parameters

Name	Type	Description	Required
PK	String	Easify Private Key	Yes

### VB.Net Example

```
' Assume we have imported the InboundProducts web service
Dim EasifyProductWS As New EasifyInboundProducts
Dim SubCategoriesArray() As SubCategories

Try
    SubCategoriesArray = EasifyProductWS.GetSubCategories("ChangeThisToYourPrivateKey")
    For Each SubCategory In SubCategoriesArray
        Debug.Print(SubCategory.Description)
    Next
Catch ex As Exception
    ' See exception handling section at the end of this guide.
End Try
```

### SubCategories Class Details

Property	Type	Description
Id	Integer	Unique Product Sub Category Id
Description	String	Product subcategory description
CategoryId	Integer	Product subcategory's parent category id

## EasifyInboundProducts.GetSubCategoriesByCategory Method

Returns an array of Easify Product SubCategories for a given product category.

### Web Service URL

</Inbound/EasifyInboundProducts.asmx?op=GetSubCategoriesByCategory>

### Parameters

Name	Type	Description	Required
CategoryId	Integer	Easify Product Category to retrieve subcategories for	Yes
PK	String	Easify Private Key	Yes

### VB.Net Example

```
' Assume we have imported the InboundProducts web service
Dim EasifyProductWS As New EasifyInboundProducts
Dim SubCategoriesArray() As SubCategories

Try
    SubCategoriesArray = EasifyProductWS.GetSubCategoriesByCategory(1, _
                                                                    "ChangeThisToYourPrivateKey")
    For Each SubCategory In SubCategoriesArray
        Debug.Print(SubCategory.Description)
    Next
Catch ex As Exception
    ' See exception handling section at the end of this guide.
End Try
```

### SubCategories Class Details

Property	Type	Description
Id	Integer	Unique Product Sub Category Id
Description	String	Product subcategory description
CategoryId	Integer	Product subcategory's parent category id

## EasifyInboundProducts.GetManufacturers Method

Returns an array of Easify Product Manufacturers.

### Web Service URL

</Inbound/EasifyInboundProducts.asmx?op=GetManufacturers>

### Parameters

Name	Type	Description	Required
PK	String	Easify Private Key	Yes

### VB.Net Example

```
' Assume we have imported the InboundProducts web service
Dim EasifyProductWS As New EasifyInboundProducts
Dim ManufacturersArray() As Manufacturers

Try
    ManufacturersArray = EasifyProductWS.GetManufacturers("ChangeThisToYourPrivateKey")
    For Each Manufacturer In ManufacturersArray
        Debug.Print(Manufacturer.Description)
    Next
Catch ex As Exception
    ' See exception handling section at the end of this guide.
End Try
```

### Manufacturers Class details

Property	Type	Description
Id	Integer	Unique Manufacturer Id
Description	String	Manufacturer description

## EasifyInboundProducts.GetSuppliers Method

Returns an array of Easify Suppliers.

### Web Service URL

</Inbound/EasifyInboundProducts.asmx?op=GetSuppliers>

### Parameters

Name	Type	Description	Required
PK	String	Easify Private Key	Yes

### VB.Net Example

```
' Assume we have imported the InboundProducts web service
Dim EasifyProductWS As New EasifyInboundProducts
Dim SuppliersArray() As Suppliers

Try
    SuppliersArray = EasifyProductWS.GetSuppliers("ChangeThisToYourPrivateKey")
    For Each Supplier In SuppliersArray
        Debug.Print(Supplier.Description)
    Next
Catch ex As Exception
    ' See exception handling section at the end of this guide.
End Try
```

### Suppliers Class Details

Properties	Type	Description
Id	Integer	Unique Supplier Id
Description	String	Supplier description

## EasifyInboundProducts.GetTaxRates Method

Returns an array of Easify Tax Rates.

</Inbound/EasifyInboundProducts.aspx?op=GetTaxRates>

### Parameters

Name	Type	Description	Required
PK	String	Easify Private Key	Yes

### VB.Net Example

```
' Assume we have imported the InboundProducts web service
Dim EasifyProductWS As New EasifyInboundProducts
Dim TaxRatesArray() As TaxRates

Try
    TaxRatesArray = EasifyProductWS.GetTaxRates("ChangeThisToYourPrivateKey")
    For Each TaxRate In TaxRatesArray
        Debug.Print(TaxRate.Description)
    Next
Catch ex As Exception
    ' See exception handling section at the end of this guide.
End Try
```

### TaxRates Class Details

Property	Type	Description
Id	Integer	Unique Tax Rate Id
Description	String	Tax Rate description
TaxCode	String	Tax Code
TaxRate	Double	Tax Rate
IsDefault	Boolean	Indicates whether this is the default Tax Rate applied to Order Items

## EasifyInboundProducts Exception Enumerations

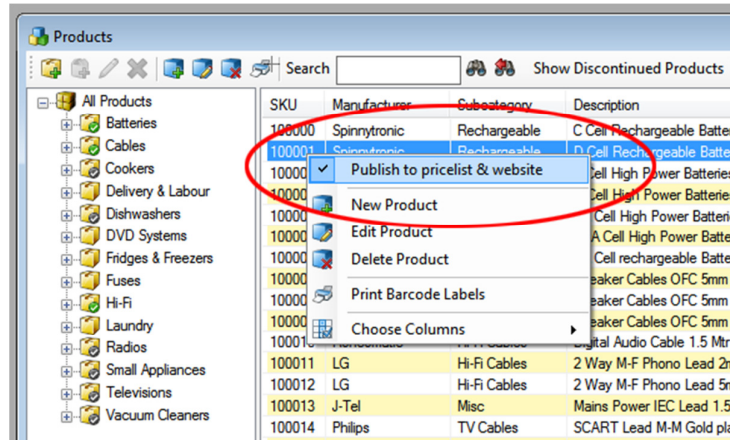
### GeneralExceptions

Value
EASIFY_INVALID_PRIVATE_KEY
EASIFY_UNCAUGHT_EXCEPTION
EASU_LICENSE_IS_NOT_VALID_OR_EXPIRED

## Handling outbound product updates

### Overview

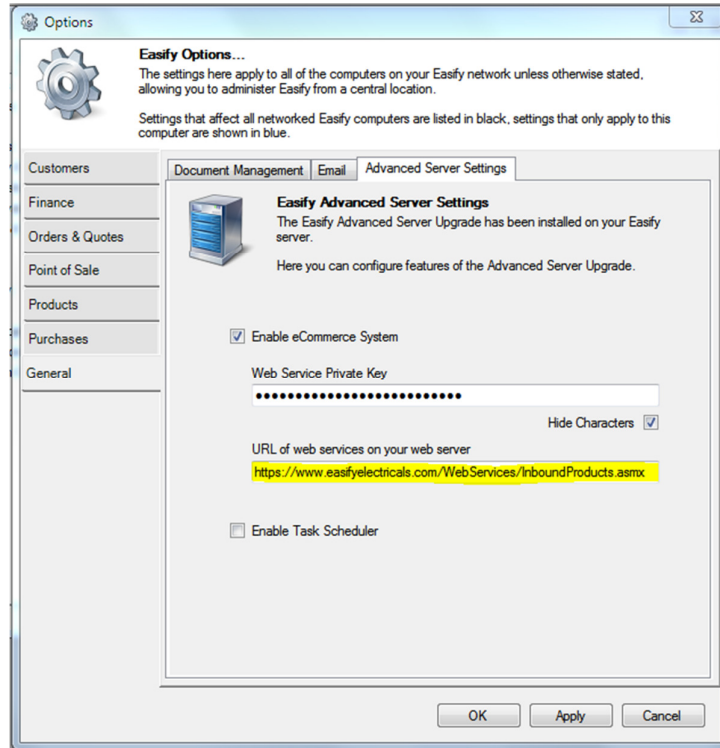
With eCommerce enabled and configured in Easify any product updates that you do for 'published' products in Easify can be propagated via EASU to your website. By 'published' we mean that the product has been marked as published in the Easify back end.



This includes adding a new product to Easify, updating product information and product image, change of product stock levels, change of product price etc...

Whenever a product is published or an already published product is modified, Easify will call a pre-defined web service on your web server, passing it the changed information.

You implement the web service yourself (see source code samples below) and configure Easify options to point to the location of the web service that you have implemented.



In the event that the internet connection between EASU and your website should go down, EASU queues up outgoing product updates to ensure that as soon as the link comes back up, any pending product updates will be uploaded.

The product update web service that you implement must surface the following web methods:

Web Method Name	Description
UploadProduct()	Accepts an entire product upload from Easify and inserts it into website database
UploadProductInfo()	Accepts product HTML description and product image from Easify and inserts it into website database / file system
UploadStockLevel()	Accepts a new stock level for a product from Easify and updates website stock level
DeleteProduct()	Called when a product is unpublished in Easify

## UploadProduct Method

Accepts an entire product upload from Easify and inserts it into your website database.

### Method Name

UploadProduct()

### Parameters

Name	Type	Description	Required
Product	ProductDetails	Class containing product details	Yes
PK	String	Easify Private Key	Yes

### ProductDetails Class Details

Property	Type	Description
SKU	Integer	Easify SKU of the product
OurStockCode	String	Your stock code for the product
ManufacturerStockCode	String	Manufacturers stock code
SupplierStockCode	String	Suppliers stock code
EANCode	String	Product EAN number
Description	String	Product description
CategoryId	Integer	Category Id number
SubcategoryId	Integer	Subcategory Id number
ManufacturerId	Integer	Manufacturer Id
CostPrice	Decimal	Cost price
Notes	String	Product notes (not publicly visible)
StockLevel	Integer	Stock level
Discontinued	Boolean	Set True if product is discontinued in Easify
MinimumStockLevel	Integer	Minimum stock level to trigger re-order warning
ReorderAmount	Integer	Amount to re-order when stock low
ReorderWhenLow	Boolean	Whether to re-order product when stock low
DateStockLevelLastUpdated	Date	Date stock level last updated
DatePriceLastChanged	Date	Date product price last changed
SupplierId	Integer	Supplier Id
RetailMargin	Double	Retail margin to apply to cost price
TradeMargin	Double	Trade margin to apply to cost price
TaxId	Integer	Tax Id
Published	Boolean	Set true if product published to web site
Allocatable	Boolean	Set true if this product can be allocated
Picture	String	Product image
HTMLDescription	String	HTML description of product

### Notes

This web method is called by Easify via EASU whenever a product is published in Easify, or if an already published product is modified.

Your web method should handle this method by extracting any relevant information and update your website product database.

This method is called both when a product is published and when it is modified, therefore your web method should handle both scenarios by either inserting a new product into your website database, or by updating an existing product record in your database.

## UploadProductInfo Method

Accepts product HTML description and product image from Easify and inserts it into your website database / file system.

### Method Name

UploadProductInfo()

### Parameters

Name	Type	Description	Required
SKU	ProductSKU	Easify SKU of the product	Yes
Picture	Byte()	Byte array containing a serialised copy of the product image from Easify	Yes
HTMLDescription	String	HTML description of the product from Easify product info	Yes
PK	String	Easify Private Key	Yes

### Notes

UploadProductInfo() is called whenever the product picture or HTML description is modified in Easify.

## UploadStockLevel Method

Accepts a new stock level for a product from Easify and updates website stock level.

### Method Name

UploadStockLevel()

### Parameters

Name	Type	Description	Required
SKU	ProductSKU	Easify SKU of the product	Yes
StockLevel	Integer	The new stock level for the product	Yes
PK	String	Easify Private Key	Yes

### Notes

UploadStockLevel() is called whenever a product stock level changes in Easify.

A stock level change can be triggered either by a user manually changing a stock level in the product details in Easify, by a product being sold via Easify or by new stock being checked in in Easify's purchasing system.

## DeleteProduct Method

Called when a product is unpublished in Easify.

### Method Name

DeleteProduct()

### Parameters

Name	Type	Description	Required
SKU	ProductSKU	Easify SKU of the product	Yes
PK	String	Easify Private Key	Yes

### Notes

DeleteProduct () is called whenever a product is unpublished or discontinued in Easify.

## Web Service example code

The following examples of web services demonstrate how you might implement your product update web service either in VB.Net or in PHP.

Your web service must implement all four web methods as described above. If you do not wish to implement the functionality of a web method, you should still implement it in your web service but do no processing within it. This will allow EASU to believe it has successfully processed its pending product updates and prevents a build-up of unprocessed messages in EASU.

How you choose to implement the internals of each web method is entirely up to you and will depend on the exact shopping cart you are programming for, as well as how your business model operates. In the examples below we have received each web method and processed it against a fictitious business logic class. In reality you would create your own code here to update products and stock levels on your web server.

## VB.Net Example Web Service

```
Imports System.Web.Services
Imports System.Web.Services.Protocols
Imports System.ComponentModel

' To allow this Web Service to be called from script, using ASP.NET AJAX,
' uncomment the following line.
' <System.Web.Script.Services.ScriptService()> _
<System.Web.Services.WebService(Namespace:="http://tempuri.org/")> _
<System.Web.Services.WebServiceBinding(ConformsTo:=WsiProfiles.BasicProfile1_1)> _
<ToolboxItem(False)> _
Public Class InboundProducts
    Inherits System.Web.Services.WebService

    Private PrivateKey As String = "ChangeThisToYourPrivateKey"

    Private Function IsPKValid(ByVal PK As String) As Boolean
        Return IIf(PK = PrivateKey, True, False)
    End Function

    <WebMethod()> _
    Public Sub UploadProduct(ByVal Product As ProductDetails, ByVal PK As String)
        Try
            ' check for authorised session
            If Not IsPKValid(PK) Then Return

            Dim db As New BusinessLogic ' your data handling object
            Dim prd As New ShoppingCartProduct ' your product object

            If Not db.IsExistingProduct(Product.SKU) Then
                ' parse Easify product into your desired format
                Dim DecimalPlaces As Integer = 2
                Dim Price As Decimal = Math.Round((Product.CostPrice / _
                    (100 - Product.RetailMargin) * _
                    100), DecimalPlaces)

                prd.EasifySKU = Product.SKU
                prd.Description = Product.Description
                prd.CategoryId = Product.CategoryId
                prd.Price = Price
                prd.StockLevel = Product.StockLevel
                prd.Active = Product.Discontinued
                prd.RetailMargin = Product.RetailMargin
                prd.TaxId = Product.TaxId
                prd.ProductImage = Product.Picture
                prd.LongDescription = Product.HTMLDescription
                ' insert new product into your database
                db.InsertProductIntoDatabase(prd)
            Else
                ' update your existing product
                db.UpdateProductInDatabase(Product)
            End If
        End Try
    End Sub
End Class
```

```

    Catch ex As Exception
        EventLog.WriteEntry("Application", _
            "eCommerce.InboundProducts.UploadProduct() - " & ex.Message, _
            EventLogEntryType.Information)
    End Try
End Sub

<WebMethod()> _
Public Sub UploadProductInfo(ByVal ProductSKU As Integer, _
    ByVal Picture As Byte(), _
    ByVal HTMLDescription As String, _
    ByVal PK As String)

    Try
        ' check for authorised session
        If Not IsPKValid(PK) Then Return

        Dim db As New BusinessLogic ' your data handling object

        If db.IsExistingProduct(ProductSKU) Then
            ' update your existing product
            db.UpdateProductInfoInDatabase(ProductSKU, Picture, HTMLDescription)
        Else
            ' product doesn't exist, log error message
            Throw New Exception("Product doesn't exist")
        End If

    Catch ex As Exception
        EventLog.WriteEntry("Application", _
            "eCommerce.InboundProducts.UploadProductInfo() - " & _
            ex.Message, EventLogEntryType.Information)
    End Try
End Sub

<WebMethod()> _
Public Sub UploadStockLevel(ByVal ProductSKU As Integer, _
    ByVal StockLevel As Integer, ByVal PK As String)

    Try
        ' check for authorised session
        If Not IsPKValid(PK) Then Return

        Dim db As New BusinessLogic ' your data handling object

        If db.IsExistingProduct(ProductSKU) Then
            ' update your existing product
            db.UpdateProductStockLevelInDatabase(ProductSKU, StockLevel)
        Else
            ' product doesn't exist, log error message
            Throw New Exception("Product doesn't exist")
        End If

    Catch ex As Exception
        EventLog.WriteEntry("Application", _
            "eCommerce.InboundProducts.UploadStockLevel() - " & _
            ex.Message, EventLogEntryType.Information)
    End Try
End Sub

<WebMethod()> _
Public Sub DeleteProduct(ByVal ProductSKU As Integer, ByVal PK As String)
    Try
        ' check for authorised session
        If Not IsPKValid(PK) Then Return

        Dim db As New BusinessLogic ' your data handling object

        If db.IsExistingProduct(ProductSKU) Then
            ' delete your existing product
            db.DeleteProductFromDatabase(ProductSKU)
        Else
            ' product doesn't exist, log error message
            Throw New Exception("Product doesn't exist")
        End If

    Catch ex As Exception
        EventLog.WriteEntry("Application", _
            "eCommerce.InboundProducts.DeleteProduct() - " & ex.Message, _

```

```

        EventLogEntryType.Information)
    End Try
End Sub
End Class

```

## PHP Example Web service

```

<?php
include('BusinessLogic.php');

class UploadProduct {
    public $Product; // ProductDetails
    public $PK; // string
}

class ProductDetails {
    public $SKU; // int
    public $OurStockCode; // string
    public $ManufacturerStockCode; // string
    public $SupplierStockCode; // string
    public $EANCode; // string
    public $Description; // string
    public $CategoryId; // int
    public $SubcategoryId; // int
    public $ManufacturerId; // int
    public $CostPrice; // decimal
    public $Notes; // string
    public $StockLevel; // int
    public $Discontinued; // boolean
    public $MinimumStockLevel; // int
    public $ReorderAmount; // int
    public $ReorderWhenLow; // boolean
    public $DateStockLevelLastUpdated; // dateTime
    public $DatePriceLastChanged; // dateTime
    public $SupplierId; // int
    public $RetailMargin; // double
    public $TradeMargin; // double
    public $TaxId; // int
    public $Published; // boolean
    public $Allocatable; // boolean
    public $Picture; // string
    public $HTMLDescription; // string
}

function IsPKValid($PK)
{
    $PrivateKey = "ChangeThisToYourPrivateKey";
    return $PK == $PrivateKey ? true : false;
}

function UploadProduct($params)
{
    $valueArray = get_object_vars($params);
    $Product = $valueArray["Product"];
    $PK = $valueArray["PK"];

    if (!IsPKValid($PK)) return false;

    $TestData = "~ UploadProduct ~\n\n" .
        "Product.SKU: " . $Product->SKU . "\n" .
        "Product.OurStockCode: " . $Product->OurStockCode . "\n" .
        "Product.ManufacturerStockCode: " . $Product->ManufacturerStockCode . "\n" .
        "Product.SupplierStockCode: " . $Product->SupplierStockCode . "\n" .
        "Product.EANCode: " . $Product->EANCode . "\n" .
        "Product.Description: " . $Product->Description . "\n" .
        "Product.CategoryId: " . $Product->CategoryId . "\n" .
        "Product.SubcategoryId: " . $Product->SubcategoryId . "\n" .
        "Product.ManufacturerId: " . $Product->ManufacturerId . "\n" .
        "Product.CostPrice: " . $Product->CostPrice . "\n" .
        "Product.Notes: " . $Product->Notes . "\n" .
        "Product.StockLevel: " . $Product->StockLevel . "\n" .
        "Product.Discontinued: " . $Product->Discontinued . "\n" .
        "Product.MinimumStockLevel: " . $Product->MinimumStockLevel . "\n" .
        "Product.ReorderAmount: " . $Product->ReorderAmount . "\n" .

```

```

"Product.ReorderWhenLow: " . $Product->ReorderWhenLow . "\n" .
"Product.DateStockLevelLastUpdated: " . $Product->DateStockLevelLastUpdated . "\n" .
"Product.DatePriceLastChanged: " . $Product->DatePriceLastChanged . "\n" .
"Product.SupplierId: " . $Product->SupplierId . "\n" .
"Product.RetailMargin: " . $Product->RetailMargin . "\n" .
"Product.TradeMargin: " . $Product->TradeMargin . "\n" .
"Product.TaxId: " . $Product->TaxId . "\n" .
"Product.Published: " . $Product->Published . "\n" .
"Product.Allocatable: " . $Product->Allocatable . "\n" .
"Product.Picture: " . $Product->Picture . "\n" .
"Product.HTMLDescription: " . $Product->HTMLDescription . "\n" .
"PK: " . $PK . "\n";

$db = new BusinessLogic(); // your data handling object
$prd = new ShoppingCartProduct(); // your product object
$db->WriteLog($TestData);

if (!$db->IsExistingProduct($Product->SKU))
{
    // parse Easify product into your desired format
    $DecimalPlaces = 2;
    $Price = round(($Product->CostPrice / (100 - $Product->RetailMargin) * 100),
DecimalPlaces);
    $prd->EasifySKU = $Product->SKU;
    $prd->Description = $Product->Description;
    $prd->CategoryId = $Product->CategoryId;
    $prd->Price = $Product->CostPrice;
    $prd->StockLevel = $Product->StockLevel;
    $prd->Active = $Product->Discontinued;
    $prd->RetailMargin = $Product->RetailMargin;
    $prd->TaxId = $Product->TaxId;
    $prd->ProductImage = $Product->Picture;
    $prd->LongDescription = $Product->HTMLDescription;
    // insert new product into your database
    $db->InsertProductIntoDatabase($prd);
} else {
    $db->UpdateProductInDatabase($Product);
}
}

function UploadProductInfo($params)
{
    $valueArray = get_object_vars($params);
    $ProductSKU = $valueArray["ProductSKU"];
    $Picture = $valueArray["Picture"];
    $HTMLDescription = $valueArray["HTMLDescription"];
    $PK = $valueArray["PK"];

    if (!IsPKValid($PK)) return false;

    $TestData = "~ UploadProductInfo ~\n\n" .
    "ProductSKU: " . $ProductSKU . "\n" .
    "Picture: " . $Picture . "\n" .
    "HTMLDescription: " . $HTMLDescription . "\n" .
    "PK: " . $PK . "\n";

    $db = new BusinessLogic(); // your data handling object
    $db->WriteLog($TestData);

    if ($db->IsExistingProduct($ProductSKU))
    {
        // update existing product
        $db->UpdateProductInfoInDatabase($ProductSKU, $Picture, $HTMLDescription);
    } else {
        // product doesn't exist, log error message
        $db->WriteLog("Product doesn't exist");
    }
}

function UploadStockLevel($params)
{
    $valueArray = get_object_vars($params);
    $ProductSKU = $valueArray["ProductSKU"];
    $StockLevel = $valueArray["StockLevel"];
    $PK = $valueArray["PK"];

    if (!IsPKValid($PK)) return false;

```

```

$TestData = "~ UploadStockLevel ~\n\n" .
"ProductSKU: " . $ProductSKU . "\n" .
"StockLevel: " . $StockLevel . "\n" .
"PK: " . $PK . "\n";

$db = new BusinessLogic(); // your data handling object
$db->WriteLog($TestData);

if ($db->IsExistingProduct($ProductSKU))
{
    // update existing product
    $db->UpdateProductStockLevelInDatabase($ProductSKU, $StockLevel);
} else {
    // product doesn't exist, log error message
    $db->WriteLog("Product doesn't exist");
}
}

function DeleteProduct($params)
{
    $valueArray = get_object_vars($params);
    $ProductSKU = $valueArray["ProductSKU"];
    $PK = $valueArray["PK"];

    if (!IsPKValid($PK)) return false;

    $TestData = "~ DeleteProduct ~\n\n" .
"ProductSKU: " . $ProductSKU . "\n" .
"PK: " . $PK . "\n";

    $db = new BusinessLogic(); // your data handling object
    $db->WriteLog($TestData);

    if ($db->IsExistingProduct($ProductSKU))
    {
        // update existing product
        $db->DeleteProductFromDatabase($ProductSKU);
    } else {
        // product doesn't exist, log error message
        $db->WriteLog("Product doesn't exist");
    }
}

ini_set("soap.wsdl_cache_enabled", "0");

$classmap = array(
    "UploadProduct" => "UploadProduct",
    "ProductDetails" => "ProductDetails"
);
$server = new SoapServer("InboundProducts.wsdl", array("classmap" => $classmap));
$server->addFunction("UploadProduct");
$server->addFunction("UploadProductInfo");
$server->addFunction("UploadStockLevel");
$server->addFunction("DeleteProduct");
$server->handle();
?>

```

## Exception handling

Any errors that are encountered by an EASU web service method call are raised as SoapExceptions so that you can handle them appropriately within your code.

The use of structured exception handling means all of your error handling can be located in a single place where you can take appropriate action based on the nature of the exception or error that was encountered.

## Custom exception properties

All errors raised by the web service are raised using the SoapException exception class. When an exception is raised three custom error properties are added to the Detail object of the SoapException class which are described below.

Custom property	Description
Errorcode	The Easify error code
errormessage	The Easify error message that can be converted to an enum
Exceptionmessage	The original exception message raised by the web service. This will only contain a value when the errormessage enum value is EASIFY_UNCAUGHT_EXCEPTION

## Converting the custom errormessage property to an enum

To convert the errormessage property of the exception Detail to an enum you can use the function **GetExceptionMessageAsEnum** that is listed below. This function does not exist in any EASU classes, and you should implement it yourself by copying and pasting it into a suitable location within your own code.

You pass this routine the SoapException and the type of the enum to be created and it returns an enum which you can then make comparisons against. Each web service call has an enum associated with its exception handling so your call should use the enum appropriate to the web service call.

## VB.Net Example

```
Private Function GetExceptionMessageAsEnum(ByVal Ex As SoapException, _
                                           ByVal EnumType As System.Type) As System.Enum
    If Ex.Detail.Item("errormessage").IsEmpty = True Then
        Return Nothing
    ElseIf System.Enum.IsDefined(EnumType, _
                                   Ex.Detail.Item("errormessage").InnerText) = False Then
        Return Nothing
    Else
        Dim em As System.Enum = System.Enum.Parse(EnumType, _
                                                    Ex.Detail.Item("errormessage").InnerText)
        Return em
    End If
End Function
```

## Handling exceptions thrown by the EasifyInboundCustomers.InsertCustomer method

In the following example we demonstrate how you would call the **EasifyInboundCustomers.InsertCustomer** method to insert a new customer record, along with exception handling code to trap any thrown exceptions, and log them to the debug output stream.

## VB.Net Example

```
' Assume we have imported the InboundCustomers web service
Dim EasifyInboundCustomersWS As New EasifyInboundCustomers
```

```

Dim Customer As New CustomerDetails

Try
    Customer.ExtCustomerId = "100000"
    Customer.CompanyName = "Some Test Co"
    Customer.Title = "Mr"
    Customer.FirstName = "Test"
    Customer.Surname = "Bloke"
    Customer.JobTitle = "Test Manager"
    Customer.Address1 = "23 Test Street"
    Customer.Town = "Sometown"
    Customer.County = "Countyshire"
    Customer.Postcode = "AA77 88JJ"
    Customer.Country = "United Kingdom"

    Customer.WorkTel = "01305 303040"
    Customer.Mobile = "N/A"
    Customer.Email = "info@sometestco.co.uk"
    Customer.Fax = "N/A"

    Customer.SubscribeToNewsletter = False
    Customer.TradeAccount = False
    Customer.CreditLimit = 0
    Customer.CustomerTypeId = 1
    Customer.PaymentTermsId = 1
    Customer.RelationshipId = 1

    ' Insert the customer into Easify
    EasifyInboundCustomersWS.InsertCustomer(Customer, "ChangeThisToYourPrivateKey")
Catch ex As SoapException
    ' Assume we have imported System.Web.Services.Protocols to support SoapException

    ' Create a variable that will hold an error message to be displayed to the user
    Dim errText As String = String.Empty

    ' Create an InsertCustomerExceptions enumeration from the the error message
    ' returned by the exception
    Dim em As InsertCustomerExceptions
    em = GetExceptionMessageAsEnum(ex, GetType(GeneralExceptions))

    ' Check the enum was created
    If em <> Nothing Then
        ' Compare the error message enum against the InsertCustomerExceptions enums and
        ' set an error message to display to the user accordingly. Here we're just
        ' creating an error message but you might want to take evasive action
        ' depending on the nature of the error.
        Select Case em
            Case wsCustomers.InsertCustomerExceptions.CustomerTypeId_must_be_assigned_a_value
                errText = "Customer type was not assigned a value."
            Case wsCustomers.InsertCustomerExceptions.ExtCustomerId_must_be_assigned_a_value
                errText = "ExtCustomerId was not assigned a value."
            Case wsCustomers.InsertCustomerExceptions.PaymentTermsId_must_be_assigned_a_value
                errText = "PaymentTermsId was not assigned a value."
            Case wsCustomers.InsertCustomerExceptions.RelationshipId_must_be_assigned_a_value
                errText = "RelationshipId was not assigned a value."
                ' An uncaught exception was raised so the original raw exception message
                ' is contained in the custom property exceptionmessage
            Case wsCustomers.InsertCustomerExceptions.EASIFY_UNCAUGHT_EXCEPTION
                errText = "Uncaught error occurred with the message: " & vbCrLf & vbCrLf & _
                    ex.Detail.Item("exceptionmessage").InnerText
        End Select
    Else
        errText = "An uncaught error occurred with the message: " & _
            vbCrLf & vbCrLf & ex.Message
    End If

    Debug.Print("Customer Insert Failed: " & errText)
End Try

```

In the above example an error message is logged to the debug output stream, but for production quality code you would probably want to take different actions depending on the exception that was thrown.