Sage 100 2023

Web Services Guide

February 2023

This is a publication of Sage Software, Inc.

Last updated: February 27, 2023

Contents

Introduction to eBusiness Web Services 1

Getting Started 2

- Installing eBusiness Web Services 3
- Uninstalling eBusiness Web Services 3

Programming with eBusiness Web Services 4

- Web Services Description Language (WSDL) 4
- Using SOAP Efficiently with eBusiness Web Services 5
- Date and Time Values 7
- Credit Card Processing 8
- UDFs and Development Partner Fields 9
- Sage eBusiness Web Services Configuration Utility 10
- Error Handling 12
- Quick Start 15

Administration 27

- Security and Encryption 27
- Error Logging 29
- Object Cache 29
- Permissions 30

APIs 31

- GetContractInformation 31
- GetDiagnosticInformation 31
- GetSalesOrder 32
- GetNextSalesOrderNo 33
- GetSalesOrderTemplate 33
- PreviewSalesOrder 35
- CreateSalesOrder 36
- UpdateSalesOrder 37
- DeleteSalesOrder 38

GetCustomer ... 39
GetNextCustomerNo ... 40
CreateCustomer ... 40
UpdateCustomer ... 41
DeleteCustomer ... 42
GetCustomerContact ... 43
CreateCustomerContact ... 44
UpdateCustomerContact ... 45
DeleteCustomerContact ... 46
AddCreditCardToVault ... 47
PreAuthorizeCreditCard ... 48

Troubleshooting ... 50
Downloading WSDL - Wrong Host Name ... 50
Downloading WSDL - XSD not Available ... 52
Error 404 with Internet Information Services (IIS) version 6 ... 52
Could Not Load Type "System.ServiceModel.Activation.HttpModule" in IIS 7 ... 53
General Errors ... 53

Index ... 55

Introduction to eBusiness Web Services

eBusiness Web Services is a SOAP based programming interface that provides read, create, update, and delete functionality for the Sage 100 sales order, customer, and customer contact entities. eBusiness Web Services is built using Windows Communication Foundation and runs in Internet Information Services (IIS).

The services can be consumed by any development environment that supports SOAP, WSDL, and HTTPS. There is no specific language or operating system requirement for client applications written to consume eBusiness Web Services.

Some popular environments that can consume the services are:

- Visual Studio .NET 2008 or later using C# or VB.NET
- NetBeans IDE 6.5 or later using Java with JAX-WS
- PHP using the built-in SoapClient class

eBusiness Web Services is ideal for applications that run across the Internet, such as ecommerce solutions. The services are used with the HTTPS protocol. This provides a secure and reliable method of communication.

> **Note:** This manual is written for the experienced Microsoft Internet Information Services (IIS) Administrator. Knowledge of this area is required.

Getting Started

This chapter covers the installation process for eBusiness Web Services. For installation information regarding the Sage 100 software, refer to your applicable *Installation and System Administrator's Guide*. For a complete list of system requirements, refer to the Supported Platform Matrix.

The eBusiness Web Services installation process is more complicated than standard modules. Every effort has been made to make this process as simple as possible; however, before you start, you must know some important information about your network configuration.

eBusiness Web Services must be installed on the same machine where your Sage 100 installation resides. Internet Information Services (IIS) must be installed before installing eBusiness Web Services.

The installation process consists of the following

1. Install a server certificate for Internet Information Services. For more information, see Configuring Server Certificates in IIS 7.
2. Ensure that the .NET Framework 4.0 is installed.
3. Enable required extensions and features.
   - IIS 6: Allow the ASP.NET v4.0.30319 Web Service Extension through Internet Information Services Manager
   - IIS 7: No additional steps are required
   - IIS 8: Install the .NET Framework 4.5 WCF Services HTTP Activation feature from Add Roles and Features
4. Install eBusiness Web Services.
5. Enable Web Services for the system.
6. Enable Web Services for the company.
7. Enable Web Services for the user.

Installing eBusiness Web Services

After installing Sage 100 with an eBusiness Web Services enabled activation key, you are ready to install eBusiness Web Services.

To begin the installation process

1. Click the applicable Sage 100 product from the installation Autorun screen, click Productivity Applications, and then click Install eBusiness Web Services to begin the installation.
2. Click Next to proceed with the installation. A Windows user account named eBusinessWebServices is required to run the eBusiness Web Services service.

Uninstalling eBusiness Web Services

You must first uninstall eBusiness Web Services prior to uninstalling Sage 100.

Programming with eBusiness Web Services

eBusiness Web Services provides a platform independent programming interface. Any programming language in any development environment and operating system that has support for SOAP, WSDL, and HTTPS can be used to call eBusiness Web Services.

How a particular environment invokes the web services operations from a programming perspective, is not defined by web services; therefore, we cannot document how to call eBusiness Web Services from all possible development environments. For examples of calling eBusiness Web Services from some popular environments, see "Quick Start" (page 15).

All eBusiness Web Services operations are stateless and atomic. An operation either completes successfully or does not complete at all. For example, if a network connection is dropped during the CreateSalesOrder operation, and a sales order has been partially transmitted, the operation will not create a partial sales order; it will not create a sales order at all.

eBusiness Web Services operations must be called from a SOAP client. How the client is created depends on the development platform, but all clients must point to the URL where eBusiness Web Services resides.

eBusiness Web Services operations utilize SOAP faults when they error. The development environment for the client application determines how to capture and handle the faults. Most environments require that SOAP faults be handled by catching exceptions through the use of try/catch blocks.

For instructions on creating a SOAP client and for examples of how to use try/catch blocks for some popular development environments, see "Quick Start" (page 15).

Web Services Description Language (WSDL)

eBusiness Web Services uses the WSDL standard to describe the services it provides. The use of WSDL allows a programmer to point to the URL where eBusiness Web Services resides and begin programming against the web services. The programmer

does not need to concern themselves with forming proper SOAP/XML requests when using a development platform capable of consuming WSDL.

eBusiness Web Services provides two sets of WSDL:

- Full: Targeted for Microsoft Visual Studio Service Reference (sometimes referred to as a WCF client).
- Basic: Targeted for all other environments, including Microsoft Visual Studio's Web Reference, Java's JAX-WS client, and PHP's integrated SoapClient.

The WSDL that a client obtains is determined by the entry point of the URL. eBusiness Web Services provides two default entry points:

- MasService.svc - Full WSDL
- MasBasicWsdl.svc - Basic WSDL

Administrators can define other entry points by using the advanced options in the Sage 100 eBusiness Web Services configuration utility.

Choosing the appropriate WSDL to consume will help ensure that the client application generates efficient SOAP requests and will help reduce the complexity of any generated proxy classes.

For example, a Visual Studio developer can point a Service Reference to https://server/ebusinesswebservices/masservice.svc?wsdl while a Java developer can point JAX-WS to https://server/ebusinesswebservices/masbasicwsdl.svc?wsdl.

Using SOAP Efficiently with eBusiness Web Services

eBusiness Web Services uses the W3C standard SOAP protocol to communicate between the client and server.

It is the use of SOAP that helps ensure that requests can traverse the Internet and that operations can be called from numerous platforms such as PCs, PDAs, and SmartPhones.

SOAP is an XML based protocol and; therefore can be quite bandwidth intensive if not used with care. eBusiness Web Services can be configured and programmed to reduce the amount of bandwidth used.

For example, as part of the response from a call to the GetCustomer operation, the following snippet of XML may be sent:

```
<Customer xmlns="http://mas90.sage.com/ws"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
<OtherFields />
<ARDivisionNo>01</ARDivisionNo>
<AddressLine1>2131 N. 14th Street</AddressLine1>
<AddressLine2>Suite 100</AddressLine2>
<AddressLine3 />
<AgingCategory1>0</AgingCategory1>
<AgingCategory2>0</AgingCategory2>
<AgingCategory3>0</AgingCategory3>
<AgingCategory4>0</AgingCategory4>
<AvgDaysOverDue>0</AvgDaysOverDue>
```

eBusiness Web Services can be configured to not emit XML for empty strings and zero values in server responses. When configured with these options, the above snippet would appear as follows:

```
<Customer xmlns="http://mas90.sage.com/ws"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
<OtherFields />
<ARDivisionNo>01</ARDivisionNo>
<AddressLine1>2131 N. 14th Street</AddressLine1>
<AddressLine2>Suite 100</AddressLine2>
```

As the example demonstrates, bandwidth is significantly reduced when the data returned contains many empty strings and zero values. A side effect of not emitting empty strings and zero values is that these values will appear as null values at the client. The client application must be able to process the null values returned to it.

eBusiness Web Services also allows specific fields to be blocked for read operations. When a field is blocked, XML for the field is never emitted. Blocking fields will generally yield the most bandwidth reduction, but care must be taken not to block fields that either the client application or eBusiness Web Services require to function properly. The eBusiness Web Services reference section can be used to determine the fields required for a particular operation. For more information, see "APIs" (page 31).

eBusiness Web Services has no control over XML generation at the client. Reducing bandwidth for client requests is the sole responsibility of the client application. Most development platforms will not emit XML for a Sage 100 entity field if the field value is set to null. Setting unneeded fields to null at the client is generally the most effective way of reducing bandwidth in client requests.

Date and Time Values

Sage 100 systems natively store dates and times in separate fields. eBusiness Web Services converts these separate fields into one field that returns the SOAP standard DateTime type.

For example, native Sage 100 Standard sales orders contain the following two fields in the payment object:

- AuthorizationDate$
- AuthorizationTime$

eBusiness Web Services converts these two fields into a single field called: AuthorizationDateTime

Some native data in Sage 100 systems have only a date component. For example, the sales order OrderDate$ field has no corresponding time field. Sage 100 will still return these fields as a DateTime type, but will distinguish date-only fields from date-time fields by omitting "Time" from the field name. In the case of OrderDate$, eBusiness Web Services exposes this field as OrderDate rather than OrderDateTime.

When a date and time is blank in Sage 100, null will be returned by eBusiness Web Services when the date and time are read through a Get operation.

Update and Create operations treat null values as "not set"; eBusiness Web Services performs no operation at all for a field when it is null. Setting a DateTime value to null during an Update operation will not set the Sage 100 date and time to blank. Instead, a special NullDateTime value must be used. This value can be obtained by calling the GetContractInformation operation. This value is constant and only needs to be obtained once by the client application.

Credit Card Processing

Credit card processing has changed significantly from previous version of eBusiness Web Services. It no longer allows eBusiness Web Services to enter a credit card number and CVV code directly into the SalesOrder contract because it no longer contains these fields. Instead, the SalesOrder contract contains a CreditCardGUID field that corresponds to a credit card number stored in the Sage Exchange Vault. In addition, the Payments collection of the SalesOrder contract contains CreditCardTransactionID and CreditCardAuthorizationNo fields that relate to a pre-authorization performed on a credit card.

The AddCreditCardToVault operation adds a credit card number and expiration date to the Sage Exchange Vault. This method will return a credit card GUID that should be stored by the calling application for future use. For more information, see "AddCreditCardToVault" (page 47).

A PreAuthorizeCreditCard operation explicitly pre-authorizes a credit card by the calling application. This call must be used to process a credit card using a CVV code. The operation will return a PreAuthorizationResult contract that contains the fields CreditCardTransactionID and CreditCardAuthorizationNo. The values in these fields, along with the credit card GUID obtained from AddCreditCardToVault , may be set in the Payments collection of the SalesOrder contract submitted to the CreateSalesOrder or UpdateSalesOrder operation to attach the pre-authorization to the sales order for an eventual completion of payment in Sage 100. For more information, see "PreAuthorizeCreditCard" (page 48).

As a security feature, credit card numbers and CVV codes are never returned by eBusiness Web Services.

UDFs and Development Partner Fields

UDFs and Development Partner fields are not exposed by default. UDFs can be automatically exposed by setting the AutoExposeUDFs option to Yes in eBusiness Web Services - Advanced Settings. UDFs and Development Partner fields can also be selectively exposed by adding them to the eBusiness Web Services - Advanced Settings Additional Fields tab. Development Partner fields cannot be automatically exposed; they must always be defined in the Additional Fields tab.

Both UDFs and Development Partner fields are exposed in the OtherFields collection of an entity. OtherFields contains a collection of Field objects. The Field object contains two fields:

- MasFieldName: The name of the UDF or Development Partner Field (for example, "UDF_NICKNAME$")
- Value: The value of the UDF. Value is always a string type, even when the native type for the field in the Sage 100 system is numeric.

To read a field, the OtherFields collection must be iterated through and MasFieldName compared against the desired field name before reading the Value field. Do not to assume that a particular field will always be at the same index position in the collection.

To change a field value for an Update operation, the OtherFields collection must be iterated through and MasFieldName must be compared against the appropriate field name before setting the field's Value field. The appropriate field may not be found if it has been explicitly blocked for read access in the advanced configuration settings. If this is the case, and write access is allowed for the field, a new Field object must be added to the OtherFields collection with MasFieldName and Value set to their appropriate values.

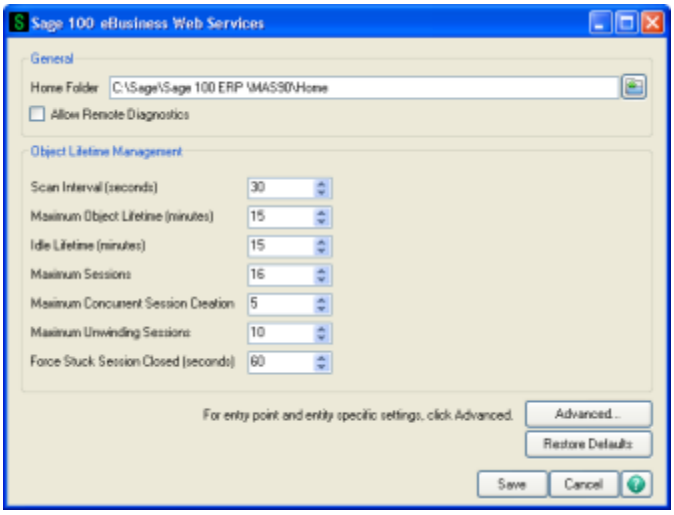
To set a field value during a Create operation using an entity instantiated at the client, a new Field object must be added to the OtherFields collection with MasFieldName and Value set to their appropriate values. The OtherFields collection will also need to be instantiated using the appropriate collection type for the platform.

For example, to set UDF_NICKNAME$ before calling CreateSalesOrder, the following C# code can be used with a Microsoft Service Reference:

```
masws.SalesOrder salesOrder = new masws.SalesOrder();
salesOrder.OtherFields = new BindingList<masws.Field>();
masws.Field udfField = new masws.Field();
udfField.MasFieldName = "UDF_NICKNAME$";
udfField.Value = "JOEY";
salesOrder.OtherFields.Add(udfField);
```

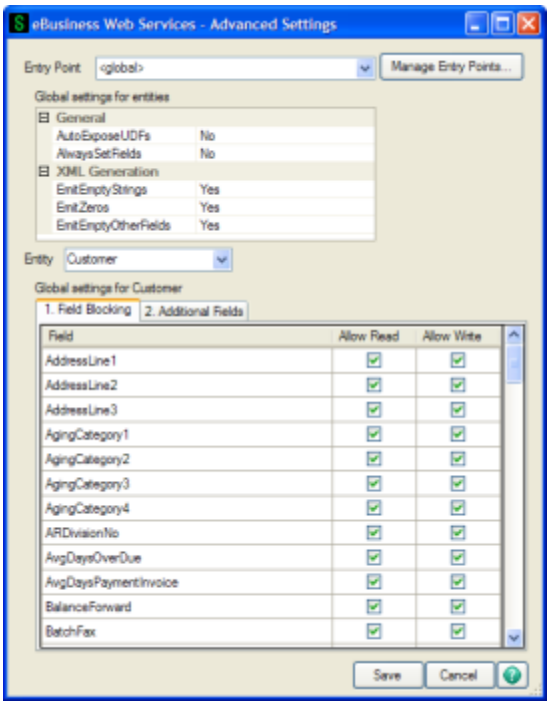
Sage eBusiness Web Services Configuration Utility

Use the Sage 100 eBusiness Web Services configuration utility to further define your settings for eBusiness Web Services.



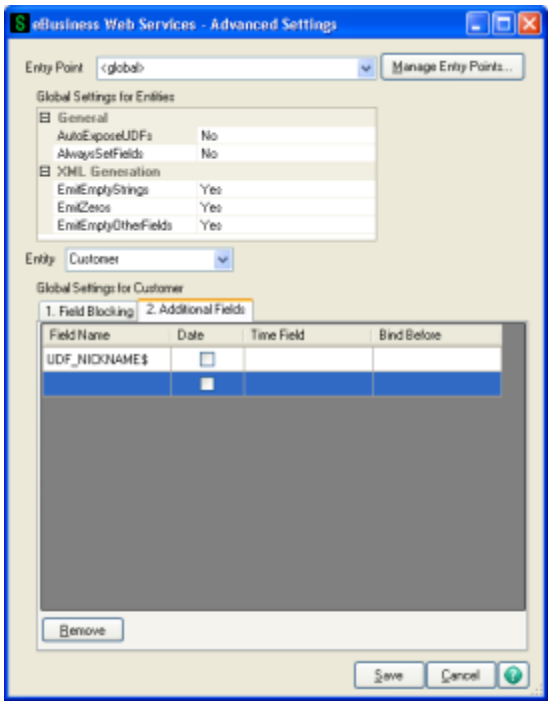
Click Advanced on the Sage 100 eBusiness Web Services window to access eBusiness Web Services - Advanced Settings. Use the advanced settings in this window to modify settings for specific entry points and entities.

Fields can be blocked by clearing the Allow Read or Allow Write check boxes.



Fields can be exposed by explicitly defining them on the Additional Fields tab. When fields are explicitly defined, the fields will appear at the bottom of the grid on the Field Blocking tab where they can also be explicitly blocked.

Any field that exists in Sage 100 for an entity can be defined on the Additional Fields tab. The field does not have to be a UDF.

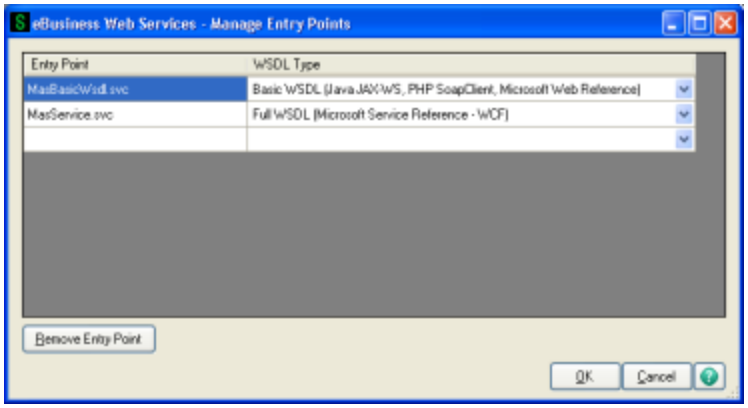


Manage Entry Points

Entry points can be created and removed from eBusiness Web Services - Manage Entry Points.

Entry points are typically used to hold application specific settings. For example, two clients may be used with eBusiness Web Services. One client can allow the XML for zero values and empty strings to be omitted, while the other client can require the XML to be emitted. Both clients can be served by the same installation of eBusiness Web Services by defining two entry points with different settings for EmitEmptyStrings and EmitZeros.

An entry point must be suffixed with .svc and must have a WSDL type specified.



Error Handling

eBusiness Web Services uses SOAP faults to return errors. SOAP faults contain a faultcode element that will be set to an error code and a faultstring that will be set to a descriptive error message. In addition, SOAP faults generated by the eBusiness Web Services component will include a MasFault element in the detail element of the SOAP fault.

> **Note:** eBusiness Web Services does not report warnings from the Sage 100 system.

The MasFault element contains the following items:

- ClassName: The name of the class that generated the error. This item can be blank.
- MemberName: The name of the method or property that generated the error. This item can be blank.
- ErrorMessage: The error message. This item is always set.

  The faultstring element will usually contain a more descriptive message than ErrorMessage as faultstring contains additional detail from the eBusiness Web Services component, while ErrorMessage is typically an error message directly from the Sage 100 system.
- ErrorCode: The error code. This item is always set. Both ErrorCode and faultcode will contain the same error code.

Most languages handle SOAP faults through the use of try/catch blocks.

For examples of how to catch eBusiness Web Services SOAP faults for some popular development environments, "Quick Start" (page 15).

eBusiness Web Services-Specific Errors

The following error codes are specific to eBusiness Web Services:

WS_AlreadyExists: "Record already exists."

This error occurs when a create method is called and the record being created already exists. This error can generally be ignored if it is received while retrying a timed out create request as it indicates that the previous call was successful.

WS_EventLog: "Could not write to Windows event log."

All eBusiness Web Services errors are written to the server's Windows Event Log. This error occurs when eBusiness Web Services is unable to write to the Windows Event Log. This is an abnormal condition that may be caused by a permissions or installation issue.

WS_Exception: "An exception occurred."

This error occurs when an unexpected exception occurs while executing an eBusiness Web Services method. The exception detail is not returned by eBusiness Web Services for security reasons. The exception detail can be viewed in the Windows Event Log on the server hosting eBusiness Web Services.

WS_FieldMissing: "The {Business Object} field {Field Name} does not exist in this record."

This error occurs when the record contained in the Sage 100 system does not contain the same schema as defined in eBusiness Web Services. This error will occur when an invalid additional field is defined using the Sage 100 eBusiness Web Services configuration utility. The error may also occur when data is corrupted in the Sage 100 system.

WS_FieldNotAllowed: "Field does not exist or is blocked."

This error occurs when a blocked field is written to or when a field does not exist.

WS_HaveSession: "Session already exists."

This is an eBusiness Web Services internal error and represents an abnormal condition. Report the error to Sage 100 Customer Support.

WS_NullValue: "Value is null."

This error occurs when a required value, such as a key value, is not set.

WS_Unknown: "Unknown error."

This error occurs when eBusiness Web Services is unable to obtain an error code from the Sage 100 system. This error represents an abnormal condition. Report the error to Sage 100 Customer Support.

WS_PreAuthFailed: "Could not pre-authorize credit card."

This error occurs when the PreAuthorizeCreditCard operation fails.

Common Errors Not Specific to eBusiness Web Services

The following are common error codes that are not specific to eBusiness Web Services

CI_NoKey: "Record does not exist."

This error occurs when the record for a get or update operation does not exist.

SY_InvalidPassword

- "The user logon or password does not match or the user logon is not enabled for web services."
- "Blank passwords cannot be used with web services."

This error occurs when the logon credentials for the Sage 100 system are incorrect. This error will occur when the user name does not exist, the password is incorrect, or the user name is not enabled for web services.

SY_NoAccess: "You are not authorized to perform this action."

This error occurs when the logon does not have proper rights in the Sage 100 system role security to perform an operation. For information on the rights required to perform an operation, see "APIs" (page 31).

This error may also occur when either the Sage 100 system or company is not enabled for web services. When this occurs, the SOAP faultstring will be:

- "Web services are not enabled for the system."
- "Web services are not enabled for this company."

The ErrorMessage under MasFault will remain "You are not authorized to perform this action."

SY_UserLocked: "This user account has been locked. See your system administrator."

This error occurs when the user account has the locked flag set in the Sage 100 system.

Quick Start

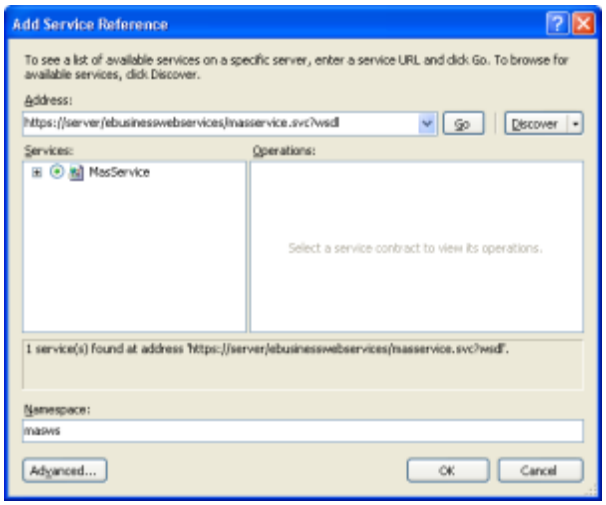
This section is designed to get you started with eBusiness Web Services quickly and easily. This section assumes that eBusiness Web Services has been installed with the default settings.

The following code examples are available on the Sage 100 installation program in the Documents\eBusiness Web Services Examples folder.

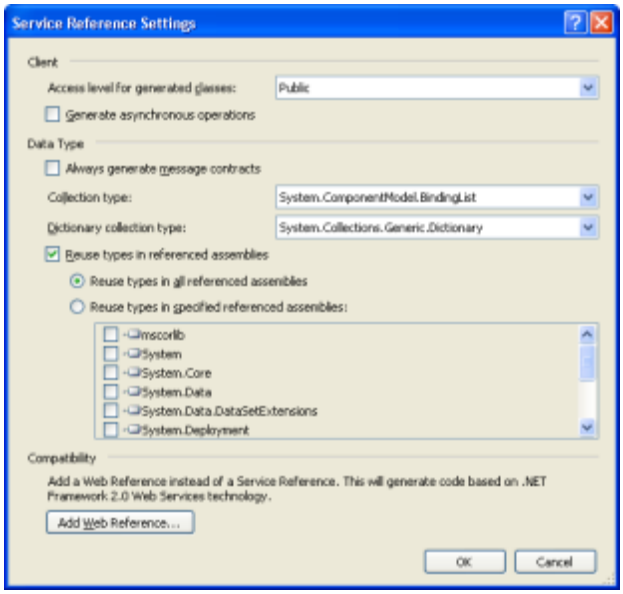
Visual Studio 2013 or 2008 (Other than Smart Device Projects)

1. Create a new project.
2. From the menu, choose **Project/Add Service Reference**.

3. At the Address field, enter:
   https://<servername>/ebusinesswebservices/masservice.svc?wsdl
4. Click **Go**. MasService will appear in the Services section.
5. Type **masws** in the **Namespace** field.



6. Click **Advanced**.
7. Select **System.ComponentModel.BindingList** in the **Collection type** field.



8. Click **OK** to close the Service Reference Settings window.
9. Click **OK** to close the Add Service Reference window.

Your project is now set up to use eBusiness Web Services.

C# Example

```
// An example of how to create a sales order using ABC demo
data
public void CreateSalesOrder()
{
// Create a sales order object
masws.SalesOrder salesOrder = new masws.SalesOrder();
// Create the lines collection for the sales order
salesOrder.Lines = new BindingList<masws.SalesOrderLine>();
// Set the customer information for the sales order
salesOrder.ARDivisionNo = "01";
salesOrder.CustomerNo = "ABF";
// Create a sales order line object
masws.SalesOrderLine salesOrderLine = new masws.SalesOrderLine
();
// Set the line information
salesOrderLine.ItemCode = "1001-HON-H252";
salesOrderLine.QuantityOrdered = 1;
// Add the line to the sales order
salesOrder.Lines.Add(salesOrderLine);
// Create a logon object
masws.Logon logon = new masws.Logon();
// Set the username and password to be used for Sage 100 Web
Services
logon.Username = "webservices";
```

```
logon.Password = "password";
// Create a string with the company code to use
string companyCode = "ABC";
// Use try/catch blocks to catch any errors
try
{
// Create a client to communicate with Sage 100 Web Services
masws.MasServiceClient client = new masws.MasServiceClient();
// Get a new sales order number
salesOrder.SalesOrderNo = client.GetNextSalesOrderNo(logon,
companyCode);
// Create the sales order
client.CreateSalesOrder(logon, companyCode, salesOrder);
// Display the order number
MessageBox.Show("Created order# " + salesOrder.SalesOrderNo);
}
// Catch Sage 100 specific faults that occur at the web server
catch (System.ServiceModel.FaultException<masws.MasFault> ex)
{
// Display the error message
MessageBox.Show(ex.Message);
}
// Catch other exceptions (e.g. a connectivity error)
catch (Exception ex)
{
// Display the error message
```

```
MessageBox.Show(ex.Message);
}
}
```

VB.Net Example

```
' An example of how to create a sales order using ABC demo
data
Sub CreateSalesOrder()
' Create a sales order object
Dim salesOrder As masws.SalesOrder = New masws.SalesOrder
' Create the lines collection for the sales order
salesOrder.Lines = New System.ComponentModel.BindingList(Of
masws.SalesOrderLine)
' Set the customer information for the sales order
salesOrder.ARDivisionNo = "01"
salesOrder.CustomerNo = "ABF"
' Create a sales order line object
Dim salesOrderLine As masws.SalesOrderLine = New
masws.SalesOrderLine
' Set the line information
salesOrderLine.ItemCode = "1001-HON-H252"
salesOrderLine.QuantityOrdered = 1
' Add the line to the sales order
salesOrder.Lines.Add(salesOrderLine)
' Create a logon object
Dim logon As masws.Logon = New masws.Logon
```

```
' Set the username and password to be used for Sage 100 Web
Services

logon.Username = "webservices"

logon.Password = "password"

' Create a string with the company code to use

Dim companyCode As String = "ABC"

' Use try/catch blocks to catch any errors

Try

' Create a client to communicate with Sage 100 Web Services

Dim client As masws.MasServiceClient = New
masws.MasServiceClient

' Get a new sales order number

salesOrder.SalesOrderNo = client.GetNextSalesOrderNo(logon,
companyCode)

' Create the sales order

client.CreateSalesOrder(logon, companyCode, salesOrder)

' Display the order number

MessageBox.Show("Created order# " + salesOrder.SalesOrderNo)

' Catch Sage 100 specific faults that occur at the web server

Catch ex As System.ServiceModel.FaultException(Of
masws.MasFault)

' Display the error message

MessageBox.Show(ex.Message)

' Catch other exceptions (e.g. a connectivity error)

Catch ex As Exception

' Display the error message

MessageBox.Show(ex.Message)
```

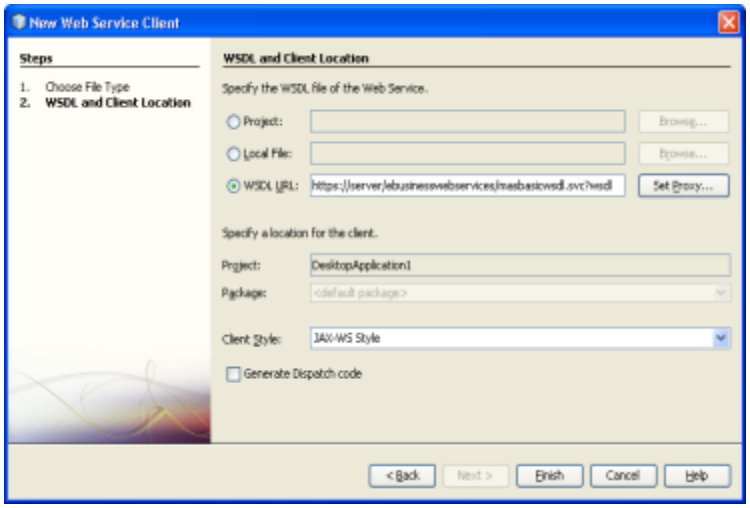
```
End Try

End Sub
```

Java using NetBeans IDE

1. Create a new project.
2. Right click the project and select **New/Web Service Client**.
3. In the New Web Service Client page, select the **WSDL URL** option and enter the following:

   https://<servername>/ebusinesswebservices/masbasicwsdl.svc?wsdl
4. Click **Finish**.



Your project is now set up to use eBusiness Web Services.

Java Example

```
// An example of how to create a sales order using ABC demo
data

public void CreateSalesOrder()

{

// Create a sales order object
```

```
com.sage.mas90.ws.SalesOrder salesOrder = new
com.sage.mas90.ws.SalesOrder();

// Create the lines collection

salesOrder.setLines(new
com.sage.mas90.ws.ArrayOfSalesOrderLine());

// Set the customer information for the sales order

salesOrder.setARDivisionNo("01");

salesOrder.setCustomerNo("ABF");

// Create a sales order line object

com.sage.mas90.ws.SalesOrderLine salesOrderLine = new
com.sage.mas90.ws.SalesOrderLine();

// Set the line information

salesOrderLine.setItemCode("1001-HON-H252");

salesOrderLine.setQuantityOrdered(1.0);

// Add the line to the lines collection then attach the lines
to the sales order

salesOrder.getLines().getSalesOrderLine().add(salesOrderLine);

// Create a logon object

com.sage.mas90.ws.Logon logon = new com.sage.mas90.ws.Logon();

// Set the username and password to be used for Sage 100 Web
Services

logon.setUsername("webservices");

logon.setPassword("password");

// Create a string with the company code to use

String companyCode = "ABC";

// Use try/catch blocks to catch any errors

try
```

```
{
  // Create a client to communicate with Sage 100 Web Services
com.sage.mas90.ws.MasBasicWsdlService service = new
com.sage.mas90.ws.MasBasicWsdlService();
com.sage.mas90.ws.IMasService client =
service.getBasicHttpBindingIMasService();
// Get a new sales order number
salesOrder.setSalesOrderNo(client.getNextSalesOrderNo(logon,
companyCode));
// Create the sales order
client.createSalesOrder(logon, companyCode, salesOrder);
// Display the order number
JOptionPane.showMessageDialog(null, "Created order# " +
salesOrder.getSalesOrderNo());
}
// Catch Sage 100 specific faults that occur at the web server
catch
(com.sage.mas90.ws.IMasServiceGetNextSalesOrderNoMasFaultFault
Message ex)
{
// Display the error message
JOptionPane.showMessageDialog(null, ex.getMessage());
}
// Catch Sage 100 specific faults that occur at the web server
catch
(com.sage.mas90.ws.IMasServiceCreateSalesOrderMasFaultFaultMes
sage ex)
{
```

```
// Display the error message
JOptionPane.showMessageDialog(null, ex.getMessage());
}
// Catch other exceptions (e.g. a connectivity error)
catch (Exception ex)
{
// Display the error message
JOptionPane.showMessageDialog(null, ex.getMessage());
}
}
```

PHP 5

PHP 5 is a loosely typed scripting language that does not generate proxies for its SOAP client. There is no setup procedure required to use eBusiness Web Services with PHP.

PHP 5 Example

```
function CreateSalesOrder()
{
// Set the customer information for the sales order
$salesOrder->ARDivisionNo = "01";
$salesOrder->CustomerNo = "ABF";
// Set the line information
$salesOrder->Lines[0]->ItemCode = "1001-HON-H252";
$salesOrder->Lines[0]->QuantityOrdered = 1;
// Set the username and password to be used for Sage 100 Web
Services
$logon->Username = "webservices";
$logon->Password = "password";
```

```
// Create a string with the company code to use
$companyCode = "ABC";
// Use try/catch blocks to catch any errors
try
{
// Create a client to communicate with Sage 100 Web Services
$client = new SoapClient
("https://server/ebusinesswebservices/masbasicwsdl.svc?wsdl");
// Get a new sales order number
$response = $client->GetNextSalesOrderNo(array(
"logon"=>$logon,
"companyCode"=>$companyCode)
);
$salesOrder->SalesOrderNo = $response-
>GetNextSalesOrderNoResult;
// Create the sales order
$client->CreateSalesOrder(array(
"logon"=>$logon,
"companyCode"=>$companyCode,
"salesOrder"=>$salesOrder)
);
// Display the order number
echo "Created order# $salesOrder->SalesOrderNo";
}
// Catch all exceptions
catch (Exception $ex)
```

```
{
// Display error message
echo $ex->getMessage();
}
}
```

Administration

Security and Encryption

eBusiness Web Services must be used with SSL encryption to be secure. The URL for eBusiness Web Services will start with https:// if SSL is being used. Sage 100 eBusiness Web Services should never be used without SSL encryption.

Access to the Sage 100 system from eBusiness Web Services is controlled at three levels:

- System
- Company
- User

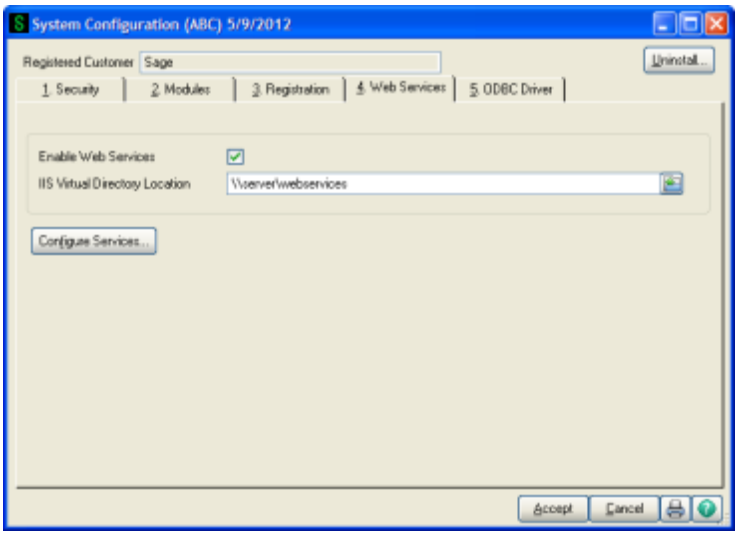
eBusiness Web Services works with Sage 100 role-based security. eBusiness Web Services calls are treated as maintenance actions (Create, Modify, Remove, or View) in the context of role-based security.

To determine which tasks must be enabled for a particular web services method, see "APIs" (page 31).

For example, to allow creation of a new customer, in Library Master Role Maintenance, click the Tasks tab and select Accounts Receivable > Main > Customer Maintenance > Create.

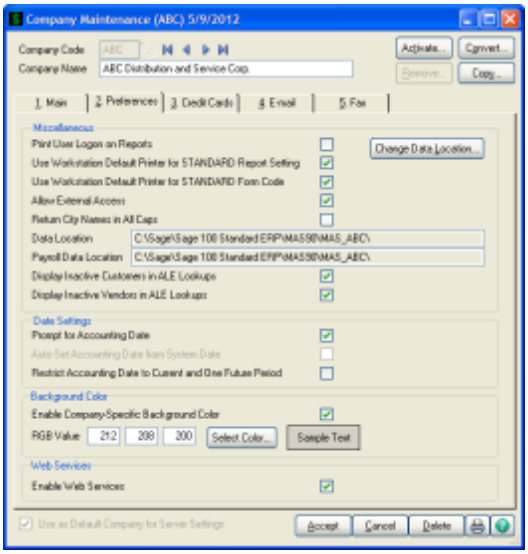
System Configuration Setup

eBusiness Web Services must be enabled for the system by selecting the Enable Web Services check box in Library Master Setup menu > System Configuration.



Company Maintenance Setup

eBusiness Web Services must be enabled for each company where access is required by selecting the Enable Web Services check box in Library Master Main menu > Company Maintenance.

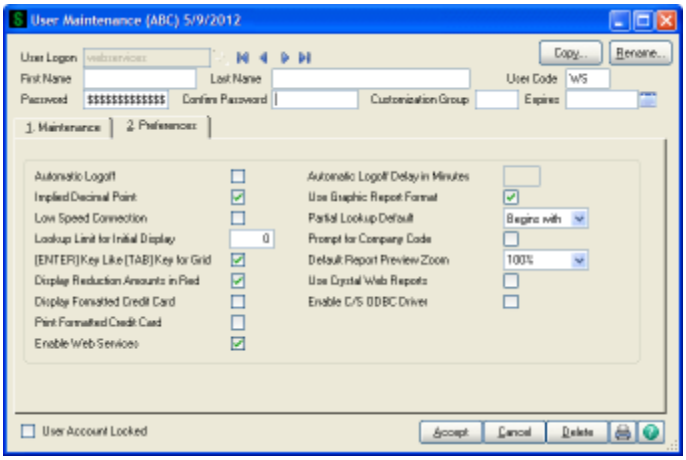


User Maintenance Setup

eBusiness Web Services must be enabled for each user that is allowed access to the Sage 100 system through eBusiness Web Services by selecting the Enable Web

Services check box in User Maintenance.

If you purchased Sage 100 through a subscription service, User Maintenance is on the Security menu; otherwise, it's on the Main menu.



Error Logging

eBusiness Web Services logs all errors to the Windows Event Log in the eBusiness Web Services log.

The details of unexpected exceptions are hidden from the web services client for security reasons. When these errors occur, the client will receive the message: "An exception occurred." in the SOAP fault. The full details of the error will be available in the Windows Event Log.

Object Cache

eBusiness Web Services uses an object cache to handle requests efficiently. When a request is made, eBusiness Web Services checks the cache to see if existing objects are available to fulfill the request. If objects are available then they are used. If objects are not available, then new objects are created and are added to the cache after the request is completed.

Objects that are added to the cache are available only to the same user name that created the objects. For this reason, the cache is more efficient if a common user name and password is used by the client application. For example, most ecommerce sites have no need to log onto eBusiness Web Services with multiple user names as the ecommerce

users purchasing items over the Internet will not have a Sage 100 account. Therefore, an ecommerce site would use a dedicated account to access eBusiness Web Services.

Certain maintenance operations in the Sage 100 products, such as adding a UDF, require exclusive use of the Sage 100 data files. Due to the use of caching, eBusiness Web Services can have data files open even if there is no activity. The eBusiness Web Services application pool should be stopped through Internet Information Services Manager before running operations that require exclusive use of the Sage 100 data files. Stopping the application pool will clear the cache and reject incoming web services requests that could interfere with the maintenance operations.

Permissions

eBusiness Web Services runs in the context of the eBusinessWebServices user account. The eBusinessWebServices user account requires permissions to the Sage 100 application and the files that the Sage 100 application accesses. The eBusiness Web Services installer will attempt to set these permissions; however, permissions may need to be set manually under certain circumstances such as when a new alternate directory is created.

The eBusinessWebServices user account must have the following permissions set:

- Full Control of the MAS90 base directory
- Full Control of all alternate directories that will be accessed from eBusiness Web Services
- Read & Execute permissions for the Program Files\Common Files\Sage\Common Components folder
- Read & Execute permissions of the eBusiness Web Services installation folder
- List Folder, Read Data, and Delete permissions for the Windows\Temp folder

APIs

GetContractInformation

Parameters

None

Return Value

Type: ContractInfomation
General information about the web services contracts.

Remarks

The returned value contains three fields:

ContractVersion: This field returns an integer value that represents the version of the eBusiness Web Services operations and data contracts.

- A client application can use ContractVersion to determine if it is compatible with the instance of eBusiness Web Services it is connected to without the need to download the complete WSDL.
- Newer versions of eBusiness Web Services will generally be backwards compatible with clients designed for older versions of eBusiness Web Services.
- Our goal is to maintain backwards compatibility; however, design considerations may cause incompatibles to arise from time to time.

AssemblyVersion: This field returns a string that represents the version of the eBusiness Web Services assembly.

NullDateTime: The value from this field can be used to explicitly set an entity's DateTime field to blank.

GetDiagnosticInformation

Parameters

None

Return Value

Type: DiagnosticInformation
Statistical data that may aid in diagnosing performance issues, etc.

Remarks

This method is for use by Sage 100 Customer Support as a diagnostic tool and is not intended to be called by applications.

Remote access to this method is turned off by default. Remote access to this method should only be enabled when requested by a Sage 100 Customer Support representative. Remote access can be enabled using the Sage 100 eBusiness Web Services configuration utility.

GetSalesOrder

Parameters

logon

> Type: Logon
> The user name and password to be used for the operation

companyCode

> Type: string
> The company from which to obtain the sales order

salesOrderNo

> Type: string
> The number of the sales order to retrieve

Return Value

Type: SalesOrder
The requested sales order

Remarks

The returned value may be modified and used with the UpdateSalesOrder operation. For more information, see "UpdateSalesOrder" (page 37).

For information on the structure and data contained in the SalesOrder type, refer to SO_SalesOrderHeader and SO_SalesOrderDetail in File Layouts and Program Information.

Security Requirements

In Library Master Role Maintenance, click the Tasks tab and select Sales Order > Main > Sales Order Entry > View.

GetNextSalesOrderNo

Parameters

logon

Type: Logon
The user name and password to be used for the operation

companyCode

Type: string
The company from which to obtain the next sales order number

Return Value

Type: string
The next sales order number

Remarks

The returned value can be used to set the SalesOrderNo field of a SalesOrder object before calling CreateSalesOrder.

Each call to GetNextSalesOrderNo will return a new sales order number for the specified company regardless of whether the number from a previous call has been used or not.

Security Requirements

In Library Master Role Maintenance, click the Tasks tab and select Sales Order > Main > Sales Order Entry > Create.

GetSalesOrderTemplate

Parameters

logon

Type: Logon
The user name and password to be used for the operation

companyCode

Type: string
The company from which to obtain the sales order template

arDivisionNo

Type: string
The division number of the customer

customerNo

Type: string
The customer number to use for the template

Return Value

Type: SalesOrder
An empty sales order with the default customer information set

Remarks

This operation retrieves default customer information that can be used for display on a data entry form. Sales order fields that are blank or zero in the Sage 100 system are returned as null in order to reduce network traffic.

The sales order returned from this operation can be modified and used for the CreateSalesOrder operation; however, it is generally more efficient to create a new SalesOrder object to pass to CreateSalesOrder.

For information on the structure and data contained in the SalesOrder type, refer to SO_SalesOrderHeader and SO_SalesOrderDetail in File Layouts and Program Information.

Security Requirements

In Library Master Role Maintenance, click the Tasks tab and select Sales Order > Main > Sales Order Entry. Select Create and View.

PreviewSalesOrder

Parameters

logon

Type: Logon
The user name and password to be used for the operation

companyCode

Type: string
The company from which to preview the sales order

salesOrder

Type: SalesOrder
The sales order to preview

Return Value

Type: SalesOrder
A sales order that includes the default values and totals that would be committed if the submitted sales order was passed to the CreateSalesOrder operation

Remarks

This operation allows a sales order to be previewed before being submitted to the CreateSalesOrder operation. For example, it would be appropriate for an ecommerce application to call PreviewSalesOrder and display the order totals to the end user before final checkout.

The sales order returned by this operation is not intended to be passed to the CreateSalesOrder operation. Read-only fields can be set on the returned value. This will cause an error if submitted to the CreateSalesOrder operation.

For information on the structure and data contained in the SalesOrder type, refer to SO_SalesOrderHeader and SO_SalesOrderDetail in File Layouts and Program Information.

Security Requirements

In Library Master Role Maintenance, click the Tasks tab and select Sales Order > Main > Sales Order Entry. Select Create and View or Modify and View.

CreateSalesOrder

Parameters

logon

Type: Logon
The user name and password to be used for the operation

companyCode

Type: string
The company in which to create the sales order

salesOrder

Type: SalesOrder
The sales order to create

Return Value

None

Remarks

This operation creates a sales order in the Sage 100 system. The SalesOrderNo field of salesOrder must be set to create a sales order. The GetNextSalesOrderNo operation can be used to obtain a sales order number.

Network connectivity issues can cause errors with web services. When this occurs, it can be difficult to determine if the call completed successfully, failed, or never made it to the server.

When a connectivity error occurs with this operation, such as a timeout error, CreateSalesOrder should be called again with the same sales order. It is important that the value for SalesOrderNo remain the same as in the first call. If the previous call to CreateSalesOrder completed successfully, the new call will error out and MasFault will have ErrorCode set to WS_AlreadyExists. This error can be ignored as it indicates the previous call successfully created the sales order. For more information, see "Error Handling" (page 12).

For information on the structure and data contained in the SalesOrder type, refer to SO_SalesOrderHeader and SO_SalesOrderDetail in File Layouts and Program Information.

Security Requirements

In Library Master Role Maintenance, click the Tasks tab and select Sales Order > Main > Sales Order Entry > Create.

UpdateSalesOrder

Parameters

logon

Type: Logon
The user name and password to be used for the operation

companyCode

Type: string
The company from which to update the sales order

salesOrder

Type: SalesOrder
The sales order to update

Return Value

None

Remarks

This operation updates a sales order in the Sage 100 system.

A sales order must first be obtained from the GetSalesOrder operation. This sales order can then be modified and passed to the UpdateSalesOrder operation.

The SalesOrderNo field of salesOrder determines which Sage 100 sales order is updated. SalesOrderNo must not be modified after the call to GetSalesOrder.

Do not modify the LineKey field of any of the SalesOrderLine objects contained in the Lines collection of salesOrder. The LineKey field is an advisory field that is used to determine lines that have been updated, inserted, or deleted from the sales order. Modifying the LineKey field will result in inefficiencies with the UpdateSalesOrder operation.

"Allow Read" should remain enabled for the LineKey field of the SalesOrderLine entity when using the advanced settings of the Sage Web Services Configuration utility.

For information on the structure and data contained in the SalesOrder type, refer to SO_SalesOrderHeader and SO_SalesOrderDetail in File Layouts and Program Information.

Security Requirements

In Library Master Role Maintenance, click the Tasks tab and select Sales Order > Main > Sales Order Entry > Modify.

DeleteSalesOrder

Parameters

logon

Type: Logon
The user name and password to be used for the operation

companyCode

Type: string
The company from which to delete the sales order

salesOrderNo

Type: string
The sales order to delete

Return Value

None

Remarks

This operation deletes the specified sales order from the Sage 100 system.

Security Requirements

In Library Master Role Maintenance, click the Tasks tab and select Sales Order > Main > Sales Order Entry > Remove.

GetCustomer

Parameters

logon

Type: Logon
The user name and password to be used for the operation

companyCode

Type: string
The company from which to obtain the customer

arDivisionNo

Type: string
The division number of the customer

customerNo

Type: string
The customer number to use for the template

Return Value

Type: Customer
The requested customer

Remarks

The returned value can be modified and used with the UpdateCustomer operation. For more information, see "UpdateCustomer" (page 41).

For information on the structure and data contained in the Customer type, refer to AR_Customer in File Layouts and Program Information.

Security Requirements

In Library Master Role Maintenance, click the Tasks tab and select Accounts Receivable > Main > Customer Maintenance > View.

GetNextCustomerNo

Parameters

logon

Type: Logon
The user name and password to be used for the operation

companyCode

Type: string
The company from which to obtain the next customer number

Return Value

Type: string
The next customer number

Remarks

The returned value can be used to set the CustomerNo field of a Customer object before calling CreateCustomer.

Each call to GetNextCustomerNo will return a new customer number for the specified company regardless of whether the number from a previous call has been used or not.

Security Requirements

In Library Master Role Maintenance, click the Tasks tab and select Accounts Receivable > Main > Customer Maintenance > Create.

CreateCustomer

Parameters

logon

Type: Logon
The user name and password to be used for the operation

companyCode

Type: string
The company in which to create the customer

customer

Type: Customer
The customer to create

Return Value

None

Remarks

This operation creates a customer in the Sage 100 system.

The ARDivisionNo and CustomerNo fields must be set to create a customer. The GetNextCustomerNo operation can be used to obtain a customer number.

Network connectivity issues can cause errors with web services. When this occurs, it can be difficult to determine if the call completed successfully, failed, or never made it to the server.

When a connectivity error occurs with this operation, such as a timeout error, CreateCustomer should be called again with the same customer. It is important that the values for ARDivisionNo and CustomerNo remain the same as in the first call. If the previous call to CreateCustomer completed successfully, the new call will error out and MasFault will have ErrorCode set to WS_AlreadyExists. This error can be ignored as it indicates the previous call successfully created the customer. For more information, see "Error Handling" (page 12).

For information on the structure and data contained in the Customer type, refer to AR_Customer in File Layouts and Program Information.

Security Requirements

In Library Master Role Maintenance, click the Tasks tab and select Accounts Receivable > Main > Customer Maintenance > Create.

UpdateCustomer

Parameters

logon

Type: Logon
The user name and password to be used for the operation

companyCode

Type: string
The company from which to update the customer

customer

Type: Customer
The customer to update

Return Value

None

Remarks

This operation updates a customer in the Sage 100 system.

A customer must first be obtained from the GetCustomer operation. This customer can then be modified and passed to the UpdateCustomer operation.

The ARDivisionNo and CustomerNo fields determine which Sage 100 customer is updated. ARDivisionNo and CustomerNo must not be modified after the call to GetCustomer.

For information on the structure and data contained in the Customer type, refer to AR_Customer in File Layouts and Program Information.

Security Requirements

In Library Master Role Maintenance, click the Tasks tab and select Accounts Receivable > Main > Customer Maintenance > Modify.

DeleteCustomer

Parameters

logon

Type: Logon
The user name and password to be used for the operation

companyCode

Type: string
The company from which to delete the customer

arDivisionNo

Type: string
The division number of the customer

customerNo

Type: string
The customer number to delete

Return Value

None

Remarks

This operation deletes a customer in the Sage 100 system.

Security Requirements

In Library Master Role Maintenance, click the Tasks tab and select Accounts Receivable > Main > Customer Maintenance > Remove.

GetCustomerContact

Parameters

logon

Type: Logon
The user name and password to be used for the operation

companyCode

Type: string
The company from which to obtain the customer contact

arDivisionNo

Type: string
The division number of the customer

customerNo

Type: string
The customer number of the contact

contactCode

Type: string
The contact code

Return Value

Type: CustomerContact
The requested customer contact

Remarks

The customer contact returned can be modified and used with the UpdateCustomerContact operation. For more information, see "UpdateCustomerContact" (page 45).

For information on the structure and data contained in the CustomerContact type, refer to AR_CustomerContact in File Layouts and Program Information.

Security Requirements

In Library Master Role Maintenance, click the Tasks tab and select Accounts Receivable > Main > Customer Maintenance > View.

CreateCustomerContact

Parameters

logon

Type: Logon
The user name and password to be used for the operation

companyCode

Type: string
The company in which to create the customer contact

customerContact

Type: CustomerContact
The customer contact to create

Return Value

None

Remarks

This operation creates a customer contact in the Sage 100 system.

The ARDivisionNo, CustomerNo, and ContactCode fields of customerContact must be set to create a customer contact.

Network connectivity issues can cause errors with web services. When this occurs, it can be difficult to determine if the call completed successfully, failed, or never made it to the server.

When a connectivity error occurs with this operation, such as a timeout error, CreateCustomerContact should be called again with the same customer contact. It is important that the values for ARDivisionNo, CustomerNo, and ContactCode remain the same as in the first call. If the previous call to CreateCustomerContact completed successfully, the new call will error out and MasFault will have ErrorCode set to "WS_AlreadyExists". This error can be ignored as it indicates the previous call successfully created the customer contact. For more information, see "Error Handling" (page 12).

For information on the structure and data contained in the CustomerContact type, refer to AR_CustomerContact in File Layouts and Program Information.

Security Requirements

In Library Master Role Maintenance, click the Tasks tab and select Accounts Receivable > Main > Customer Maintenance > Create.

UpdateCustomerContact

Parameters

logon

Type: Logon
The user name and password to be used for the operation

companyCode

Type: string
The company from which to update the customer contact

customerContact

Type: CustomerContact
The customer contact to update

Return Value

None

Remarks

This operation updates a customer contact in the Sage 100 system.

A customer contact must first be obtained from the GetCustomerContact operation. This customer contact can then be modified and passed to the UpdateCustomerContact operation.

The ARDivisionNo, CustomerNo, and ContactCode fields of customerContact determine which Sage 100 customer contact is updated. ARDivisionNo, CustomerNo, and ContactCode must not be modified after the call to GetCustomerContact.

For information on the structure and data contained in the CustomerContact type, refer to AR_CustomerContact in File Layouts and Program Information.

Security Requirements

In Library Master Role Maintenance, click the Tasks tab and select Accounts Receivable > Main > Customer Maintenance > Modify.

DeleteCustomerContact

Parameters

logon

Type: Logon
The user name and password to be used for the operation

companyCode

Type: string
The company in which to delete the customer contact

arDivisionNo

Type: string
The division number of the customer

customerNo

Type: string
The customer number of the customer contact

contactCode

Type: string
The customer contact to delete

Return Value

None

Remarks

This operation deletes a customer contact in the Sage 100 system.

Security Requirements

In Library Master Role Maintenance, click the Tasks tab and select Accounts Receivable > Maint > Customer Maintenance > Remove.

AddCreditCardToVault

Parameters

logon

Type: Logon
The user name and password to be used for the operation

companyCode

Type: string
The company from which to add the credit card data to the vault

paymentType

Type: string
The payment type used for the credit card

creditCardData

Type: string
The credit card number and expiration date formatted as follows:
nnnnnnnnnnnnnnn|mmyy

Return Value

Type: string
A GUID that can be used in future calls to PreAuthorizeCreditCard

Remarks

This operation adds a credit card number and expiration date to the Sage Exchange secure vault and returns a unique identifier that can be used in future calls to PreAuthorizeCreditCard and is also placed in the SalesOrder contract for various sales order operations.

The following is an example of data that should be passed into the creditCardData parameter:
4111111111111111|0616

Note that the pipe character is used to separate the credit card number from the expiration date.

PreAuthorizeCreditCard

Parameters

logon

Type: Logon
The user name and password to be used for the operation

companyCode

Type: string
The company from which to add the credit card data to the vault

paymentType

Type: string
The payment type used for the credit card

preAuthorizationData

Type: PreAuthorizationData
A contract that contains the credit card information necessary to process a pre-authorization

Return Value

Type: PreAuthorizationResult
A contract that contains CreditCardTransactionID and CreditCardAuthorizationNo fields that may be placed in the Payment collection of the SalesOrder contract for use in sales order operations

Remarks

Many fields for the PreAuthorizationData contract are optional and may be left as null.

The required fields are CreditCardGUID and Amount. An exception will be thrown if the pre-authorization fails.

Troubleshooting

Downloading WSDL - Wrong Host Name

WSDL contains URLs that link to schema information that describes eBusiness Web Services. Internet Information Services uses the full computer name of the server hosting eBusiness Web Services as the host name in these URLs. When the full computer name does not match the common name on the SSL certificate, or when the full computer name is not accessible from the internet, WSDL will fail to download.

This issue can be diagnosed by checking the host name returned by Internet Information Services. Point a browser to https://<host name>/<virtual directory>/<entry point>.

For example:
https://www.sage.com/ebusinesswebservices/masservice.svc

An informational page is displayed that contains a line similar to the following:

svcutil.exe https://www.sage.com/eBusinessWebServices/
MasService.svc?wsdl

When the host name on the informational page's URL is incorrect, the issue can be resolved by changing the full computer name of the server hosting eBusiness Web Services. If the full computer name cannot be changed, then SSL host headers can be used to resolve the issue.

Setting SSL Host Headers

The SSL host headers for Internet Information Services 8 can be set through the Bindings settings for the web site using Internet Information Services Manager. Internet Information Services version 6 and 7 do not allow SSL host headers to be set from the Internet Information Services Manager; therefore, special steps must be performed.

> **Important!** Setting SSL host headers prevents the affected web site from being accessed through other host names.

Internet Information Services 7

SSL host headers may be set by using notepad to edit applicationHost.config located in %windir%\System32\inetsrv\config. A backup copy of the applicationHost.config should be made before editing.

To set SSL host headers for the web site

Set the bindingInformation attribute for the web site's https protocol by adding the proper host name after the SSL port number. Add a colon
(:) between the port number and host name.

For example:

```
<site name="Default Web Site" id="1">
<application path="/">
<virtualDirectory path="/"
physicalPath="%SystemDrive%\inetpub\wwwroot" />
</application>
<application path="/eBusinessWebServices"
applicationPool="eBusinessWebServices"
enabledProtocols="https">
<virtualDirectory path="/" physicalPath="C:\Sage\eBusiness Web
Services" />
</application>
<bindings>
<binding protocol="http" bindingInformation="*:80:" />
<binding protocol="https"
bindingInformation="*:443:www.sage.com" />
</bindings>
</site>
```

> **Note:** Microsoft Internet Information Services (IIS) version 7 does not require a restart when this change is made.

Downloading WSDL - XSD not Available

The WSDL download can fail while downloading an XSD file.

To diagnose this issue

1. Verify that the issue is not a wrong host name. For more information, see "Downloading WSDL - Wrong Host Name" (page 50).
2. Verify that the main WSDL page can be read by pointing a browser to:

   https://<host name>/<virtual directory>/<entry point>?wsdl

   For example:

   https://www.sage.com/ebusinesswebservices/masservice.svc?wsdl
3. Verify that the XSD fails to display by pointing a browser to:

   https://<host name>/<virtual directory>/<entry point>?xsd=xsd0

   For example:

   https://www.sage.com/ebusinesswebservices/masservice.svc?xsd=xsd0

   If all three items above are true, restart Internet Information Services. If the issue still occurs, change the permissions for the %windir%\Temp directory to Full Control for the eBusiness Web Services user account, and restart Internet Information Services.

Error 404 with Internet Information Services (IIS) version 6

Under certain conditions, when eBusiness Web Services is installed in Internet Information Services version 6, web services calls may fail and the entry point will display an error 404 when accessed from a browser. For example:

https://www.sage.com/ebusinesswebservices/masservice.svc

To correct the error 404

1. Ensure that the ASP.NET v4.0.30319 web service extension is set to Allowed.

   Allow the ASP.NET v4.0.30319 Web Service Extension through Internet Information Services Manager.

2. If the ASP.NET v4.0.30319 web services extension is not listed, install it by running the following command from the command prompt:

   %windir%\Microsoft.NET\Framework\v4.0.30319\aspnet_regiis -i

3. Restart Internet Information Services after issuing the above command.

4. Verify that the ASP.NET v4.0.30319 is the ASP.NET version specified for the eBusiness Web Services application.

   In Internet Information Services Manager, right-click the eBusiness Web Services virtual directory, click Properties, and then click the ASP.NET tab.

Could Not Load Type "System.ServiceModel.Activation.HttpModule" in IIS 7

Under certain conditions, when eBusiness Web Services is installed in Internet Information Services version 7, web services calls may fail and the entry point will display an error referencing System.ServiceModel.Activation.HttpModule when accessed from a browser.

To correct the error

Run the following command from the command prompt using administrative privileges:

%windir%\Microsoft.NET\Framework\v4.0.30319\aspnet_regiis –i

General Errors

An error that occurs in eBusiness Web Services, but does not occur when accessing the same data directly from the Sage 100 system, may be caused by a permissions issue.

The most common permissions issue occurs when an alternate data directory is created after eBusiness Web Services is installed. The eBusiness Web Services user account must explicitly be given full control to the alternate directory. For more information, see "Permissions" (page 30).

Internet Information Services (IIS) must be restarted after correcting permissions issues.

Index

A

administration security, encryption 27
Advanced Settings
  MD fields 9
  UDFs 9
API
  AddCreditCardToVault 47
  CreateCustomer 40
  CreateCustomerContact 44
  CreateSalesOrder 36
  DeleteCustomer 42
  DeleteCustomerContact 46
  DeleteSalesOrder 38
  GetContractInformation 31
  GetCustomer 39
  GetCustomerContact 43
  GetDiagnosticInformation 31
  GetNextCustomerNo 40
  GetNextSalesOrderNo 33
  GetSalesOrder 32
  GetSalesOrderTemplate 33
  PreAuthorizeCreditCard 48
  PreviewSalesOrder 35
  UpdateCustomer 41
  UpdateCustomerContact 45
  UpdateSalesOrder 37

C

cache, object cache 29
Company Maintenance, setup 28
configuration,setup 27
credit cards
  decryption 8
  encryption 8
  processing 8

E

eBusiness Web Services
  advanced settings 10
  configuration utility 10
  date conversion 7
  installing 3
  manage entry points 12
  time conversion 7
  uninstalling 3
entry point, default 5
error handling, eBusiness Web Services 13

error messages
general 53
in IIS 6 52
in IIS 7 53

I

installing
eBusiness Web Services 3
user account 3

J

Java example 21

L

log, error logging 29

P

permissions, user account 30
PHP5 example 24
protocols, SOAP 6

S

setup
company maintenance 28
system configuration 27
user maintenance 28
SSL
encryption 27
security 27
standard language, WSDL 4
System Configuration, setup 27

T

troubleshooting 50

U

uninstalling, eBusiness Web Services 3
User Maintenance, setup 28

V

Visual Studios 2008 examples
C# 17
VB.Net 19

W

WSDL
types 5
wrong host name 50
XSD 52

X

XSD, WSDL 52