How to…

Configure the SOAP Web Services Tester

Contents

[Introduction 3](#_Toc434307348)

[Sage X3 Soap Web Services updates 4](#_Toc434307349)

[Sage X3 SOAP WS Tester Design 5](#_Toc434307350)

[Using the Sage X3 Web Services Tester 6](#_Toc434307351)

[UI Details 6](#_Toc434307352)

[Configuring Sage X3 11](#_Toc434307353)

[Sage X3 Soap WS Tester Authentication 13](#_Toc434307354)

[V6 Authentication 13](#_Toc434307355)

[U9 Authentication 13](#_Toc434307356)

[Reference 16](#_Toc434307357)

[Appendix 17](#_Toc434307358)

[Script XLISTBPC for Sub program LISTBPC 17](#_Toc434307359)

[Script XLISTBPC1 for subprogram LISTBPC1 18](#_Toc434307360)

[Script XGETITMPRI for Sub program GETITMPRI 19](#_Toc434307361)

# Introduction

This document outlines the design and development of the Sage X3 SOAP Web Services Tester and assumes knowledge of SOAP Web Services in Sage X3.

# Sage X3 Soap Web Services updates

In Sage X3 update 9, SOAP Web Services have been rewritten based on the Syracuse Framework. This change is primarily a technology rewrite of the setup and configuration. Publishing and consuming SOAP Web Services have not changed. Because SOAP Web Services are now rendered through the Syracuse Web Server, Apache Tomcat is no longer required.

**Setup:** Web service pools are now configured from the Sage X3 Administration menu.

**Licensing:** SOAP Web Services uses a new licensing model. The new model does not restrict the number of pools you can start or use. Instead, it controls the amount of data transmitted over these pools.

**Publish:** Publishing SOAP Web Services remains the same and stillsupports all methods, configurations, and parameters. The results are returned in the same formats. Messages, Reports, Traces, and Reply descriptions remain unchanged.

**Usage:** With the development of the new web services tester, the former web services tester has been deprecated.

**Authentication:** SOAP clients are now required to authenticate before calling a web service method.

**Debugging:** Using a specific port is no longer supported.

# Sage X3 SOAP WS Tester Design

The SageX3WsTester is a .NET application developed using C#. It uses Windows Forms (WinForms) for the user interface (UI).

The development environment:

* WinForm application developed in C# using Visual Studio 2013
* .NET Framework 4.5

The Sage X3 SOAP Web Service Tester solution uses one project with the following components:

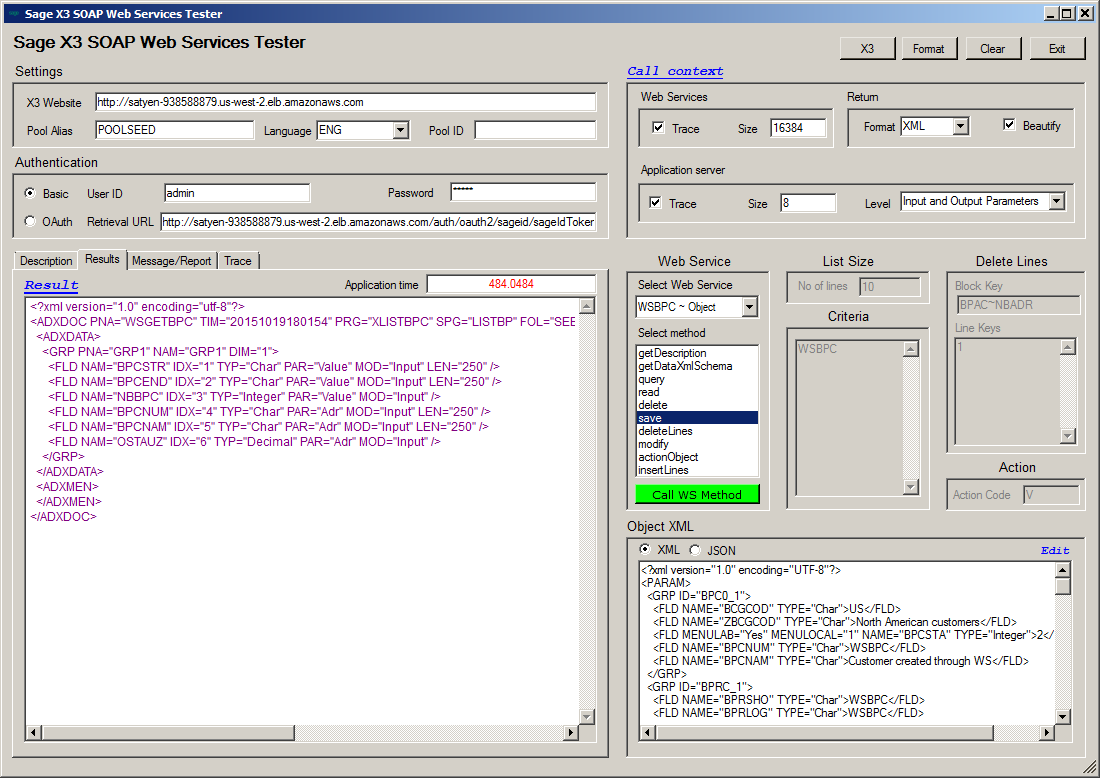
* A web reference to the Sage X3 CAdxWebServiceXmlCC object
* CAdxWebServiceXmlCCServiceBasic class inherited from CAdxWebServiceXmlCCService
* This class overrides the GetWebRequest authorization method before calling the web method
* Supports both Basic authentication and Sage ID authentication
* A WinForm (FrmMain) that drives the UI
* An app.config file that provides default values

# Using the Sage X3 Web Services Tester

Other than authentication, consuming SOAP Web Services remains the same and works similar to the former web tester. Most of the UI components in the tool have been automated to simplify its usage.

## UI Details

The tool uses a single dialog box to consume SOAP Web Services that Sage X3 offers.



At the top of the interface, you have four options:

**X3:** Click this button to launch Sage X3. The URL for Sage X3 is derived by concatenating the Sage X3 website as follows:

X3 application location = X3 web site + "/auth/login/page"

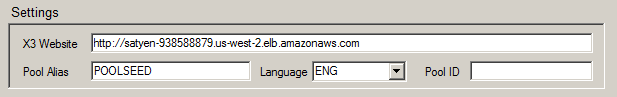
**Format:** Click this button to format results on the **Result** tab. The data returned by the web server are not formatted unless you select Beautify under **Call context**.

**Clear:** Click this button to clear all settings and results.

**Exit:** Click this button to leave the application.

Settings

This section defines the basic details for the SOAP Web Services.



**Sage X3 Website**: Enter the basic site address of your Sage X3 installation. The tool concatenates the location strings to this URL to determine the web services location and Sage X3 login location as follows:

Web Service location = X3 web site + "/soap-generic/syracuse/collaboration/syracuse/CAdxWebServiceXmlCC"

X3 application location = X3 web site + "/auth/login/page"

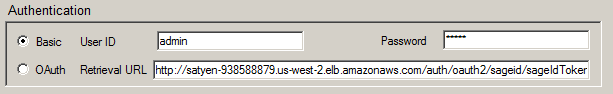
**Pool Alias**: Enter the alias name of the pool configuration. Pool configurations are defined in Sage X3 using the **Classic SOAP Pools configuration** entity from the **Administration** menu.

**Language**: Select the language to use for the Sage X3 connection. If not entered, the web service uses the default language as defined in the pool alias.

**Pool ID:** Pool ID (X3Pid) that services the web service request. This is optional.

Authentication

This section defines the authentication type (Basic or OAuth) and the details of that authentication.



**Basic**: Select this option if using Sage X3 basic authentication.

**User ID**: Enter your Sage X3 user ID.

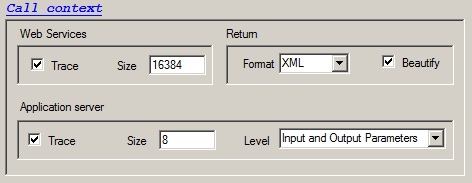
**Password**: Enter your Sage X3 password.

**OAuth**: Select this option if using Sage ID for authentication.

**Retrieval URL**: Enter the URL you received form Sage X3 when authenticating using OAuth.

Call context

This section prepares the request configuration. The requestConfig property in the CAdxCallContext class contains a string with elements separated by the ampersand (&) character. It specifies request processing settings. Click **Call context** to view the requestConfig in the string format.



**Web Services**

**Trace:** Select this check box to activate the web server trace option.

**Size:** Enter the maximum trace file size.

**Return**

**Format:** Select **XML or JSON** from the drop down menu to return the results in the selected format, XML or JSON.

**Beautify**: Select this option to format the return string.

**Application Server**

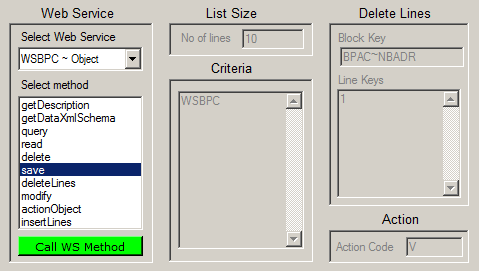
**Trace**: Select this option to activate the X3 server trace option.

**Size**: Enter the maximum trace size (Clob size does not change).

**Level**: Select **Input Parameters**, **Output Parameters**, or **Input and Output Parameters** from the drop down menu for the trace level.

Web Service

Use this section to select the web service name and method for calling the web service. The web service names are data driven. The values in the **Select Web Service** drop down list are populated on the fly by reading all the subfolder names under the folder **ParamFiles** to generate this list.



The following example illustrates how to extend the UI to support a new web service for Purchase Orders.

In Sage X3, publish the object **POH** and name the web service **WSPOH**.

* Adding “WS” to the name is an example of a naming convention. It is important to use this same name for the folder.
* Create the folder **ParamFiles\Object\WSPOH**. Once created, this folder displays in the Select Web Service drop down menu.
* Create the required supporting files within the folder **WSPOH** to populate the Object XML.

**Select Web Service**: This list contains a list of web services that you published in Sage X3. Sage X3 allows for publishing of two types of web services: objects and subprograms.

* The web service methods populate automatically based on the type of web service selected.

**Select method**: Select the web service method you want to call.

* The web service methods populate automatically based on the type of web service you selected. Other UI elements are enabled, disabled, or auto populated based on the method you select.

**Call WS Method:** Use this button to perform the web service operation you have selected and configured.

List Size

**No of lines**: Enter the number of records you want to return for the Query.

Criteria

Use this section to define the filters you want to use for the Query and Read methods.

**Criteria**: Enter the filter values for the Query or Read operation. In this section you enter just the values, the fields names are picked from the web service description.

Delete Lines

Sage X3 allows you to delete the detail lines for a given object. For example, you can delete a Customer address from a customer object, or Line item details from a Sales Order object. Use this section to define the details for the delete line operation.

**Block Key**: Enter the block name that you want to delete.

Each master can have one or more details. Each detail line is defined by a block key, refer to XML definitions for details.

**Line Keys**: Specify the line number you want to delete

Action

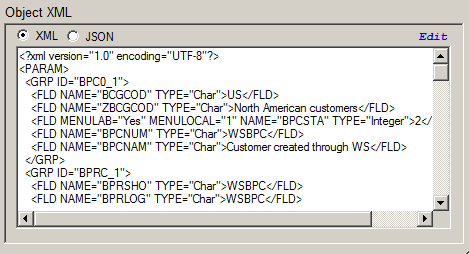
**Action Code**: Enter the object action code here.

Sage X3 object allows additional object actions—such as the button code in the object dictionary—to be accessed through a web service. Refer to the ADXSER tag of the XML description file.

* Each action might require parameters. You need to pass the appropriate parameter for the actions.

Object XML

Web service operations like Run, Save, and Action Code requires you to pass input values through the Object XML parameter. Sage X3 web services allow you to enter these values either in XML or JSON. This tool includes the Object XML parameters for sample web services. These files, included as XML or JSON, and are located in the subfolder under **ParamFiles**.

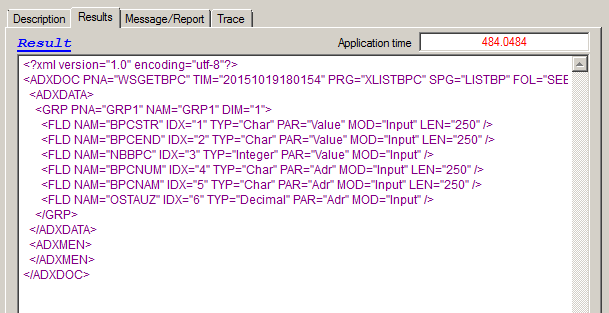


**XML:** Select this button for the **XML** version of the Object XML parameter file. If the file is not available, you can enter the parameter as an XML string.

**JSON:** Select this button for the **JSON** version of the Object XML parameter file. If the file is not available, you can enter the parameter as a JSON string.

**Edit:** Click **Edit** to modify the Object XML parameter file (\*.XML / \*.JSON) included with the tool.

Result



**Description:** This tab displays the reply description that include status, trace response, durations, etc.

**Results:** This tab displays the results of web service operations. The details show if the operation was successful, or an error message if the operation failed. Click **Format** at the top of the UI or select the **Beautify** check box under **Call context** to format the results.

**Message/Report:** This tab displays the error message and the report returned when a web service operation fails.

**Trace:** This tab displays the Web Service and Application service traces if the **Traces** check box is selected under **Call context**.

## Configuring Sage X3

The tool includes sample parameter files for many web services. In order to run pre-configure samples, you must first publish the samples in Sage X3.

**Objects**

* WSAUS — Publish AUS object can name it WSAUS
* WSBPC — Publish BPC object can name it WSBPC
* WSITM — Publish ITM object can name it WSITM
* WSITS — Publish ITS object can name it WSITS
* WSREP — Publish REP object can name it WSREP
* WSSOH\_WS — Publish REP object (alternately, you could name it WSSOHWS)
* Creates a new **Sales Order Type** named **WSSOH\_WS** that allows you enter the Sales Order number.
* Publishes the new **Sales Order Type**.

**Subprograms**

* WSGETBPC
* Gets customer list using the starting/ending string
* Creates a Sub-program LISTBP using the script XLISTBP
* Publishes the subprogram LISTBP as WSGETBPC
* WSGETBPC1 — Publishes the BPC object; you could name it WSBPC
* Gets 50 customer-based input parameters
* Creates a subprogram LISTBP1 using the script XLISTBP1
* Publishes the subprogram LISTBP1 as WSGETBPC1
* WSGETITM — Publishes the ITM object; you could name it WSITM
* Gets price for an item passed in as parameter
* Creates a Sub-program GETITMPRI using the script XGETITMPRI
* Publishes the sub-program XGETITMPRI as WSGETITM

# Sage X3 Soap WS Tester Authentication

On the client side, the only major change between the V6 and U9 is the authentication.

## V6 Authentication

SOAP calls in V6 do not require pre-authentication and therefore the authentication details are passed in the call context. The sample code below shows how to implement the authentication in V6.

CAdxWebServiceXmlCC.CAdxCallContext callContext = new CAdxWebServiceXmlCC.CAdxCallContext();

callContext.codeUser = ‘admin’ // X3 user id

## U9 Authentication

Sage X3 is now available in the cloud as a SAS model and as the conventional on premise solution. Each implementation has different authentication methods.

**Basic Authentication**

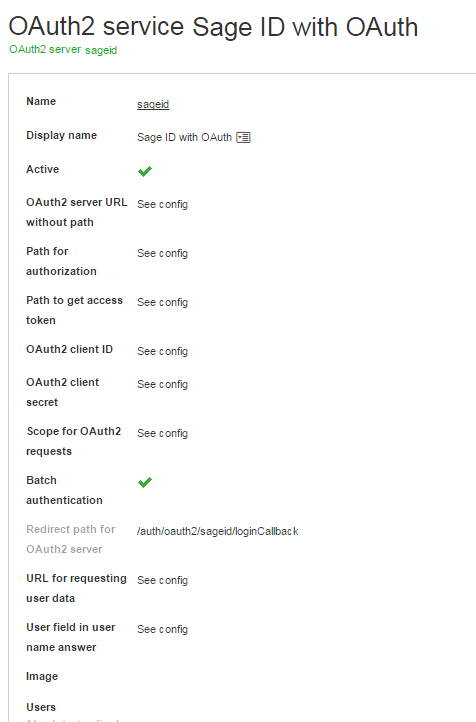
The on premise version of Sage X3 supports the basic authentication, so you can use a Syracuse login and password to connect to Sage X3. Starting with U9, the SOAP client requires a pre-authentication for making SOAP calls. This is achieved by extending the CAdxWebServiceXmlCCService object, intercepting the GetWebRequest method, and passing in the authentication details. To access web services for an on premise install, use the basic authentication option and enter the Syracuse user ID and password.

**Sage-ID OAuth**

Sage X3 cloud does not support basic authentication. Users have to be authenticated using Sage ID in order to connect to Sage X3. The SOAP client also requires a Sage OAuth authentication. This is achieved by extending the CAdxWebServiceXmlCCService object, intercepting the GetWebRequest method, and passing in the OAuth access token. To access web services for Sage X3 cloud solution, use the following steps:

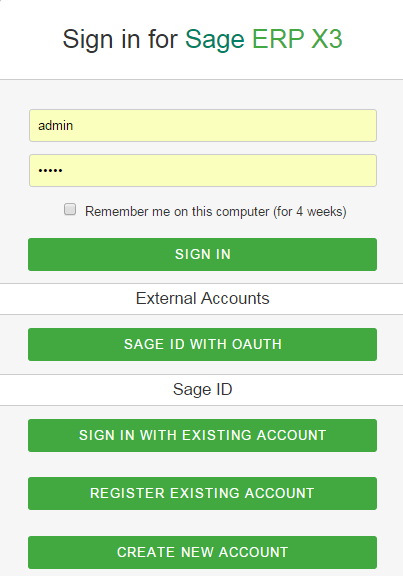
1. Configure OAuth2 Server

* Log in into Sage X3
* From the **Administration** menu, launch the **OAuth2 Sever** entity.
* Create a new OAuth2 service as follows
* Name = sageid
* Display name : Sage ID with OAuth (or anything that makes sense)
* Active = check
* Batch authentication = Check
* Enter **See config** for everything else

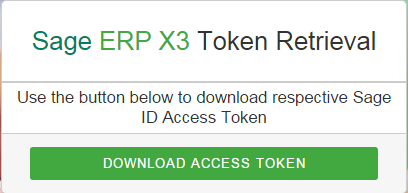


1. Get Retrieval URL

* Log out of Sage X3.
* Navigate to your Sage X3 website to view the new entry on the login page.



* Click **Sage ID WITH OAUTH** to login using Sage ID authentication.



* Click **Download Access Token** to open the download accessToken.json file

{

"\_id": "ff90e83a-67ec-4b5d-984c-2ac696d1d403",

"tokenData": {

"access\_token": …..,

"refresh\_token": "…..",

"expirationDate": "2015-10-19T18:14:52.294Z",

"retrievalUrl": "http://xxxxx/auth/oauth2/sageid/sageIdTokenRetrieval/xxxxx"

}

}

* Cut and paste the retrieval URL in the .Net teste app.config file.

# Reference

Detailed information on creating and using Sage X3 Soap Web Services is available through Sage University.

# Appendix

## Script XLISTBPC for Sub program LISTBPC

#\* XLISTBPC

#\* BPCLIST: List of customers designed For Web Services

#\* @353 BPCSTR Start range

#\* @353 BPCEND End range

#\* @353 NBBPC Number of customers 1n the 11st (dimension)

#\* @353 BPCNUM Customer code

#\* @353 BPCNAM Customer name

#\* @353 OSTAUZ Customer ;;gg,;1

#\*!

$INITWS

BPCSTR = "AU001"

BPCEND = "BB001"

NBBPC=10

Local Char XBPCNUM (40) (0..NBBPC)

Local Char XBPCNAM (40) (0..NBBPC)

Local Decimal XOSTAUZ (0..NBBPC)

Call LISTBP(BPCSTR, BPCEND, NBBPC, XBPCNUM, XBPCNAM, XOSTAUZ)

#Call LISTBP(BPCSTR, BPCEND, NBBPC)

For J = 0 To NBBPC-1

Infbox XBPCNUM(J)

# Infbox XBPCNAM(J)

Next

End

$RESULTWS

Local Integer I

For I = 1 To NBBPC

Call ECR\_TRACE("Name of the customer: " -BPCNAM(I), 0) From GESECRAN

Call ECR\_TRACE("Code of the customer: " -BPCNUM(I), 0) From GESECRAN

Call ECR\_TRACE("AUTHORIZED CREDIT: " -num$(OSTAUZ(I)), 0) From GESECRAN

Call ECR\_TRACE("------------------",0) From GESECRAN

Next

Return

#- WS subprog - get customer list

Subprog LISTBP(BPCSTR, BPCEND, NBBPC, BPCNUM, BPCNAM, OSTAUZ)

Value Char BPCSTR

Value Char BPCEND

Value Integer NBBPC

Variable Char BPCNUM () ()

Variable Char BPCNAM () ()

Variable Decimal OSTAUZ ()

Local File BPCUSTOMER [BPC]

Local Integer I : I = 0

For [BPC] Where BPCNUM >= BPCSTR & BPCNUM<= BPCEND

BPCNUM(I) = [BPC]BPCNUM

BPCNAM(I) = [BPC]BPCNAM

OSTAUZ(I) = [BPC]OSTAUZ

I += 1

If I >= NBBPC : Break : Endif

Next

End

## Script XLISTBPC1 for subprogram LISTBPC1

#\* XLISTBPC1

#\* BPCLIST1: List oF customers designed For Heb Services

#\* the sub-program test takign 10 parameters

#\* @353 BPCSTR Start range

#\* @353 BPCEND End range

#\* @353 NBBPC Number oF customers 1n the 11st (dimension)

#\* @353 BPCNUM Customer code

#\* @353 BPCNAM Customer name

#\* @353 OSTAUZ Customer ;;gg,;1

#\*!

$INITWS

NBBPC=10

Local Char XBPCNUM (40) (0..NBBPC)

Local Char XBPCNAM (40) (0..NBBPC)

Local Decimal XOSTAUZ (0..NBBPC)

XBPCNUM (0) = "ZA001"

XBPCNUM (1) = "PT001"

XBPCNUM (2) = "APA01"

XBPCNUM (3) = "TEST"

XBPCNUM (4) = "NZ001"

XBPCNUM (5) = "GB001"

XBPCNUM (6) = "SPA01"

XBPCNUM (7) = "NA001"

XBPCNUM (8) = "ES001"

XBPCNUM (9) = "GB001"

Call LISTBP1(NBBPC, XBPCNUM, XBPCNAM, XOSTAUZ)

For J = 0 To NBBPC-1

Infbox XBPCNAM(J)

# Infbox XBPCNAM(J)

Next

End

$RESULTWS

Local Integer I

For I = 1 To NBBPC

Call ECR\_TRACE("Name of the customer: " -BPCNAM(I), 0) From GESECRAN

Call ECR\_TRACE("Code of the customer: " -BPCNUM(I), 0) From GESECRAN

Call ECR\_TRACE("AUTHORIZED CREDIT: " -num$(OSTAUZ(I)), 0) From GESECRAN

Call ECR\_TRACE("------------------",0) From GESECRAN

Next

Return

#- WS subprog - get customer list

Subprog LISTBP1(NBBPC, BPCNUM, BPCNAM, OSTAUZ)

Value Integer NBBPC

Variable Char BPCNUM () ()

Variable Char BPCNAM () ()

Variable Decimal OSTAUZ ()

Local File BPCUSTOMER [BPC]

Local Integer I : I = 0

For I = 0 To NBBPC-1

If (BPCNUM(I) <>"")

# Filter [BPC] Where BPCNUM =BPCNUM(I)

# Look [BPC]BPCNUM=BPCNUM(I)

Read [BPC]BPC0 = BPCNUM(I)

If fstat=0

BPCNAM(I) = [BPC]BPCNAM

OSTAUZ(I) = [BPC]OSTAUZ

Endif

Else

# Populate empty values

Endif

Next

End

## Script XGETITMPRI for Sub program GETITMPRI

$INITWS

Local Char ITM(30):ITM="BMS004"

Local Decimal PRI : PRI=0

Dbgaff

Infbox "test"

Call GETITMPRI(ITM,PRI)

End

$RESULTWS

Call ECR\_TRACE("Price: " -num$(PRI), 0) From GESECRAN

Return

#- WS subprog - get product base price

Subprog GETITMPRI(ITM,PRI)

Value Char ITM

Variable Decimal PRI

Local File ITMSALES [ITS]

Read [ITS]ITS0=ITM

If fstat

Call ERREUR("Product code " + ITM + " not found") From GESECRAN

Raz[ITS]

Endif

PRI=[ITS]BASPRI

Infbox "Price: "-num$(PRI)

Close Local File [ITS]

End