
User Defined Web Services

Prepared by:
Prepared for:
Module:
Date:
Document Ref:
Version:

Documentation Team
Learning Resources
Business Intelligence
2017
LMDSY0037
11.06

Construction Industry Solutions Ltd.
11 St. Laurence Way
SL1 2EA





Copyright 2017 Construction Industry Solutions Ltd.. All rights reserved.

Information in this document is subject to change without notice. The software described in this document is furnished under a license agreement or nondisclosure agreement. The software may be used or copied only in accordance with the terms of those agreements. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or any means electronic or mechanical, including photocopying and recording for any purpose other than the purchaser's personal use without the written permission of Construction Industry Solutions Ltd..

Construction Industry Solutions Ltd.
11 St. Laurence Way
SL1 2EA

THIS USER GUIDE WAS CREATED USING MADCAP FLARE.

Contents

1 Introduction or Executive Summary	1
2 Viewing Installed Services	2
3 Testing a Service Connection (Using soapUI)	4
4 ESB COINS User	14
4 soapUI	15
5 Licencing	17
5.1 Additional Licencing	17
5.2 Roles	19
5.3 User Maintenance	21
6 User Defined Services	22
6.1 Datasets	23
6.1.1 Function	23
6.1.2 Dataset	23
6.1.3 Page	25
6.1.4 Adding the Function to the Web services Menu	26
6.1.5 Testing the Service	30
6.2 Pages	34
6.2.1 Function	34
6.2.2 Page	34
6.2.2.1 Body Form	36
6.2.2.2 Update Form	36
6.2.2.3 Detail Form	37
6.2.3 Adding the Function to the Web services Menu	38
6.2.4 Testing the Service	41
6.2.4.1 Using the ID field when adding multiple records	43
6.3 Calculation Programs	46
6.3.1 Defining the Calculation	46
6.3.2 Testing Calculation Programs (run, run\$)	47
6.3.3 Using Calculation Programs with Web Services	49
6.3.3.1 Create the Function	50
6.3.3.2 Create the Page	51
6.3.3.3 Adding the Function to the Web services Menu	52
6.3.4 Testing the Service	55
6.4 Get/Set	58



1 Introduction or Executive Summary

This document describes the capability for the user to define services using existing components of COINS.

The user is able to define the following general capabilities using OA designer:

- Datasets - to extract data records for use on reports
- Page Maintenance - which can include add, update and delete capabilities
- Calculation Programs - that can run low level methods in COINS RSPs to extract and/or manipulate data

It is now possible for the user to also define an interface that matches up to these generic features so that they can be run as a service to be exposed on an Enterprise Service Bus (ESB) or as a Webservice.

2 Viewing Installed Services

To view the services installed on your system, take your standard environment access URL, such as:





<http://195.40.14.50/cgi-bin/oartvol11/wologin.p>

and replace wologin.p (and anything after it) with wouesp.p

For example:

<http://195.40.14.50/cgi-bin/oartvol11/wouesb.p>

The following page should be displayed.

<div>     </div> <div> 1337id/1337id https://www.docker.com https://t.me/1337id 1337id@protonmail.com </div>	
<h1>COINS Services</h1>	
Service	Description
CR	Coin Base
EC	ECoinchain
FM	Fabric Management
GL	General Ledger
HN	Hyper Ledger
IB	IBM Blockchain
IC	Blockchain Suite
PL	Platform Ledger
SC	Software Ledger
NL	Native Ledger
SL	Smart Ledger
SV	Bitcoin
VP	Vault
PC	Proof
EM	Electronic Management
CV	Coinventor
CT	Coinventor, Innovation, Workchain
CC	Coinventor Core
LA	Local Apparent
CVS	Coinventor
EN	Blockchain
MB	Mobile Applications
NS	Notary
ST	Smart

The contents of this page will vary, depending on the services installed on your particular system.

Please note that this page will be displayed regardless of the licencing installed on your system - this means that although you will be able to view the services, you may not be able to run certain ones if you have not been licenced to do so.

Selecting a Module Service, for example JC - Contract Status, will display the services for that module:



COINS Services

COINS Services > Contract Status

Service	Description
JCESB001	Contract Section Maintenance
JCESB002	Project/Contract/Site Create
JCESB003	Contract Reconciliation
JCESB004	Costs-Revenue-Debtors-Credit
JCESB005	Cost Transaction Load
JCESB006	Revenue Transaction Load
JCESB007	Contract Changes
JCESB008	Company Changes
JCESB009	Contract Cost Transactions
JCESB010	Contract Section Maintenance
JCESB011	WBS Code Changes
JCESB017	Timesheet Master Data
JCESB015	Contracts and WBS Codes List
JCESB016	ic Mobile Webservice
JCESB018	Cost Code List

If you then drill down into a particular service, For example JCESB008 - Company Changes, you will be taken into a page detailing the schema, field definitions and sample messages for the service.

COINS Services

COINS Services > Contract Status > Company Changes

Service	JCESB008
Description	Company Changes
Schema	Input.xsd output.xsd exception.xsd acknowledgement.xsd SOAPInput.xsd SOAPOutput.xsd
WSDL	JCESB008.wsdl
WSDL NS	JCESB008.xsd

The companies changes service will return the companies that have changed since the specified date for a selected set of companies.

Input

Entity	Type	Documentation
COINSInterface		
Header		
id	string	A unique message identifier from the originating system.
confirm	string	Whether a confirmation message is required. Set to true or yes to have a confirmation message returned. If this is set then an id must also be provided.
action	string	An action type that will indicate why the message was created. For example, this might be CREATE, UPDATE, or DELETE for a message as a result of a maintenance of a record or PUBLISH for the publish of some information.
entity	string	The service that should consume the message.
arguments	string	Arguments that should be passed to the service.
ackID	string	An optional acknowledgement ID. If this ID is set then COINS will respond with an acknowledgement message returning this ackID in the message header.
testMsg	boolean	Whether this message is a test message. A test message will process through in exactly the same way that a normal message would except that at the point where it would normally commit the transaction to the database, the message is then backed out.
UserID	string	The user in the originating system that caused the message to be produced.
From	string	The name of the system that produced the message.
HostName	string	This is the name of the host system (on UNIX the HOSTNAME environment variable). This is automatically checked (if present in the message) and if it does not match the host name of the system then an exception message is created. If the HostName tag is missing then HostName checking is omitted.
Environment	string	This is the name of the COINS environment that is the intended target for the message. This is automatically checked based on the name of the instance of the COINS system if it does not match then an exception message is created.
Created	dateTime	Date/Time of the source system when the message was created.
Version	string	
Login		

If this page does not show the fields and sample messages, Web Services has not been configured and you will need to contact COINS Support to arrange for Tech Services to investigate



3 Testing a Service Connection (Using soapUI)

From your browser session, using the JC - Contract Service as our example, select the Company Changes option. In the first schema Panel, locate the WSDL entry.

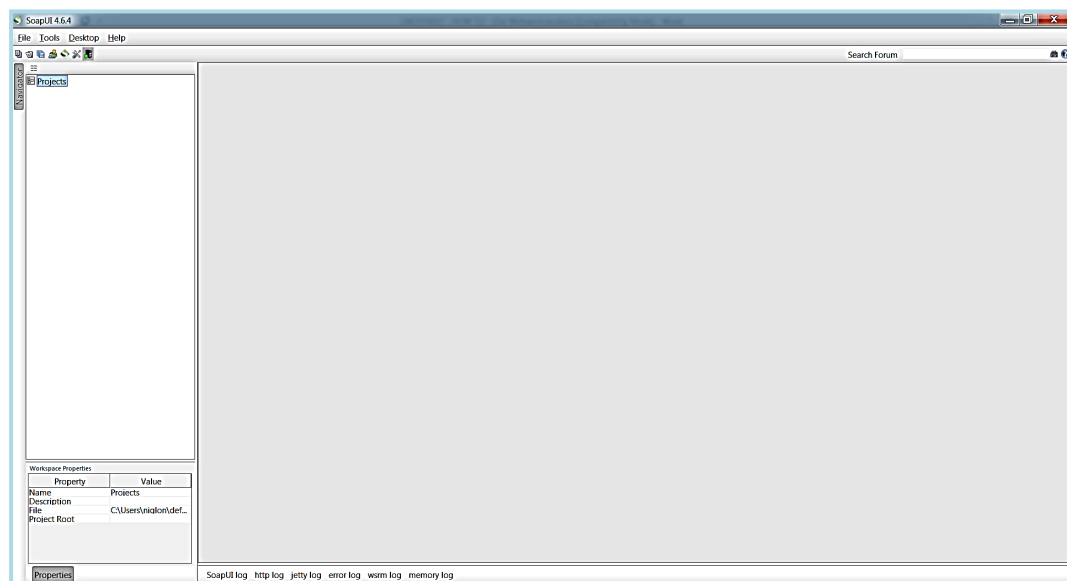
COINS Services

[COINS Services](#) > [Job Status](#) > [Company Changes](#)

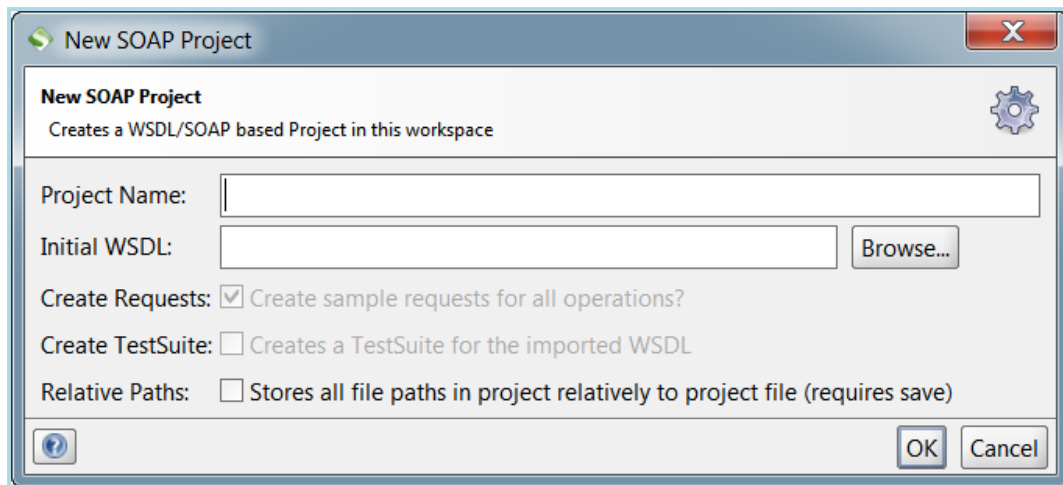
Service	JCESB008
Description	Company Changes
Schema	input.xsd output.xsd exception.xsd acknowledgement.xsd SOAPInput.xsd SOAPOutput.xsd
WSDL	JCESB008.wsdl
WSDL NS	JCESB008.wsdl

Right-click on this and select Copy Shortcut.

Launch the soapUI application.

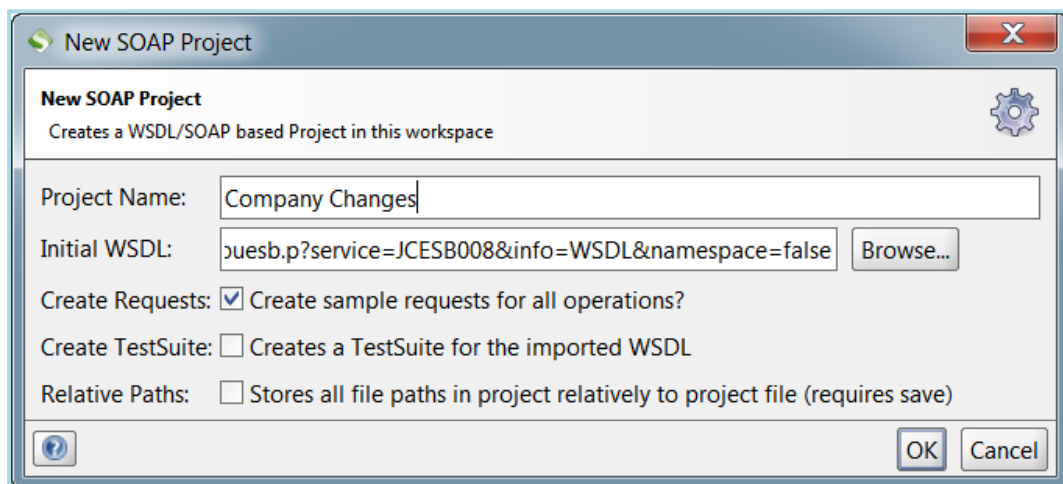


Select File/New SOAP Project.



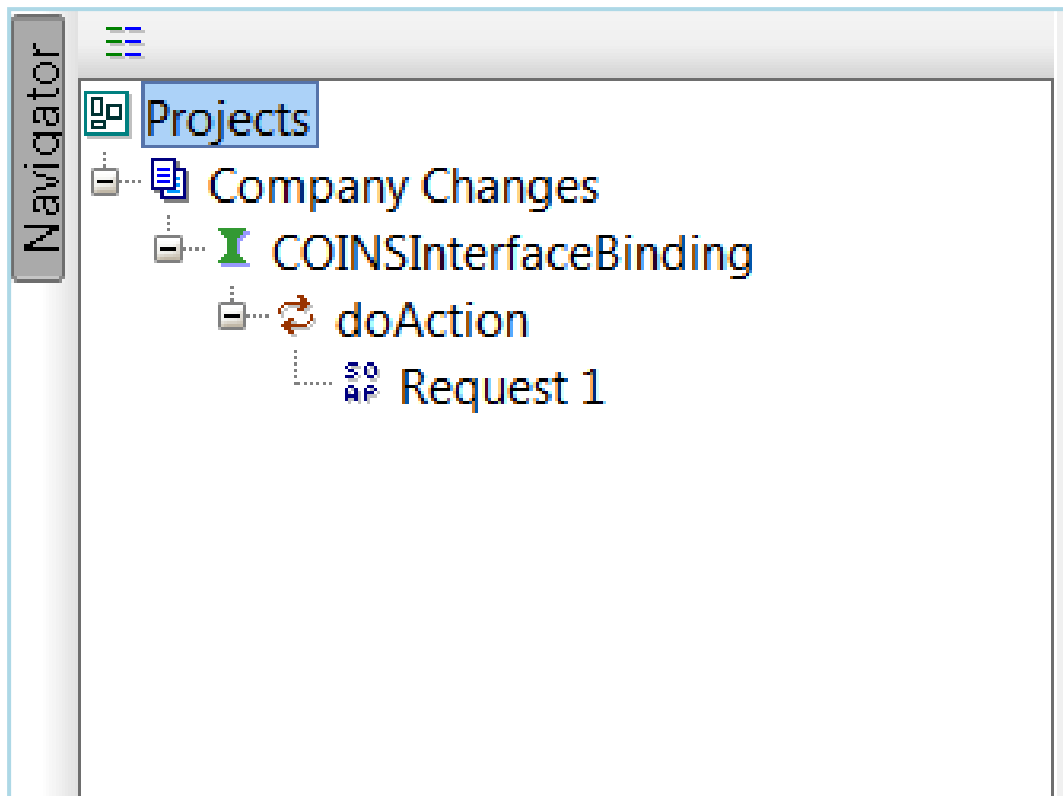
The 'New SOAP Project' dialog box is shown. It has a title bar with a green icon and a close button. The main area is titled 'New SOAP Project' and contains the subtitle 'Creates a WSDL/SOAP based Project in this workspace'. There are four input fields: 'Project Name' (empty), 'Initial WSDL' (empty), 'Create Requests' (checked), and 'Create TestSuite' (unchecked). There are also two checkboxes: 'Relative Paths' (unchecked) and 'Stores all file paths in project relatively to project file (requires save)' (unchecked). A 'Browse...' button is next to the 'Initial WSDL' field. At the bottom are 'OK' and 'Cancel' buttons.

Paste the shortcut contents into the Initial WSDL: field. Change the Project Name to something more meaningful.



The 'New SOAP Project' dialog box is shown again, but now with the 'Project Name' field filled with 'Company Changes' and the 'Initial WSDL' field filled with 'puesb.p?service=JCESB008&info=WSDL&namespace=false'. The 'Create Requests' checkbox is still checked, and the 'Create TestSuite' checkbox is still unchecked. The 'Relative Paths' checkbox is still unchecked. The 'Browse...' button is still next to the 'Initial WSDL' field. At the bottom are 'OK' and 'Cancel' buttons.

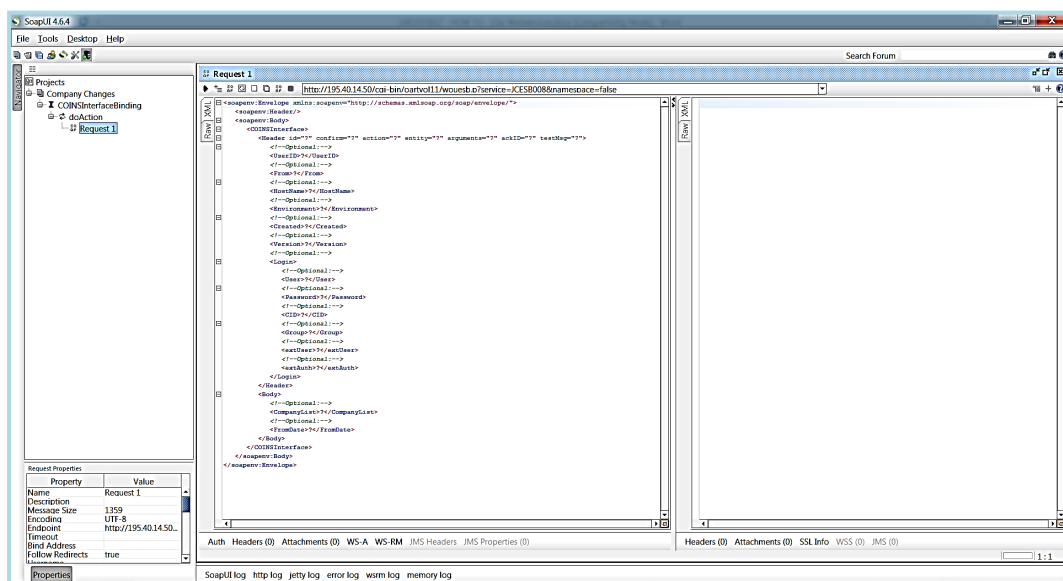
Click OK.



The application will examine the contents of the XML file from the shortcut and will build the necessary interactions.

If successful, the navigator panel should display a result similar to that shown here.

Double click on Request 1 to process the XML and build the appropriate message format.





The message details (shown below left) are the same as those generated as Sample Output in the Services Browse (shown below right). The only difference is the version in the soapUI is more verbose with details such as optional fields indicated.

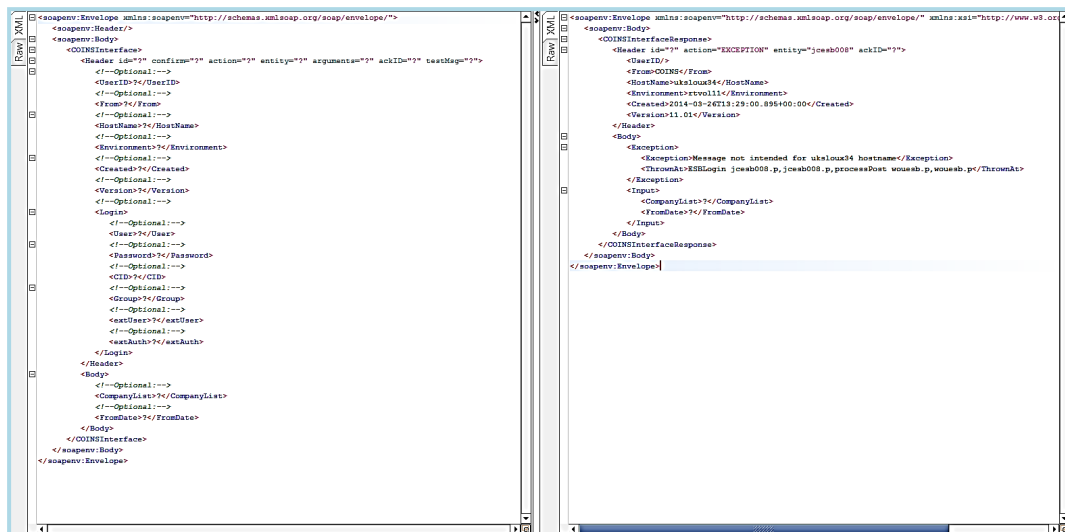
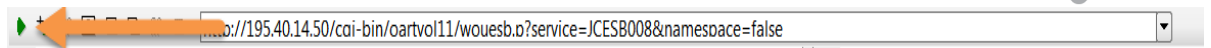


Sample

```
<COINSInterface>
  <Header id="" confirm="" action="" entity="" arguments="" ackID="" testMsg="true">
    <UserID></UserID>
    <From></From>
    <HostName></HostName>
    <Environment></Environment>
    <Created>2014-03-26T12:58:31.743+00:00</Created>
    <Version></Version>
    <Login>
      <User></User>
      <Password></Password>
      <CID>1</CID>
      <Group></Group>
      <extUser></extUser>
      <extAuth></extAuth>
    </Login>
  </Header>
  <Body>
    <CompanyList></CompanyList>
    <FromDate>2014-03-26</FromDate>
  </Body>
</COINSInterface>
```

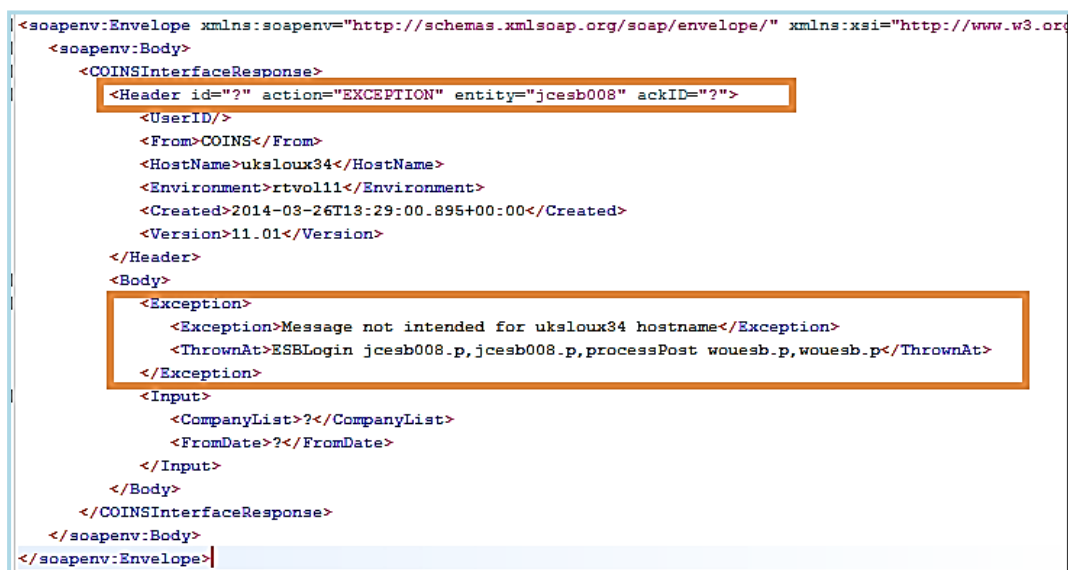
```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <COINSInterface>
      <Header id="" confirm="" action="" entity="" arguments="" ackID="" testMsg="">
        <!--Optional:-->
        <UserID?</UserID>
        <!--Optional:-->
        <From?</From>
        <!--Optional:-->
        <HostName?</HostName>
        <!--Optional:-->
        <Environment?</Environment>
        <!--Optional:-->
        <Created?</Created>
        <!--Optional:-->
        <Version?</Version>
        <!--Optional:-->
        <Login>
          <!--Optional:-->
          <User?</User>
          <!--Optional:-->
          <Password?</Password>
          <!--Optional:-->
          <CID?</CID>
          <!--Optional:-->
          <Group?</Group>
          <!--Optional:-->
          <extUser?</extUser>
          <!--Optional:-->
          <extAuth?</extAuth>
        </Login>
      </Header>
      <Body>
        <!--Optional:-->
        <CompanyList?</CompanyList>
        <!--Optional:-->
        <FromDate?</FromDate>
      </Body>
    </COINSInterface>
  </soapenv:Body>
</soapenv:Envelope>
```

To submit the request message, click the green arrow to the left of the toolbar



The message will be sent to the server and the server will issue an appropriate response - shown in the right hand panel.

Since we sent an empty message in the first instance, the request will fail as shown below:



The exception has been generated because the COINS server expects to be told which server and which environment the message is intended for. A request sent to the wrong server will fail. This is a protection measure to ensure that you do not issue requests to a LIVE environment during testing, or to a TEST environment when you go LIVE.

In our message, therefore, we need to fill in the destination details:



```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <COINSInterface>
      <Header id="?" confirm="?" action="?" entity="?" arguments="?" ackID="?" testMsg="?">
        <!--Optional:-->
        <UserID?></UserID>
        <!--Optional:-->
        <From?></From>
        <!--Optional:-->
        <HostName>dev.coins-global.com</HostName>
        <!--Optional:-->
        <Environment>dev</Environment>
        <!--Optional:-->
        <Created?></Created>
        <!--Optional:-->
        <Login>
          <!--Optional:-->
          <User?></User>
          <!--Optional:-->
          <Password?></Password>
          <!--Optional:-->
          <CID?></CID>
          <!--Optional:-->
          <Group?></Group>
          <!--Optional:-->
          <extUser?></extUser>
          <!--Optional:-->
          <extAuth?></extAuth>
        </Login>
      </Header>
      <Body>
        <!--Optional:-->
        <CompanyList?></CompanyList>
        <!--Optional:-->
        <FromDate?></FromDate>
      </Body>
    </COINSInterface>
  </soapenv:Body>
</soapenv:Envelope>
```

Once you have made the changes, send the message again.

This time the request should go further, but will fail again due to invalid username/password.



```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <COINSInterfaceResponse>
      <Header id="?" ackID="?" action="EXCEPTION" entity="jcesb008">
        <UserID/>
        <From>COINS</From>
        <HostName>dev.coins-global.com</HostName>
        <Environment>dev</Environment>
        <Created>2014-03-26T13:52:40.566+00:00</Created>
      </Header>
      <Body>
        <Exception>
          <Exception>Invalid User/Password [SY703]</Exception>
          <Inrownat>ssblogin jcesb008.p,jcesb008.p,processpost wouesb.p,wouesb.p</Inrownat>
        </Exception>
        <Input>
          <CompanyList?</CompanyList>
          <FromDate?</FromDate>
        </Input>
      </Body>
    </COINSInterfaceResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Once again, this is a security measure and needs to be specified within the request.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <COINSInterface>
      <Header id="?" confirm="?" action="?" entity="?" arguments="?" ackID="?" testMsg="?">
        <!--Optional:-->
        <UserID?</UserID>
        <!--Optional:-->
        <From?</From>
        <!--Optional:-->
        <HostName>dev.coins-global.com</HostName>
        <!--Optional:-->
        <Environment>dev</Environment>
        <!--Optional:-->
        <Created?</Created>
        <!--Optional:-->
        <Login>
          <!--Optional:-->
          <User>niglon</User>
          <!--Optional:-->
          <Password>abc1123</Password>
          <!--Optional:-->
          <CID?</CID>
          <!--Optional:-->
          <Group?</Group>
          <!--Optional:-->
          <extUser?</extUser>
          <!--Optional:-->
          <extAuth?</extAuth>
        </Login>
      </Header>
      <Body>
        <!--Optional:-->
        <CompanyList?</CompanyList>
        <!--Optional:-->
        <FromDate?</FromDate>
      </Body>
    </COINSInterface>
  </soapenv:Body>
</soapenv:Envelope>
```



Whilst we are making some more changes, we can remove the lines for Group, extUser and extAuth as these are not generally used.

Send the request again, and this time you should get a RESPONSE message instead of an EXCEPTION.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <COINSInterfaceResponse>
      <Header id="?" ackID="?" action="RESPONSE" entity="jcesb008">
        <UserID>NIGLON</UserID>
        <From>COINS</From>
        <HostName>dev.coins-global.com</HostName>
        <Environment>dev</Environment>
        <Created>2014-03-26T14:07:37.837+00:00</Created>
      </Header>
      <Body></Body>
    </COINSInterfaceResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

This indicates that we now have a correct message format.

However, whilst we got a response, the response came back with no data. This is because the service we are using as an example requests details of changes made to COINS company records since a certain date. So we need to add one more change to our request and specify, in the Body entry, which company and from which date. For example:

```
<Body>
  <!--Optional:-->
  <CompanyList>*</CompanyList>
  <!--Optional:-->
  <FromDate>01/01/12</FromDate>
</Body>
</COINSInterface>
</soapenv:Body>
</soapenv:Envelope>
```

Sending the request again now returns another RESPONSE request and some data.



```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <COINSInterfaceResponse>
      <Header id="?" ackID="?" action="RESPONSE" entity="jcesb008">
        <UserID>NIGLON</UserID>
        <From>COINS</From>
        <HostName>dev.coins-global.com</HostName>
        <Environment>dev</Environment>
        <Created>2014-03-26T14:13:21.464+00:00</Created>
      </Header>
      <Body>
        <co_configRow>
          <kco>1</kco>
          <coc_name>COINS</coc_name>
          <coc_lastrev>2013-05-09</coc_lastrev>
          <coc_vatregno>578524893</coc_vatregno>
          <coc_add1>12 The COINS Budfilding</coc_add1>
          <coc_add2>12 Tdfhe Grove</coc_add2>
          <coc_add3>Slough</coc_add3>
          <coc_add4>Berkshire</coc_add4>
          <coc_pcode>SL1 1QP</coc_pcode>
          <coc_phone>01753 501000</coc_phone>
          <coc_fax>01753 711010</coc_fax>
          <coc_reg><P>1234567890 324567832</P></coc_reg>
          <cnt_code>GB</cnt_code>
          <cur_code>GBP</cur_code>
          <coc_contref>607 1440 9</coc_contref>
          <coc_coinsid>2004-08-19T12:15:390x00079242</coc_coinsid>
        </co_configRow>
        <co_configRow>
          <kco>2</kco>
          <coc_name>COINS Limited</coc_name>
          <coc_lastrev>2013-02-02</coc_lastrev>
          <coc_vatregno>578524893</coc_vatregno>
          <coc_add1>The COINS Building</coc_add1>
          <coc_add2>12 The Grove</coc_add2>
          <coc_add3>Slough</coc_add3>
          <coc_add4>Berkshire</coc_add4>
          <coc_pcode>SL1 1QP</coc_pcode>
          <coc_phone>01753 711000</coc_phone>
          <coc_fax>01753 711010</coc_fax>
          <coc_reg/>
          <cnt_code/>
          <cur_code>GBP</cur_code>
          <coc_contref>607 1440 9</coc_contref>
          <coc_coinsid>2005-02-22T16:01:390x00079243</coc_coinsid>
        </co_configRow>
      </Body>
    </COINSInterfaceResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

We have now proved that:

- the Services are working
- we can configure the messages appropriately
- we can call them and get a response

Having established these points, we can confidently move onto other aspects of using the services knowing that we have a properly configured and working system.



4 ESB COINS User

In the previous section, we had to specify a User and Password in the message in order to obtain a correct response.

Depending on your requirements, you may wish to have a single user account dedicated to running services. To configure this, Navigate to System/System Setup/System Parameters.

Company	ID	Description	Value	Required?	History
	ESBMON	ESB monitor error email		<input type="checkbox"/>	
	ESBPORT	ESB send Port		<input type="checkbox"/>	
	ESBPWD	ESB Password	*****	<input type="checkbox"/>	
	ESBREPLY	Reply Queue		<input type="checkbox"/>	Hist
	ESBRESP	ESB response ID reference		<input type="checkbox"/>	
	ESBSEND	ESB send message script		<input type="checkbox"/>	
	ESBSQLNS	Name server connection parameters for SQLDB messages		<input type="checkbox"/>	
	ESBUID	ESB COINS User	sysadmin	<input type="checkbox"/>	Hist
	ESBURL	ESB send URL		<input type="checkbox"/>	
	ESBUSER	ESB User		<input type="checkbox"/>	

Value: sysadmin

Notes:

Regions:

- ☒ UK
- ☒ US
- ☒ Australia
- ☒ Ireland
- ☒ Canada
- ☒ Middle East
- ☒ Malaysia

Interfaces:

- ☒ COINS Plus
- ☒ OA
- ☒ GUI
- ☒ Interface

Parameter ESBUID allows you to specify a user id (licenced to use Web Services) that will be used to run a service if no other userid and password are specified. Whilst we have used it in our example, it is recommended that you do NOT use sysadmin as the userid but instead set up an account dedicated for running web services.

In our previous example, if we now remove the <Login> section, containing our original username and password, from our message....



```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <COINSInterface>
      <Header id="?" confirm="?" action="?" entity="?" arguments="?" ackID="?" testMsg="?">
        <!--Optional:-->
        <UserID?</UserID>
        <!--Optional:-->
        <From?</From>
        <!--Optional:-->
        <HostName>dev.coins-global.com</HostName>
        <!--Optional:-->
        <Environment>dev</Environment>
        <!--Optional:-->
        <Created?</Created>
        <!--Optional:-->
        <Login>
          <!--Optional:-->
          <User>niglon</User>
          <!--Optional:-->
          <Password>niglon</Password>
          <!--Optional:-->
          <CID?</CID>
        </Login>
      </Header>
    </Body>
  </COINSInterface>
</soapenv:Body>
</soapenv:Envelope>
```

....and resubmit the request, the RESPONSE message is returned with the account from the system parameter returned as the authenticated user.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <COINSInterfaceResponse>
      <Header id="?" ackID="?" action="RESPONSE" entity="jcesb008">
        <UserID>SYSADMIN</UserID>
        <From>COINS</From>
        <HostName>dev.coins-global.com</HostName>
        <Environment>dev</Environment>
        <Created>2014-03-26T15:05:45.180+00:00</Created>
      </Header>
      <Body>
        <co_configRow>
          <kco>1</kco>
          <coc_name>COINS</coc_name>
          <coc_lastrev>2013-05-09</coc_lastrev>
          <coc_vatregno>578524893</coc_vatregno>
          <coc_add1>12 The COINS Budfilding</coc_add1>
          <coc_add2>12 Tdfhe Grove</coc_add2>
        </co_configRow>
      </Body>
    </COINSInterfaceResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

4 soapUI

The examples given in this document use the Open Source tool soapUI to demonstrate the interaction of the Web Services. It is assumed that clients working with Web Services will have their own tools and applications, but if you wish to follow



the examples directly, soapUI can be downloaded from.:

<http://www.soapui.org>



5 Licencing

If you encounter any messages regarding licencing whilst using Web Services, check the following:

5.1 Additional Licencing

User Defined Services require an additional element of licencing.

In Module/Licence Maintenance, the Role %SYESB must be present

Role	Description	Users	Concurrent Users	User Defined Functions
<input type="checkbox"/> %STESB1	Stock Web Services	999	999	999
<input type="checkbox"/> %SYESB	Designer Services	999	999	999
<input type="checkbox"/> %SYESB0	System Services	999	999	999
<input type="checkbox"/> %VPESB1	VAP Integration Services	999	999	999
<input type="checkbox"/> %VPMOB1	mTick	999	999	999
<input type="checkbox"/> %VPSITE	Site Production	0	0	0
<input type="checkbox"/> %XESB	All Custom Services	0	0	0
<input type="checkbox"/> ALL	ALL Services	0	0	0
<input type="checkbox"/> CMFIN	Commercial Financials	0	0	0
<input type="checkbox"/> DBTEST	Don's Test Role	0	0	0
<input type="checkbox"/> ESB	All ESB Services	0	0	0
<input type="checkbox"/> ESS	Employee Self Service ESS	0	0	0

Within this, the functions %WXSYESB003/004/005 and 007 should be assigned:



010 - Training Contractors QA ↓ System >> Role Summary

Role %SYESB
Description Designer Services

Role Definition **Function List** User Defined Users

Function	Description
<input type="checkbox"/> %WXSYESB003	Generic Dataset
<input type="checkbox"/> %WXSYESB004	Generic Page
<input type="checkbox"/> %WXSYESB005	Generic Program
<input type="checkbox"/> %WXSYESB007	Field Get/Set
<input type="checkbox"/> %WXSYESB016	RSP Service

OPEN NEXT

Once you have checked that these are in place. Under the module/Licence Maintenance Extras Tab, there are four entries that define which user-defined services can be used:

010 - Training Contractors QA ↓ System >> Module/Licence Maintenance

Licence Modules Roles **Extras** Load Licence

Extra	Value
<input type="checkbox"/> X-ADAPT	999
<input type="checkbox"/> X-ASITE	999
<input type="checkbox"/> X-CECAT	999
<input type="checkbox"/> X-DATASERVER	50
<input type="checkbox"/> X-DESUSR	*
<input type="checkbox"/> X-ESBDA	100
<input type="checkbox"/> X-ESBPA	100
<input type="checkbox"/> X-ESBPR	100
<input type="checkbox"/> X-ESBTABLES	*
<input type="checkbox"/> X-EXCHANGE	Y
<input type="checkbox"/> X-MIN	20
<input type="checkbox"/> X-NWF	100
<input type="checkbox"/> X-OAMOD	*
<input type="checkbox"/> X-PCCL	999
<input type="checkbox"/> X-SELFXML	999

SHOW ALL + ADD COPY OPEN BULK DELETE

Search Extra

OPEN

X-ESBDA	controls how many datasets can be created within user defined services.
X-ESBPA	controls the number of pages that can be built
X-ESBPR	controls how many user defined programs
X-ESBTABLES	controls how many GET-SET services may be defined.



5.2 Roles

Web Services are licenced using Roles. These may be found within System Setup/Module Licence Maintenance. The Users column will indicate the number of users who may call the service - typically these will be named users.

Role	Description	Users	Concurrent Users	User Defined Functions
%HSEB1	HS ESB Plot Integration Services	999	999	999
%HSEB2	HS ESB PX Integration Services	999	999	999
%HSPDA	HS PDA Services	999	999	999
%JC001	Daily Reports	999	999	999
%JCCOM	Contract Status Commercial Role	0	0	0
%JCENQ	Contract Enquiries and Reports	999	999	999
%JCESB1	JC Integration Services	999	999	999
%JCEXC	Executive Reports and Enquires	0	0	0
%JCMQTY	mQty	999	999	999
%LAESB1	Land Appraisal Services	999	999	999
%M1X	M1x Data Request	0	0	0
%MAILADDIN	Mail Add In	0	0	0
%MK001	Mobile CRM App	0	0	0
%P2PAPP	PL Invoice Approval (Enquiry) Role	999	999	999
%PCEB1	Plant Web Service Dispatch and Return	999	999	999

Each role needs to be set up to give access to specific service functions

Function	Description
%WXJCESB001	[Jc Contract] Section Maintenance
%WXJCESB002	Project/Contract/Site Create
%WXJCESB003	[Jc Contract] Reconciliation
%WXJCESB004	Costs-Revenue-Debtors-Creditors
%WXJCESB005	Cost Transaction Load
%WXJCESB006	Revenue Transaction Load
%WXJCESB007	[JC Contract] Changes
%WXJCESB008	[Kco Company] Changes
%WXJCESB009	[JC Contract] Cost Transactions
%WXJCESB010	[Jc Contract] Section Maintenance
%WXJCESB017	Timesheet Master Data
%WXJCESB018	Cost Code List
%WXSCEB005	SC Certificate Transactions
%WXSCEB006	SC-Draft Certificate Create





5.3 User Maintenance

To grant access to the required service, open the User record within System/User Maintenance and select the Groups Tab.

010 - Training Contractors QA ↓ System Update User

User: niglon Nigel Longley

User: niglon Last Login: 17/01/17 10:13:53

Name: Nigel Longley

Details Groups Printing COINSplus Preferences Wiki

Roles

- ☐ All ESB Services (0)
- ☒ All Services
- ☐ ALL Services (0)
- ☐ BIM Services
- ☐ CIW Integration Services
- ☐ Commercial Financials (0)
- ☐ Concur Expenses
- ☐ Contract Enquiries and Reports
- ☐ Customer Care Services
- ☐ Customer Portal
- ☐ CVR Integration Services
- ☐ Daily Reports
- ☐ Designer Services
- ☐ DM Code Free Integration
- ☐ DM Integration Services
- ☐ Don's Test Role (0)
- ☐ Employee Self Service
- ☐ Employee Self Service - Holidays
- ☐ Employee Self Service ESS (0)
- ☐ Financial Integration Services
- ☐ FM Client Integration
- ☐ FM PDA Services
- ☐ FM WinSims Integration Services
- ☐ Health & Safety Audits
- ☐ HS ESB Plot Integration Services
- ☐ HS ESB PX Integration Services
- ☐ HS PDA Services
- ☒ JC Integration Services
- ☐ Land Appraisal Services
- ☐ mCare
- ☐ mFM
- ☐ mGRN
- ☐ Mobile Technician
- ☐ mQty
- ☐ mSite
- ☐ mTick
- ☐ PL Invoice Approval (Enquiry) Role
- ☒ Plant Web Service Dispatch and Return
- ☐ Requisition Creation
- ☐ Respond to RFI
- ☐ Service Mobile
- ☐ Stock Web Services
- ☐ Subcontractor Portal
- ☐ System Services
- ☐ VAP Integration Services

AUDIT UNDO SAVE

Tick the appropriate Roles - subject to the licensing granted to your company and click Save.



6 User Defined Services

Where a COINS Webservice does not exist that meets your requirements, User Defined Services, as their name implies, allow Web Services to be configured to access data to your specification.

User defined services come in several forms:

- Data Sets - These allow you full control of the information being retrieved from COINS
- Pages - Simple maintenance of data within COINS, such as hold codes on invoices.
- Programs - Calculation programs, such as those used in Workflow, which can be run from Web Services
- GET/SET - A service which allows you to GET information from any table and SET information in any table (subject to system security and Business Logic rules).

6.1 Datasets

Three components are required for Datasets. A function, a page and the dataset itself.

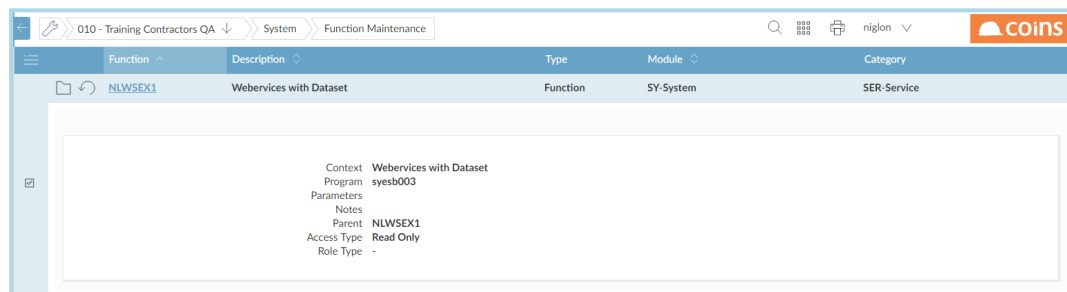


It is important that all three components are defined with the same name.

In this example, we will use the name NLWSEX1 and we will create a dataset that will retrieve specified Supplier Records from the Supplier Master File.

6.1.1 Function

Create the function called NLWSEX1 with Category of SER - Service and Program of syesb003. The Context of the function will determine the description that will appear later on the Web Service menu.



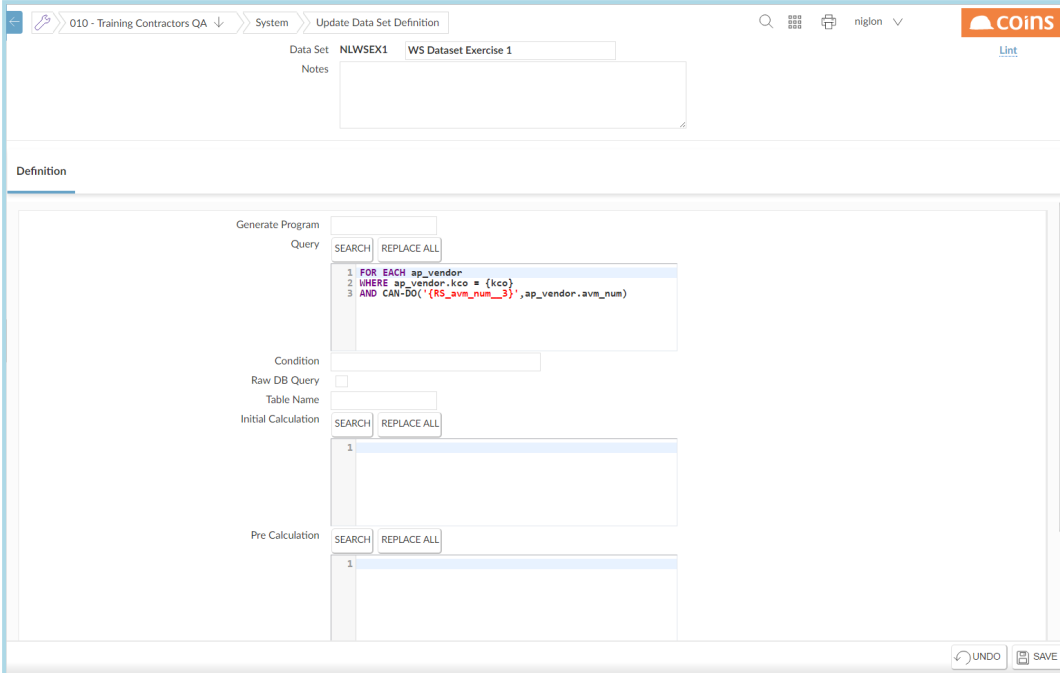
6.1.2 Dataset

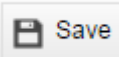
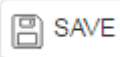
The dataset defines the output message of the webservice.

Create a dataset called NLWSEX1 with the Query:

```
FOR EACH ap_vendor
WHERE ap_vendor.kco = {kco}
AND CAN-DO('{RS_avm_num__3}',ap_vendor.avm_num)
```

This query will allow us to specify the COINS Company and a list of required Suppliers Accounts. These selections will be passed to the dataset by the page which we will create later.



Click  Save  SAVE

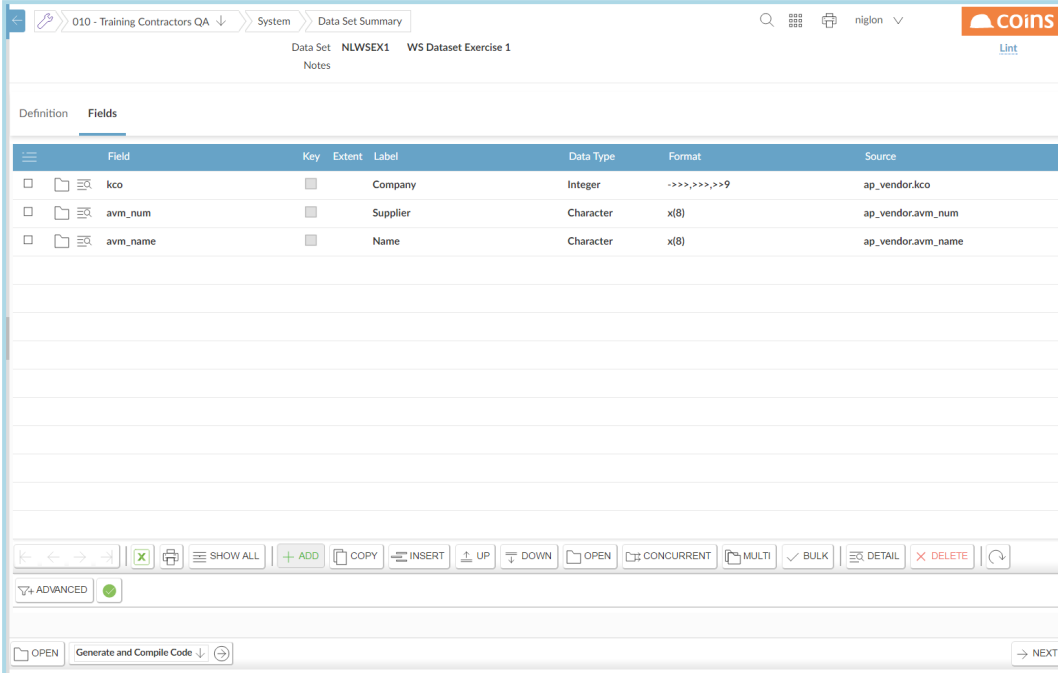
Add the fields

kco

avm_num

avm_name

The dataset is constructed in the same way as any OA Designer dataset, so fields can be calculated fields as well as taken directly from the COINS database.

Field	Key	Extent	Label	Data Type	Format	Source
kco			Company	Integer	>>>>>>>9	ap_vendor.kco
avm_num			Supplier	Character	x(8)	ap_vendor.avm_num
avm_name			Name	Character	x(8)	ap_vendor.avm_name

6.1.3 Page

The page design controls the fields that the web service will expect in order for it to perform its function.

In this example we need the web service to be passed the COINS Company (kco) and the list of required Supplier Accounts (RS_avm_num__3).



For Web Services, the Page design consists only of a Page called the same name as the function and the Dataset, and the required fields defined on an UPDATE form.

Field names can be followed by (XX) where XX is either IN,DE,DA,CH,LO meaning Integer, Decimal, Date, Character or Logical data type respectively. **If omitted then character data type is assumed.**



The page defines the INPUT schema for the BODY section of the message.

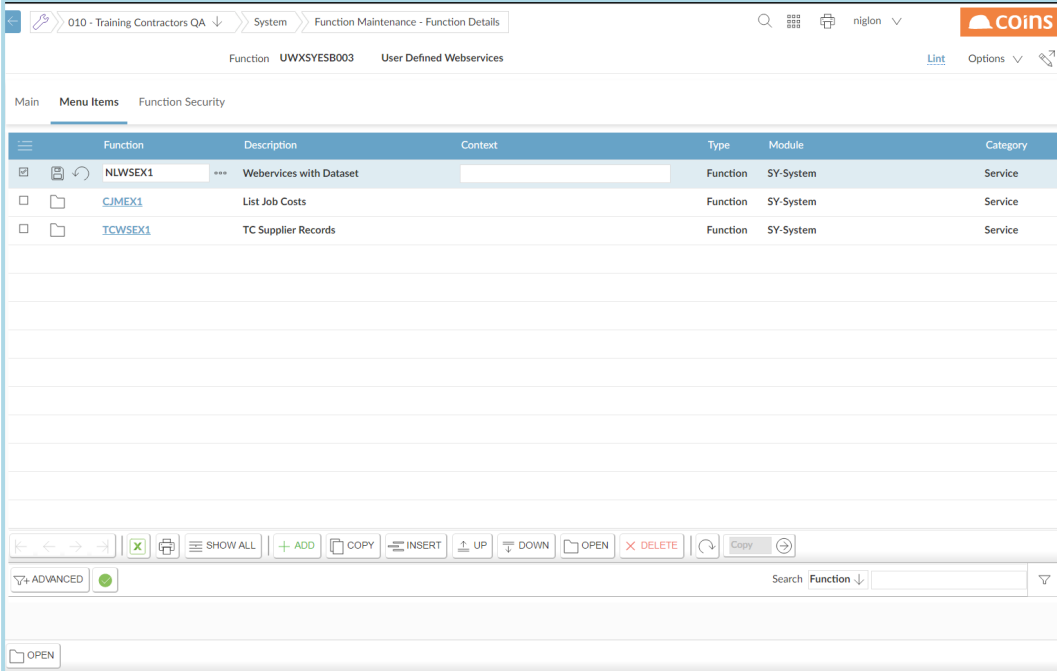
6.1.4 Adding the Function to the Web services Menu

To run the Dataset Web Service, first check that the menu function UWXSYESB003 exists. If it does not exist, then you need to create it.


Open the Function and select the Menu Items Tab. If the function already exists, there may already be entries on this tab relating to Standard Used Defined Services issued by COINS (prefixed with %) or other User Defined Services created by your company.



Click **Add** and enter the name of your new User Defined Services Function (in our example NLWSEX1).

Function	Description	Context	Type	Module	Category
<input checked="" type="checkbox"/> NLWSEX1	Webervices with Dataset		Function	SY-System	Service
<input type="checkbox"/> CJMEX1	List Job Costs		Function	SY-System	Service
<input type="checkbox"/> TCWSEX1	TC Supplier Records		Function	SY-System	Service

Click . The full details of the function are displayed.

Navigate back to the COINS Services Menu and select the SY - System Hyperlink.

COINS Services

Service	Description
CB	Cash Book
EC	eCommerce
FM	Facilities Management
GL	General Ledger
HS	House Sales
HR	Human Resources
JC	Contract Status
PL	Purchase Ledger
SC	Subcontract Ledger
SL	Sales Ledger
CS	Contract Sales Ledger
SY	System
VF	VAT
PC	Plant
DM	Document Management
PO	Procurement
CI	Company Information Workbench
CC	Customer Care
LA	Land Appraisal
CV	CVR
MK	Marketing
MB	Mobile Applications
SE	SE
ST	Stock

Locate the SYESB003 entries. Your new Function will be in this group.



COINS Services

COINS Services >System

Service	Description
SYESB000	Confirmation Message
SYESB001	Login
SYESB002	Get Waiting Message Numbers
SYESB003	Coxies ESB Service 001
SYESB003	GLBAL Dataset
SYESB003	JTS Data Set
SYESB003	Activities Data Set
SYESB003	Get jc_emprate for employee
SYESB003	BIM Projects (All)
SYESB003	BIM Projects (One)
SYESB003	BIM PM Issue - Type
SYESB003	BIM PM Issue - Impact
SYESB003	BIM PM Issue - Importance
SYESB003	BIM PM Issue - Status
SYESB003	BIM PM Issues (Navis Views)
SYESB003	BIM PM Project Team Personnel
SYESB003	BIM PM Project Team Contacts
SYESB003	BIM PM RFI (Navis Views)
SYESB003	BIM PM RFI - Status
SYESB003	BIM PM RFI - Type
SYESB003	BIM PM Issues
SYESB003	BIM PM RFI
SYESB003	Coxies Designer Webservice Dataset
SYESB003	Supplier Records
SYESB004	My Page
SYESB004	Timesheet Operator

Select the Hyperlink to view the service schema.

COINS Services

COINS Services >System >Generic Dataset

Service	Description
SYESB003 (NLWSEX1)	Generic Dataset

Schema

Schema	WSDL	WSDL NS
input.xsd output.xsd exception.xsd acknowledgement.xsd SOAPInput.xsd SOAPOutput.xsd	SYESB003.wsdl	SYESB003.wsdl

The generic dataset service allows a designer to build input parameters to a dataset and request the data from the dataset using the service.

The argument to this service is the function name of the function/page/dataset to be called.

Input

Entity	Type	Documentation
COINSInterface		
Header		
id	string	A unique message identifier from the originating system.
confirm	string	Whether a confirmation message is required. Set to true or yes to have a confirmation message returned. If this is set then an id must also be provided.
action	string	An action type that will indicate why the message was created. For example, this might be CREATE, UPDATE, or DELETE for a message as a result of a maintenance of a record or PUBLISH for the publish of some information.
entity	string	The service that should consume the message.
arguments	string	Arguments that should be passed to the service.

The Input Schema defined by the page will look something like this:

Sample

<COINSInterface>

<Header id="" confirm="" action="" entity="" arguments="" ackID="" testMsg="true">

<UserID></UserID>

<From></From>

<HostName></HostName>

<Environment></Environment>

<Created>2014-03-28T09:05:40.967+00:00</Created>

<Login>



```

<User></User>
<Password></Password>
<CID>1</CID>
<Group></Group>
<extUser></extUser>
<extAuth></extAuth>
</Login>
</Header>
<Body>
  <Results>1</Results>
  <kco>99</kco>
  <RS_avm_num__3></RS_avm_num__3>
</Body>
</COINSInterface>

```

And the Output Schema, defined by the Dataset will look like this:

Sample

```

<COINSInterface>
  <Header id="" confirm="" action="" entity="" ackID="">
    <UserID></UserID>
    <From></From>
    <HostName></HostName>
    <Environment></Environment>
    <Created>2014-03-28T09:05:41.194+00:00</Created>
  </Header>
  <Body>
    <ttRow>
      <kco>99</kco>
      <avm_num></avm_num>
      <avm_name></avm_name>
    </ttRow>
  </Body>
</COINSInterface>

```

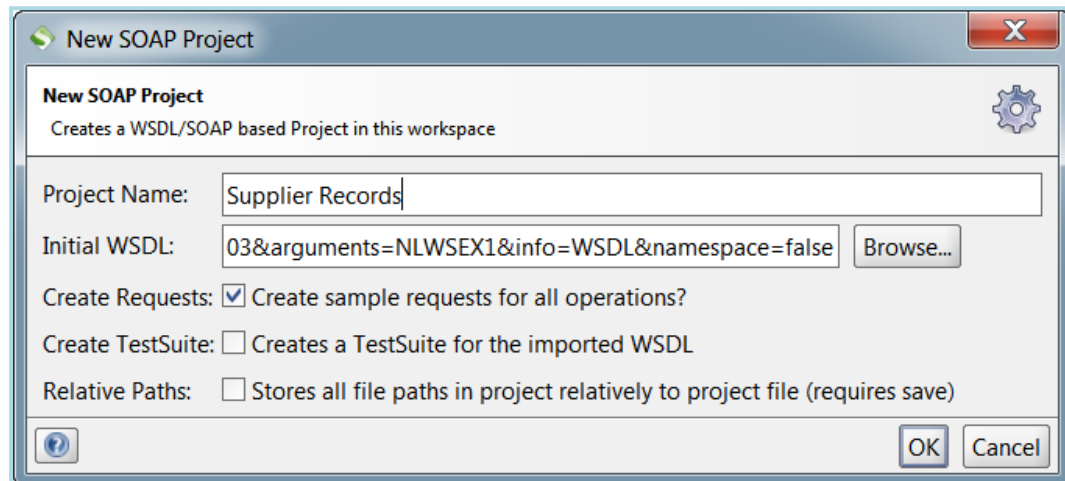



```
</ttRow>
```

```
</Body>
```

```
</COINSInterface>
```

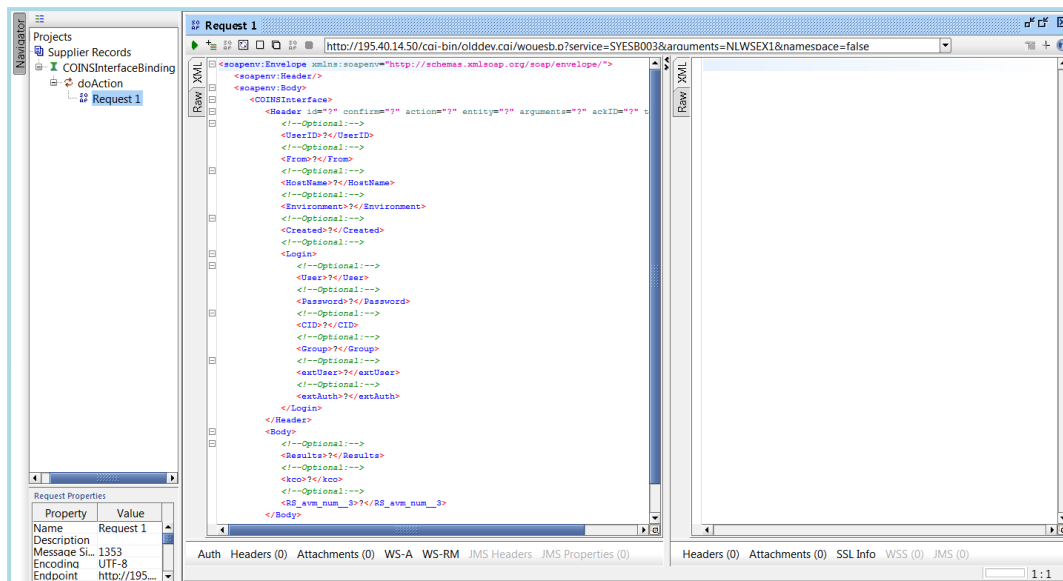
6.1.5 Testing the Service



Create a new soapUI project and from the Service Schema, copy the WSDL shortcut link into the Initial WSDL field

Click OK.

Once the Project has been created. Run Request 1 in the editor (Double-click the Request 1 entry)



As before, we can use the input message as it is. However, this time we are going to replace the input message with the one generated in the Service Schema as it is a simpler form. Clear all the text in the left hand panel.

From the Service schema, scroll to the sample input message:

RS_avm_num_3

string

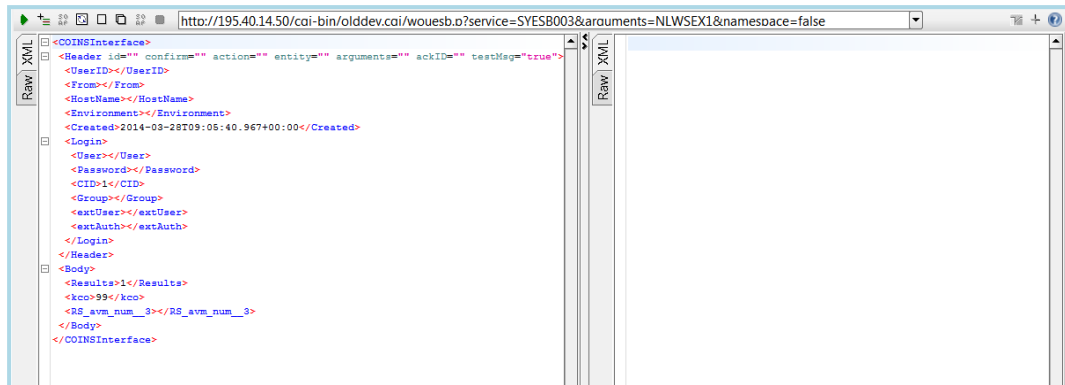
Sample

```
<COINSInterface>
  <Header id="" confirm="" action="" entity="" arguments="" ackID="" testMsg="true">
    <UserID></UserID>
    <From></From>
    <HostName></HostName>
    <Environment></Environment>
    <Created>2014-03-28T09:05:40.967+00:00</Created>
  </Header>
  <Login>
    <User></User>
    <Password></Password>
    <CID>1</CID>
    <Group></Group>
    <extUser></extUser>
    <extAuth></extAuth>
  </Login>
  <Body>
    <Results>1</Results>
    <kco>99</kco>
    <RS_avm_num_3></RS_avm_num_3>
  </Body>
</COINSInterface>
```

Output

Entity	Type	
COINSInterface		

Copy the text from <COINSInterface> to </COINSInterface> and paste into the soapUI right hand panel.



Enter the hostname, Environment and username/password detail.

Delete the lines for Group, extUser and extAuth

In the <Body> section, specify how many results are to be returned, the Company number to retrieve the records from and a comma separated list of Supplier Account numbers to retrieve (or * for all). For example:

```
<COINSInterface>
<Header id="" confirm="" action="" entity="" arguments="" ackID="" testMsg=""
  <UserID></UserID>
  <From></From>
  <HostName>dev.coins-global.com</HostName>
  <Environment>dev</Environment>
  <Created>2014-03-28T09:05:40.967+00:00</Created>
  <Login>
    <User>niglon</User>
    <Password>niglon</Password>
    <CID>1</CID>
  </Login>
</Header>
<Body>
  <Results>10</Results>
  <kco>1</kco>
  <RS_avm_num__3>*</RS_avm_num__3>
</Body>
</COINSInterface>
```

Send the message. Correct any exceptions if they occur.

If all is correct, a RESPONSE message should be returned along with the requested records:



```

<COINSInterface>
  <Header action="RESPONSE" entity="syeshb003">
    <UserID>NIGLON</UserID>
    <From>COINS</From>
    <HostName>dev.coins-global.com</HostName>
    <Environment>dev</Environment>
    <Created>2014-03-28T09:45:40.563+00:00</Created>
  </Header>
  <Body>
    <ttRow>
      <kco>1</kco>
      <avm_num>---002</avm_num>
      <avm_name>Callander Utilites (AleBak Tes</avm_name>
    </ttRow>
    <ttRow>
      <kco>1</kco>
      <avm_num>.CA001</avm_num>
      <avm_name>trete</avm_name>
    </ttRow>
    <ttRow>
      <kco>1</kco>
      <avm_num>.CA003</avm_num>
      <avm_name>Carter Inc 3</avm_name>
    </ttRow>
    <ttRow>
      <kco>1</kco>
      <avm_num>000001</avm_num>
      <avm_name>Andrews Sykes Patent Glazing</avm_name>
    </ttRow>
    <ttRow>
      <kco>1</kco>
      <avm_num>000002</avm_num>
      <avm_name>G I Neilsons</avm_name>
    </ttRow>
    <ttRow>
      <kco>1</kco>
      <avm_num>000003</avm_num>
      <avm_name>IARNROD EIREANN</avm_name>
    </ttRow>
  </Body>
</COINSInterface>

```

Each record contains the fields as defined by the dataset. To add additional fields, simply add them to the dataset definition and re-run the request. The Page definition only needs to be amended if additional selection criteria are needed in the Input message.

6.2 Pages

Within COINS, pages allow the maintenance of data through Add, Update and Delete functionality. This functionality can be replicated by Web Services.

The user-defined page maintenance webservice capability is intended to be used for straight-forward maintenance of setup tables etc. where there are relatively few fields or no fields with dependencies.

For areas such as supplier maintenance etc. there are hard-coded services for this type of functionality.



Before embarking on designing your own page maintenance webservice it is recommended you seek advice from COINS to check there is not already a service available that will fulfil your needs.

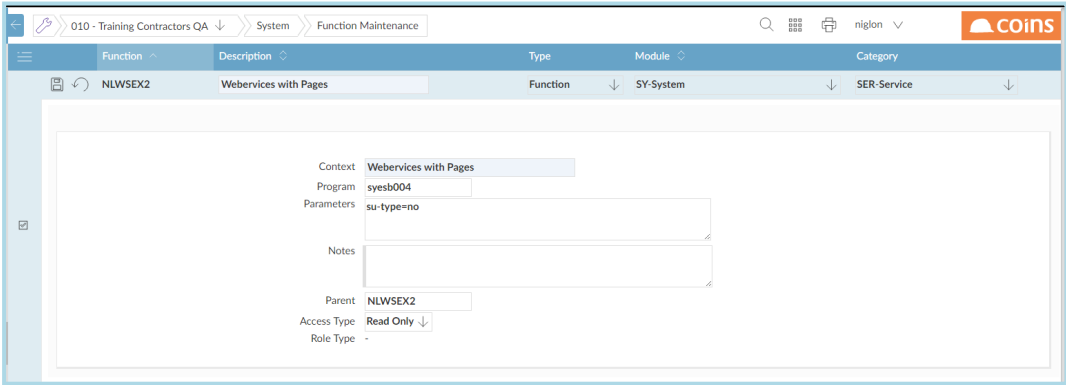
For a page webservice, two design components are required, the function(s) and a page. It is important that the function(s) and page have the same name.

In this example, we will use the name NLWSEX2 and we will design a page that will allow the maintenance of COINS UserID's.

6.2.1 Function

Create the function called NLWSEX2 with Category of SER - Service and program of syesb004. The Context of the function will determine the description that will appear later on the Web Services menu.

Since the table sysuser holds both groups and users, and we only want to access users for this example, we need to set a parameter on the function to ensure we maintain the correct records. Add the parameter su-type=no



6.2.2 Page

The page controls the record access, the Input and the Output messages. These are specified using the following forms:

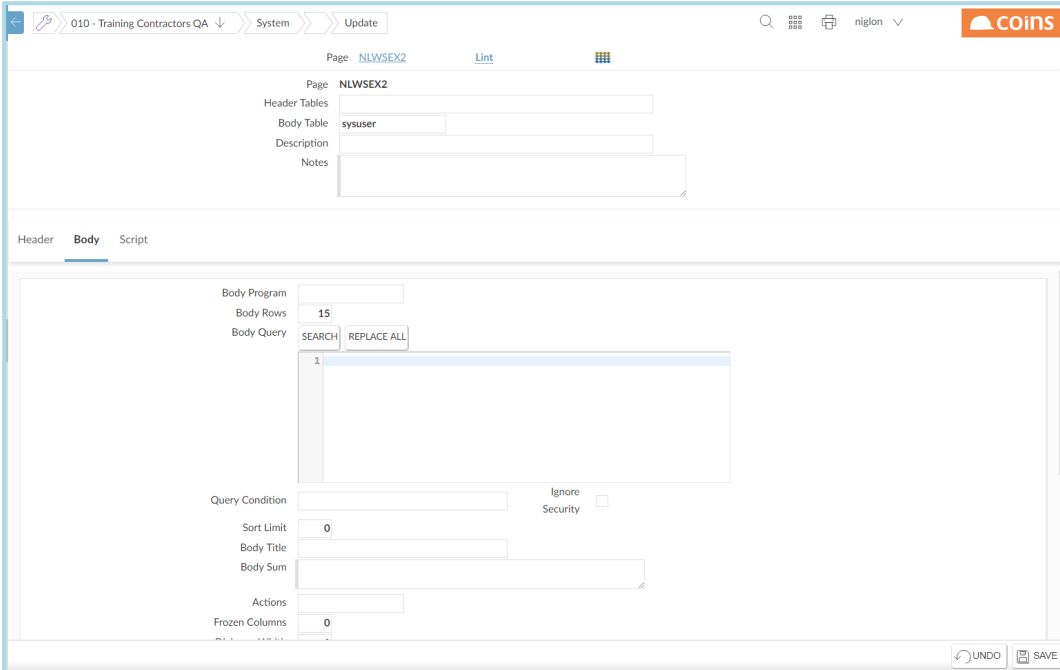
Body	Defines the fields that uniquely identify the record
Update	Defines the Input Schema
Detail	Defines the output Schema



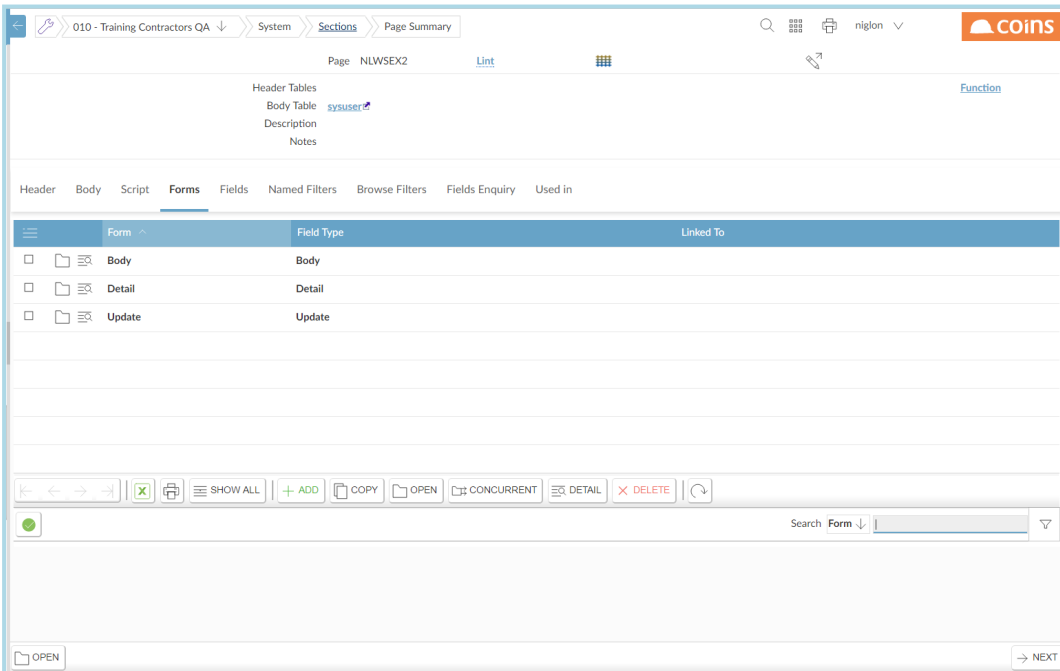
Create new page called NLWSEX2.



The body table should be set to the COINS database table that is to be maintained - in this example the table is sysuser.



Add the forms Body, Detail and Update



Form	Field Type	Linked To
Body	Body	
Detail	Detail	
Update	Update	



6.2.2.1 Body Form

The Body Form is used to define which fields will be used to uniquely identify the record when Adding, Updating or Deleting. For sysuser the key field is the userid (su-userid). Other tables may have more than one field as part of the key - all should be added to ensure the correct record is accessed.

Add field su-userid to the Body Form.



Make sure that the Add and Update fields are ticked to indicate that this is a key field.

6.2.2.2 Update Form

The update Form is used to define the Input Schema and should contain all the fields that will be maintained by the web service.

If records will be added by the Web service it is important to include all required fields (i.e. all fields that would be mandatory if added through a standard COINS page) as the business logic will not allow records to be Added without these fields.

For user records, the required fields are the User ID, the User Name and the Prime Company and a comma separated list of Companies the user can access. Our example web service will also maintain the Name User field. So our list of fields will be:

su-userid

su-name

sur-primeco



su-cos

su-gui

Add these fields to the Update Form.

010 - Training Contractors QA > System > Sections > Page Summary

Page NLWSEX2 [Lint](#)

Header Tables
Body Table [sysuser](#)
Description
Notes

Function

Header Body Script Forms **Fields** Named Filters Browse Filters Fields Enquiry Used in

	Field	Label	Width	Height	Function	Add	Upd	View As	Tab	Layout	Append	Hidden
<input type="checkbox"/>	su-userid	User	0	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>				<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	su-name	Name	0	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>				<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	sur_primeco	Prime Company	0	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>				<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	sur_cos	Companies	0	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>				<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	su-gui	Named User	0	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>				<input type="checkbox"/>	<input type="checkbox"/>

Transfer Fields

SHOW ALL + ADD COPY INSERT UP DOWN OPEN MULTI BULK DETAIL DELETE

ADVANCED

Search Field

Form Update View Grouped

OPEN NEXT

6.2.2.3Detail Form

The Detail Form defines the Output Schema and you may include any fields that you wish to see as confirmation of the record maintenance.

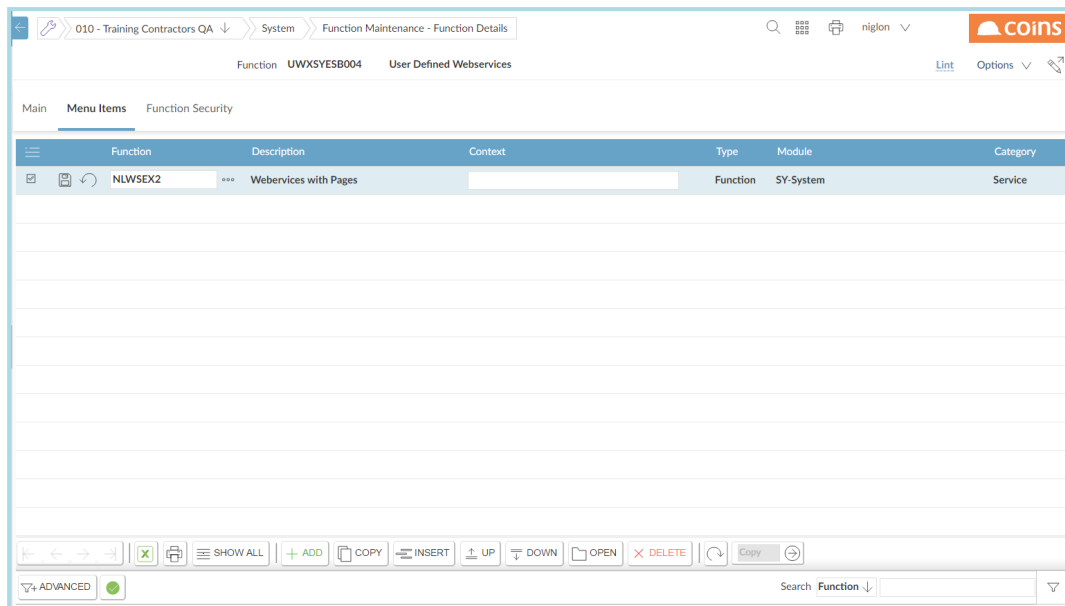


6.2.3 Adding the Function to the Web services Menu

To run the Page Web Service, first check that the menu function UWXSYESB004 exists. If it does not exist, then you need to create it.

Open the Function and select the Menu Items Tab. If the function already exists, there may already be entries on this tab relating to Standard Used Defined Services issued by COINS (prefixed with %) or other User Defined Services created by your company.

Click Add and enter the name of your new User Defined Services Function (in our example NLWSEX1).

Navigate back to the COINS Services Menu and select the SY - System Hyperlink.

COINS Services

Service	Description
CB	Cash Book
EC	eCommerce
FM	Facilities Management
GL	General Ledger
HS	House Sales
HR	Human Resources
JC	Contract Status
PL	Purchase Ledger
SC	Subcontract Ledger
SL	Sales Ledger
CS	Contract Sales Ledger
SY	System
VT	VAT
PC	Plant
DM	Document Management
PO	Procurement
CI	Company Information Workbench
CC	Customer Care
LA	Land Appraisal
CV	CVR
MK	Marketing
MB	Mobile Applications
SE	SE
ST	Stock

Locate the SYESB00 entries. Your new Function will be in this group.

COINS Services

COINS Services ~ System

Service	Description
SYLSB000	Confirmation Message
SYLSB001	Login
SYLSB002	Get Waiting Message Numbers
SYLSB003	Webervices with Database
SYLSB004	Webervices with Pages
SYLSB006	Web service for Appointments/Tasks
SYLSB007	Field Get/Set
SYLSB011	Save Completed Mobile Form
SYLSB012	Database Publish Recent Data
SYLSB013	Create User Mobile Configuration
SYLSB014	Complete Workflow Stage

Select the Hyperlink to view the service schema.

COINS Services

COINS Services>System>Generic Page

Service	SYESB004 (NLWSEX2)
Description	Generic Page
Schema	input and output and exception and acknowledgement and SOAPInput and SOAPOutput and
WSDL	SYESB004.wsdl
WSDL NS	SYESB004.wsdl

The generic page service allows a designer to build an OA page and call a bulk update using this service.

The argument to this service is the function name of the page to be called.

Input

Entity	Type	Documentation
COINSInterface		
Header		
id	string	A unique message identifier from the originating system.
confirm	string	Whether a confirmation message is required. Set to true or yes to have a confirmation message returned. If this is set then an id must also be provided.
action	string	An action type that will indicate why the message was created. For example, this might be CREATE, UPDATE, or DELETE for a message as a result of a maintenance of a record or PUBLISH for the publish of some information.
entity	string	The service that should consume the message.
arguments	string	Arguments that should be passed to the service.
ackID	string	An optional acknowledgement ID. If this ID is set then COINS will respond with a acknowledgement message returning this ackID in the message header.
testMsg	boolean	Whether this message is a test message. A test message will process through in exactly the same way that a normal message would except that at the point where it would normally commit the transaction to the database, the message is then backed out.
UserID	string	The user in the originating system that caused the message to be produced.
From	string	The name of the system that produced the message.
HostName	string	This is the name of the host system (on UNIX the HOSTNAME environment variable). This is automatically checked (if present in the message) and if it does not match the host name of the system then an exception message is created. If the HostName tag is missing then HostName checking is omitted.
Environment	string	This is the name of the COINS environment that is the intended target for the message. This is automatically checked based on the name of the instance of the COINS system if it does not match then an exception message is created.
Created	dateTime	Date/Time of the source system when the message was created.
Language	string	
User	string	The COINS user to be used to consume the message. If omitted then the user set in SY parameter ESBUID will be used. If user is specified then a password must also be given.
Pass	string	The password for the User specified above. The password can either be plain text or encoded to a 32 character hexadecimal string using an MD5 digest. For example the MD5 encoding for "The quick brown fox jumps over the lazy dog" is

The Input Schema - defined by the Update form of the page will look something like this:

[illegible]

The Output Schema - defined by the Detail Form of the page will look something like this:

```

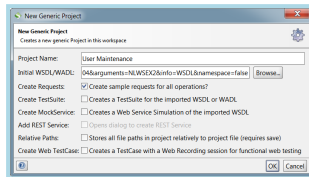
Sample

<COINSInterface>
<Header id="" confirm="" action="" entity="" requestId=""
  <UserId>/<UserId>
  <From>/<From>
  <HostName>/<HostName>
  <Environment>/<Environment>
  <Created>2014-04-08T14:02:18.055+01:00</Created>
</Header>
<Body>
  <sysuserRow rap_action="" id=""
    cru-userid</su-userid>
    cru-name</su-name>
  </sysuserRow>
</Body>
</COINSInterface>

```



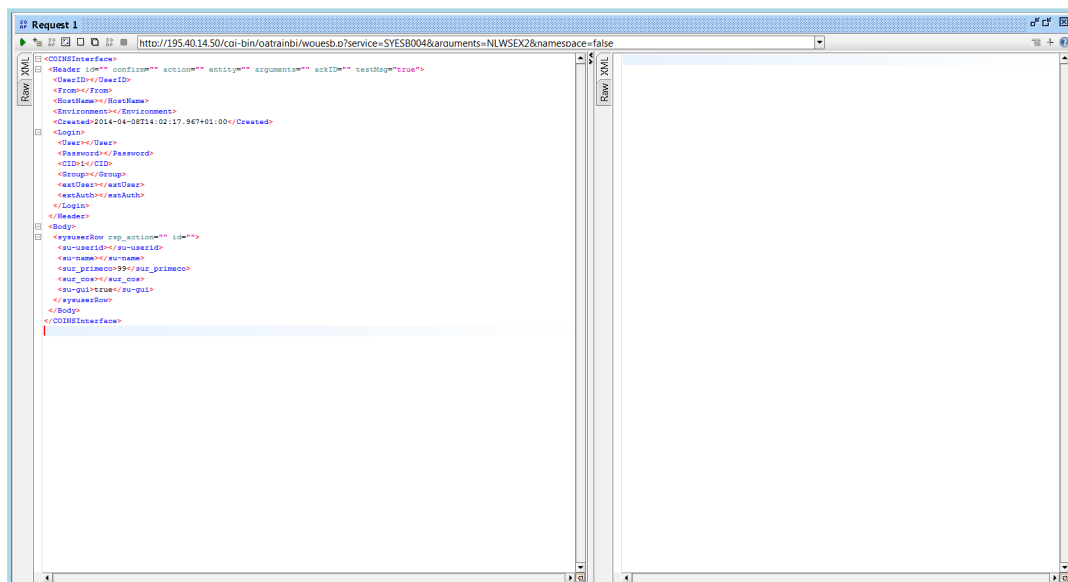
6.2.4 Testing the Service



Create a new soapUI project and from the Service Schema copy the WSDL shortcut link and paste this into the Initial WSDL field

Click OK

Once the project has been created, run Request 1 in the editor. Replace the Input message with the simplified one from the Input Schema from the Web service.



Enter the hostname, Environment and username/password details.

Delete the lines for Group, extUser and extAuth

In the <Body> Section you will notice a new line

```
<sysuserRow rsp_action="id" id="id">
```

This “Row” record must contain a valid rsp_action of one of the following:

- | | |
|---|--------|
| A | Add |
| U | Update |
| D | Delete |

The id field can contain an id record that will be useful for identifying issues when updating more than one record. We will use this later in the exercise.



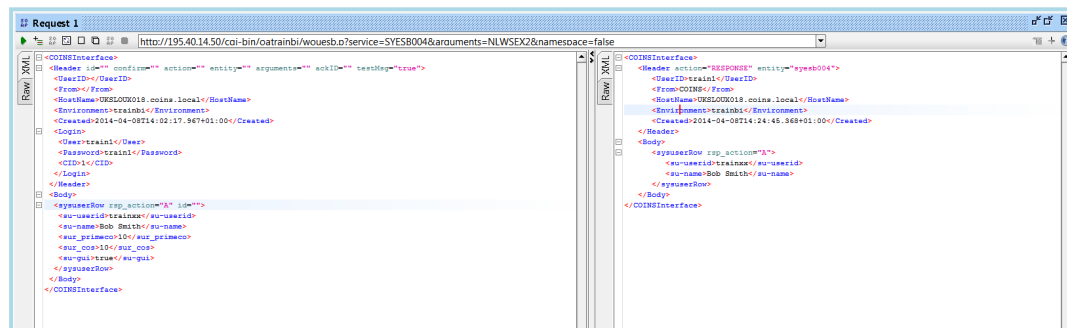
Specify an `rsp_action` of A and then fill in the appropriate user details into the remaining Body Fields.

For example:

```
<?xml version='1.0'?>
<COINSInterface>
  <Header id='confirm' action='add' entity='user' arguments='ackID=testMsg=true'>
    <UserID>UserID</UserID>
    <From>From</From>
    <Environment>COINSD019.coins.local</Environment>
    <Environment>trainbi</Environment>
    <Created>2014-04-08T14:02:17.967+01:00</Created>
    <Login>
      <User>trainbi</User>
      <Password>trainbi</Password>
    </Login>
    <Body>
      <rspuserRow rsp_action='A' id='1'>
        <su-userid>trainbi</su-userid>
        <su-name>Bob Smith</su-name>
        <su-primeco>10</su-primeco>
        <su_topid>10</su_topid>
        <su-gui>true</su-gui>
      </rspuserRow>
    </Body>
  </COINSInterface>
```

Send the message and correct any exceptions if they occur.

If all is correct, a RESPONSE message should be returned.



However, if we check the User records in User Maintenance, no new User has been created.

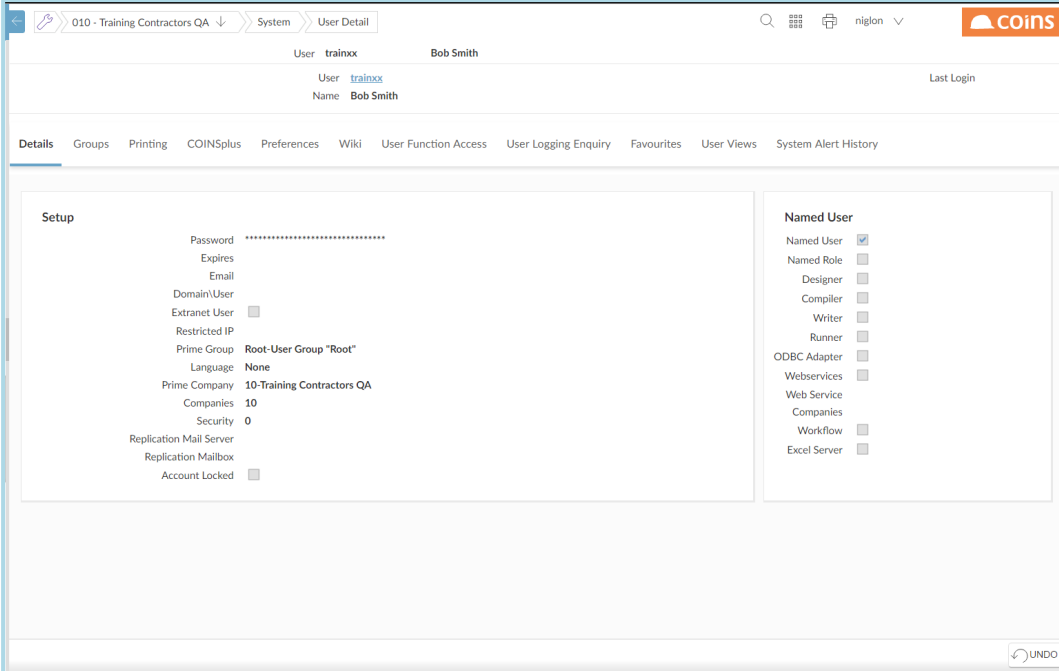
If we look at the Input Message again, at the top of the message is another new line:

```
<?xml version='1.0'?>
<COINSInterface>
  <Header id='confirm' action='add' entity='user' arguments='ackID=testMsg=true'>
```

If this is set to “true”, the details will be sent and validated but no record maintenance will take place. This allows us to test the message without affecting any data. Only when this is set to “false” will record maintenance take effect.

Set this to false and send the message again.

This time, when we check user maintenance, we should see a new user record:

The screenshot shows the 'User Detail' page for user 'trainxx' (Bob Smith). The page has a top navigation bar with '010 - Training Contractors QA' and 'System >> User Detail'. The user's name is 'Bob Smith' and the last login is shown. Below the navigation bar, there are tabs for 'Details', 'Groups', 'Printing', 'COINSplus', 'Preferences', 'Wiki', 'User Function Access', 'User Logging Enquiry', 'Favourites', 'User Views', and 'System Alert History'. The 'Details' tab is active, showing a 'Setup' section with fields for Password, Expires, Email, Domain\User, Extranet User, Restricted IP, Prime Group (Root-User Group "Root"), Language (None), Prime Company (10-Training Contractors QA), Companies (10), Security (0), Replication Mail Server, Replication Mailbox, and Account Locked. To the right, there is a 'Named User' section with a list of roles and checkboxes: Named User (checked), Named Role, Designer, Compiler, Writer, Runner, ODBC Adapter, Webservices, Web Service, Companies, Workflow, and Excel Server.

Experiment now with the `rsp_actions` of A, U and D to see the changes in the record and messages that appear in the output message. Try adding a record with the same userid for example or add a new user but leave the user name blank.

6.2.4.1 Using the ID field when adding multiple records

We can maintain more than one record at a time by duplicating the block of lines within the Body section of the Input Message. For example:



```
<Body>
  <sysuserRow rsp_action="A" id="record1">
    <su-userid>trainxx</su-userid>
    <su-name>Bob Smith</su-name>
    <sur_primeco>10</sur_primeco>
    <sur_cos>10</sur_cos>
    <su-gui>true</su-gui>
  <sysuserRow rsp_action="A" id="record2">
    <su-userid>trainxy</su-userid>
    <su-name>Barry Johns</su-name>
    <sur_primeco>10</sur_primeco>
    <sur_cos>10</sur_cos>
    <su-gui>true</su-gui>
  </sysuserRow>
</Body>
```

In the above example, in the id field of each block I have entered a unique reference for each record.

In the message I have specified an Add of two new records, but one of the user's already exists in the User records. When I then send the message one of the updates will fail. In the output message I will get an exception:

```
<COINSInterface>
  <Header action="EXCEPTION" entity="sysb004">
    <UserID>train</UserID>
    <From>COINS</From>
    <HostName>UKSLOUX018.coins.local</HostName>
    <Environment>trainbi</Environment>
    <Created>2014-04-08T14:37:11.304+01:00</Created>
  </Header>
  <Body>
    <Exception>
      <Exception>User already exists with that ID. [SY731] (Row:1 ID:record1)</Exception>
      <ThrownAt>preWrite sur-rsp.p,commit sur-rsp.p,commitRowUpdate sur-rsp.p,writeRecord sysb004.p</ThrownAt>
    </Exception>
    <Input>
      <sysuserRow id="record1" rsp_action="A">
        <su-userid>trainxx</su-userid>
        <su-name>Bob Smith</su-name>
        <sur_primeco>10</sur_primeco>
        <sur_cos>10</sur_cos>
        <su-gui>true</su-gui>
      <sysuserRow id="record2" rsp_action="A">
        <su-userid>trainxy</su-userid>
        <su-name>Barry Johns</su-name>
        <sur_primeco>10</sur_primeco>
        <sur_cos>10</sur_cos>
        <su-gui>true</su-gui>
      </sysuserRow>
    </Input>
  </Body>
</COINSInterface>
```



The exception will always give the row number that failed, but with an ID specified, the ID will also appear in the exception making it easier for me to identify the record that needs to be amended. Only when all records in the input message are correct will any changes be applied to the database.

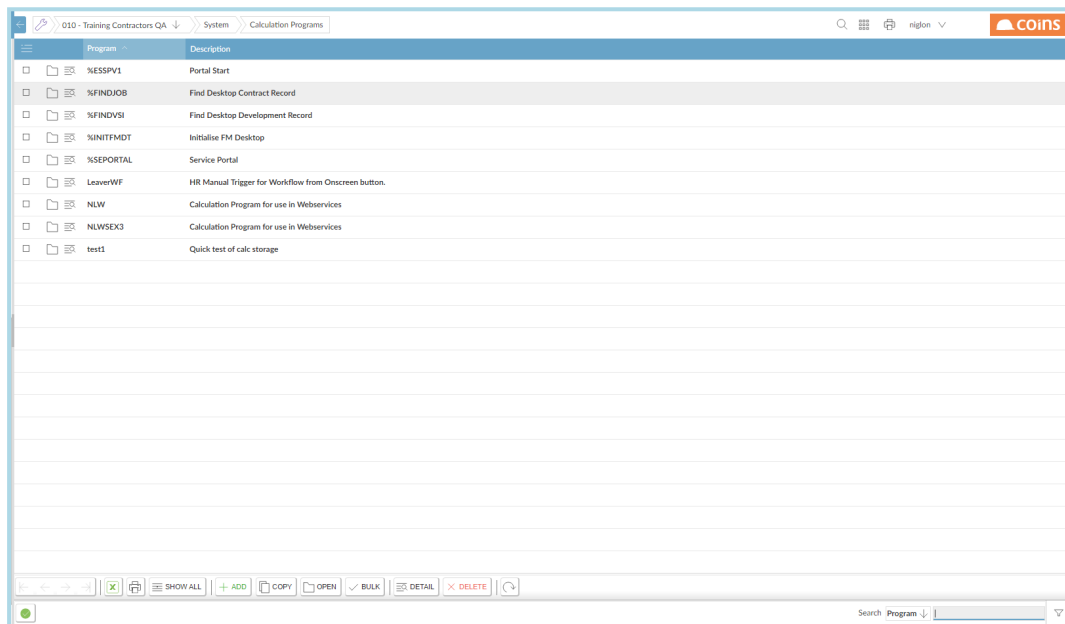


6.3 Calculation Programs

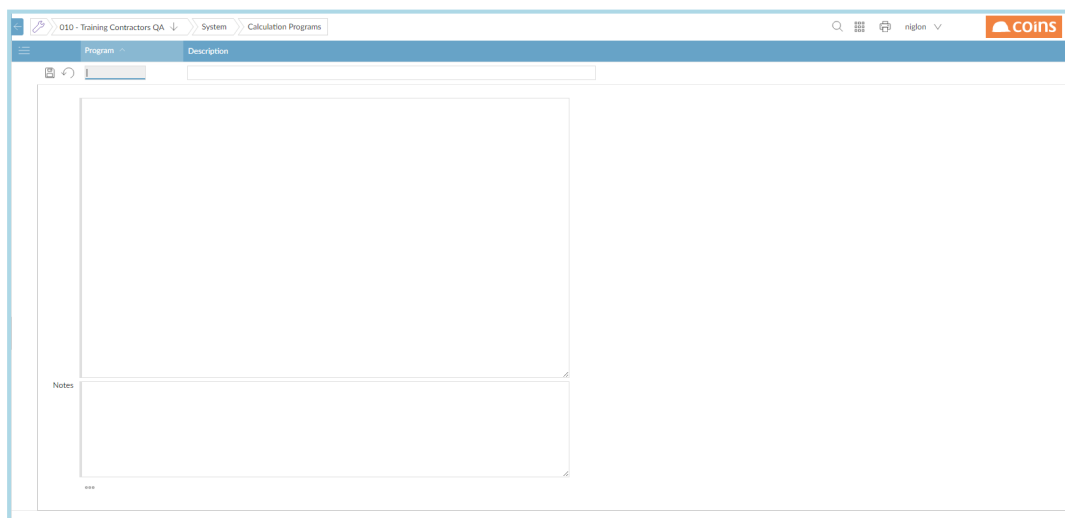
COINS provides the functionality for user-defined programs to be created and called by Web services offering greater flexibility for data manipulation and retrieval.

6.3.1 Defining the Calculation

Calculation Programs may be defined using the Calculation Program option under OA Reporting & BI Setups in the OA Reporting & BI Module.



To create a new program, click





Program	This is the name by which the calculation will be referenced. It is recommended that this name is meaningful enough for its purpose to be easily inferred when debugging in the future.
Description	Give a brief description of the purpose of the calculation here.

The next box is for the calculation. The calculation syntax used follows COINS OA standard syntax, with numerical variables for the various parameters. Each numerical variable is encased in curly braces e.g. {1}

For example, if we wanted a calculation to multiply two different numbers, we would write this as:

`{1} * {2}`

Once a variable is defined, it can be re-used in a calculation and therefore need only be passed into the calculation once.

e.g. `{1} * {1} * {1}`

Three examples of a calculation program definition are shown below:

Program	Description
num1	Multiply two numbers together
<div> <input checked="" type="checkbox"/> </div> <div> <code>{{1} * {2}}</code> </div> <div>Notes</div>	

Program	Description
datediff	Number of days between two dates
<div> <input checked="" type="checkbox"/> </div> <div> <code>datestring('{1}') - datestring('{2}')</code> </div> <div>Notes</div>	

Program	Description
cha1	Takes two variables and checks if they are the same
<div> <input checked="" type="checkbox"/> </div> <div> <code>\$IF\$(Inlist('{1}';{2}),eq,1,True,False')</code> </div> <div>Notes</div>	

The calculation code can run other user defined calculation programs in the same way as any other calculation.

If the program is run without passing the required {} parameters then they will be replaced in the calculation with blanks.

Variables defined before the calculation is run can be accessed. Variables defined or updated in the calculation program will be available following the call to the calculation program.

The Notes section allows a more detailed description of the calculation to be entered.

6.3.2 Testing Calculation Programs (run, run\$)

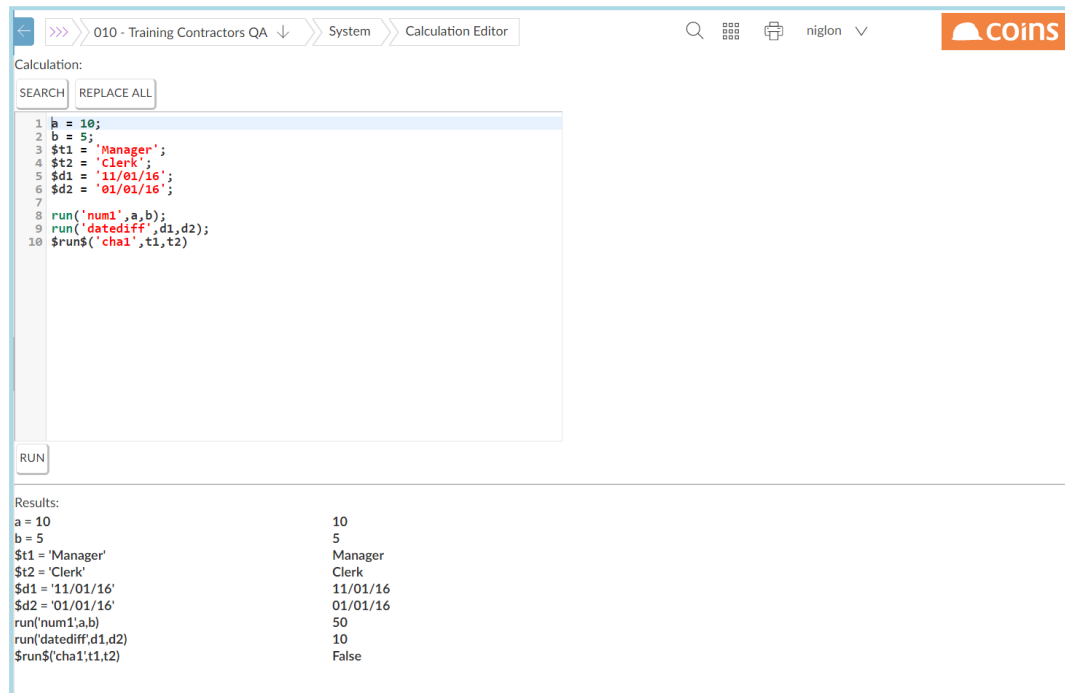
Once a calculation has been saved then it can be run by using:

run('calc',var1,var2) for numerical values

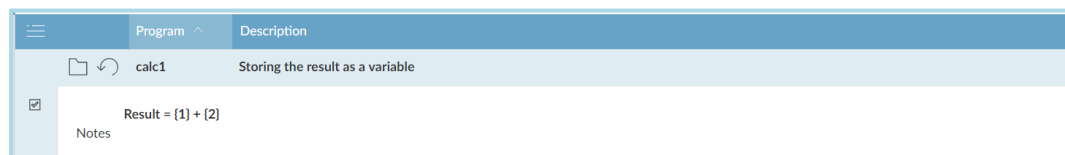
or

\$run\$('calc',var1,var2) for character values

Calculations may be tested in the Calculation editor prior to use to check they work as expected.



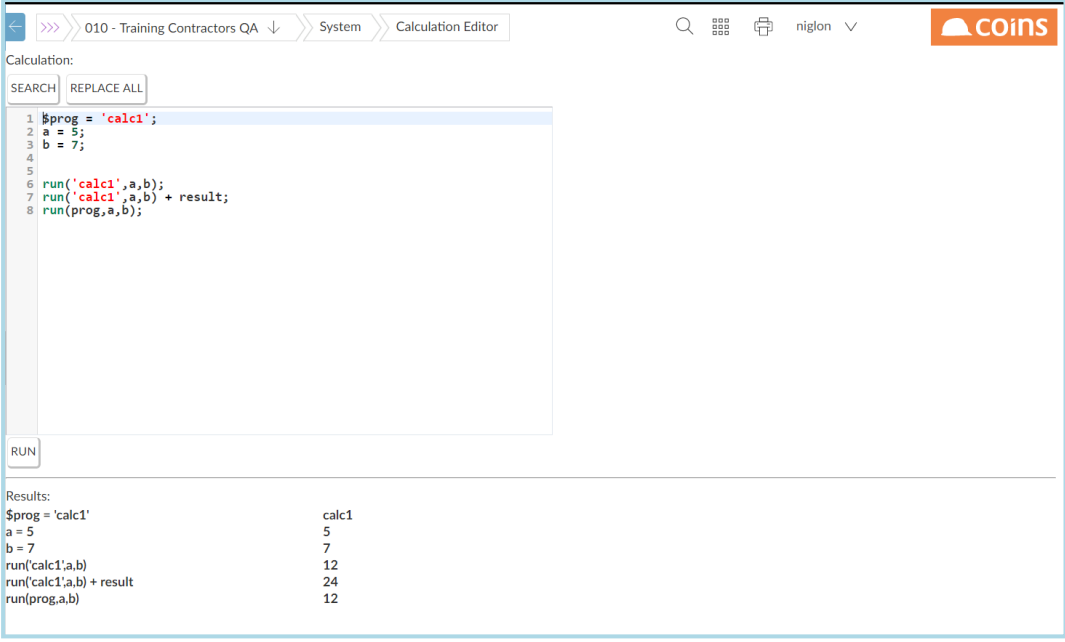
Calculation programs can be defined to both use and create variables. For example:



This will store the value of {1} + {2} in a variable called result. This can then be used directly in subsequent calculations.

The name of the Calculation program to be called may also be passed as a variable.

Examples of these are shown below:

010 - Training Contractors QA > System > Calculation Editor

Calculation:

SEARCH REPLACE ALL

```

1 $prog = 'calc1';
2 a = 5;
3 b = 7;
4
5
6 run('calc1',a,b);
7 run('calc1',a,b) + result;
8 run(prog,a,b);

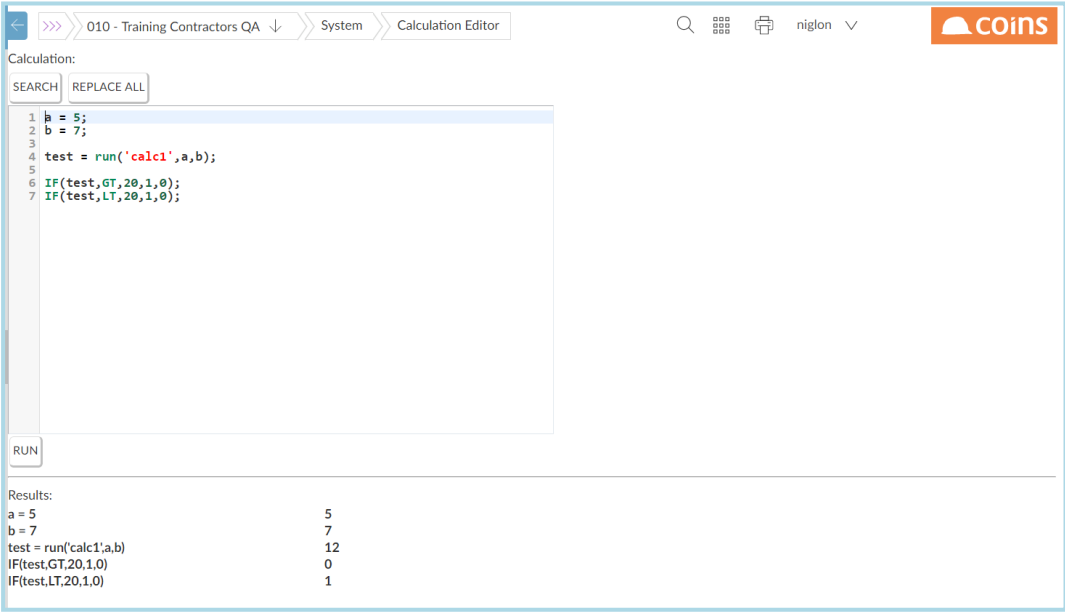
```

RUN

Results:

\$prog = 'calc1'	calc1
a = 5	5
b = 7	7
run('calc1',a,b)	12
run('calc1',a,b) + result	24
run(prog,a,b)	12

It is also possible to use calculation programs directly within other calculations:



010 - Training Contractors QA > System > Calculation Editor

Calculation:

SEARCH REPLACE ALL

```

1 a = 5;
2 b = 7;
3
4 test = run('calc1',a,b);
5
6 IF(test,GT,20,1,0);
7 IF(test,LT,20,1,0);

```

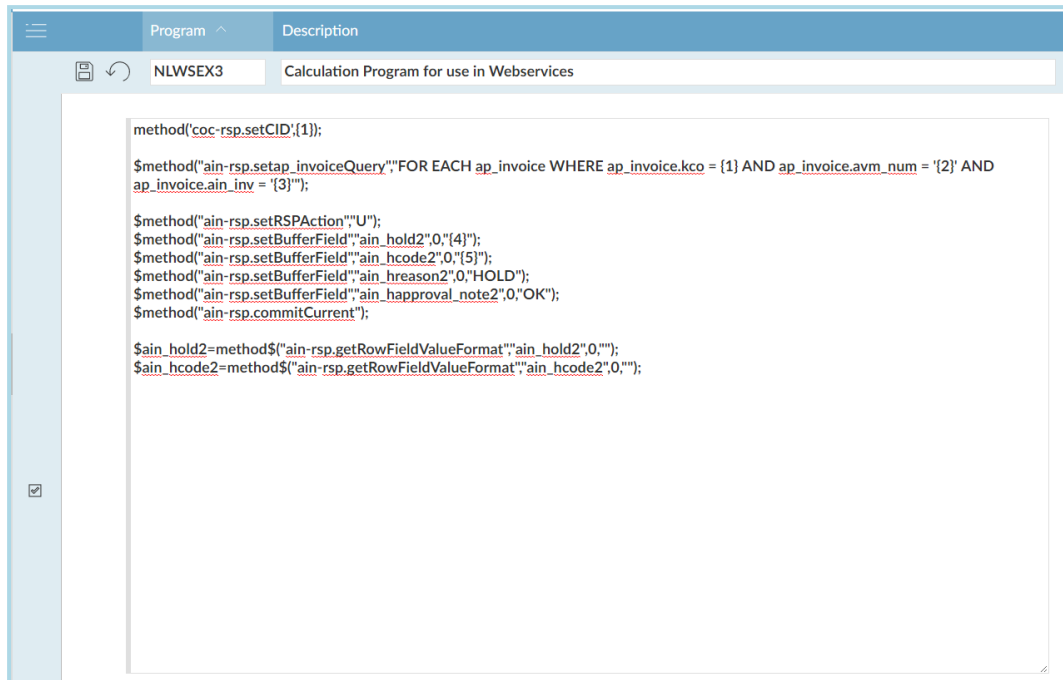
RUN

Results:

a = 5	5
b = 7	7
test = run('calc1',a,b)	12
IF(test,GT,20,1,0)	0
IF(test,LT,20,1,0)	1

6.3.3 Using Calculation Programs with Web Services

For this example, we are going to setup a calculation program to modify the hold code on an invoice. The program in this example is called NLWSEX3 and has been defined as follows:



The program will accept 5 input variables

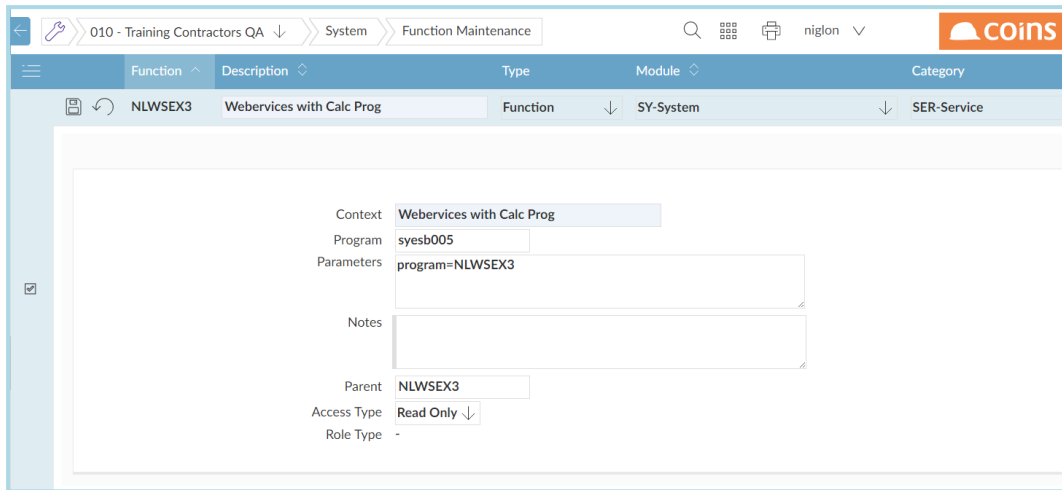
- {1} The COINS Company
- {2} The Supplier Number of the Invoice
- {3} The Invoice Number
- {4} The Hold flag (yes or no)
- {5} The hold code

The calculation will look up the appropriate invoice, set the hold flag and code, and return two variables which will be used in the output message as confirmation of the changes.

6.3.3.1 Create the Function

Create the function called NLWSEX3 with Category of SER - Service and program of syesb005. The Context of the function will determine the description that will appear later on the Web Services menu.

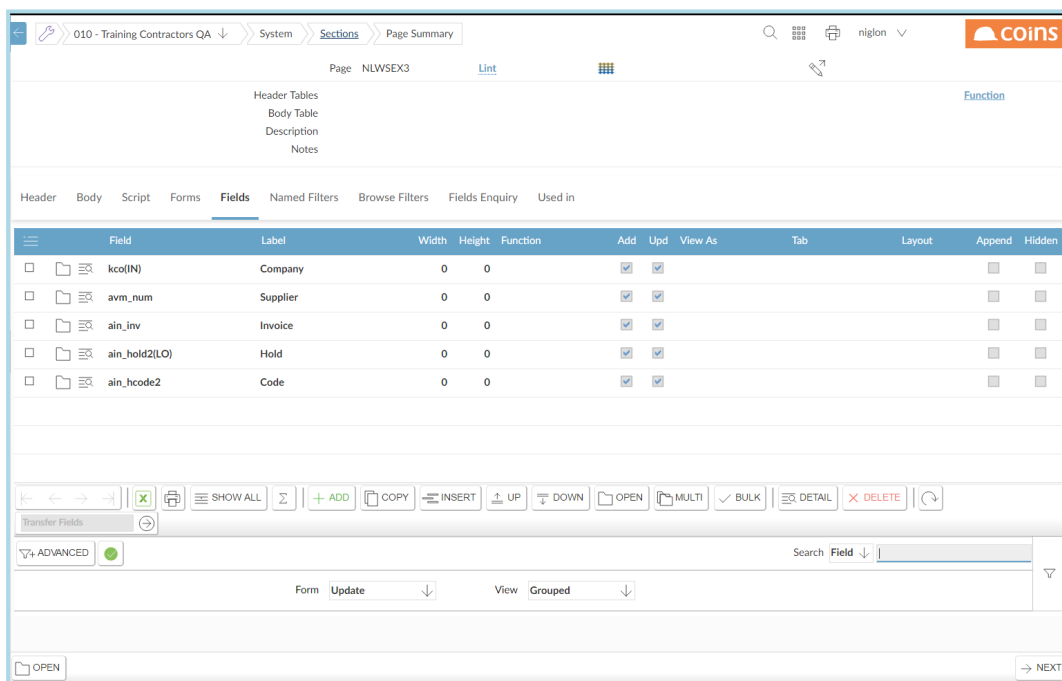
Add the parameter program=NLWSEX3

The program parameter allows you to specify the name of the calculation program to be used. Whilst it is not necessary to call the Calculation Programs the same name as the functions etc. it is probably less confusing if you do keep all the names consistent. The page and the function **MUST** have the same name.

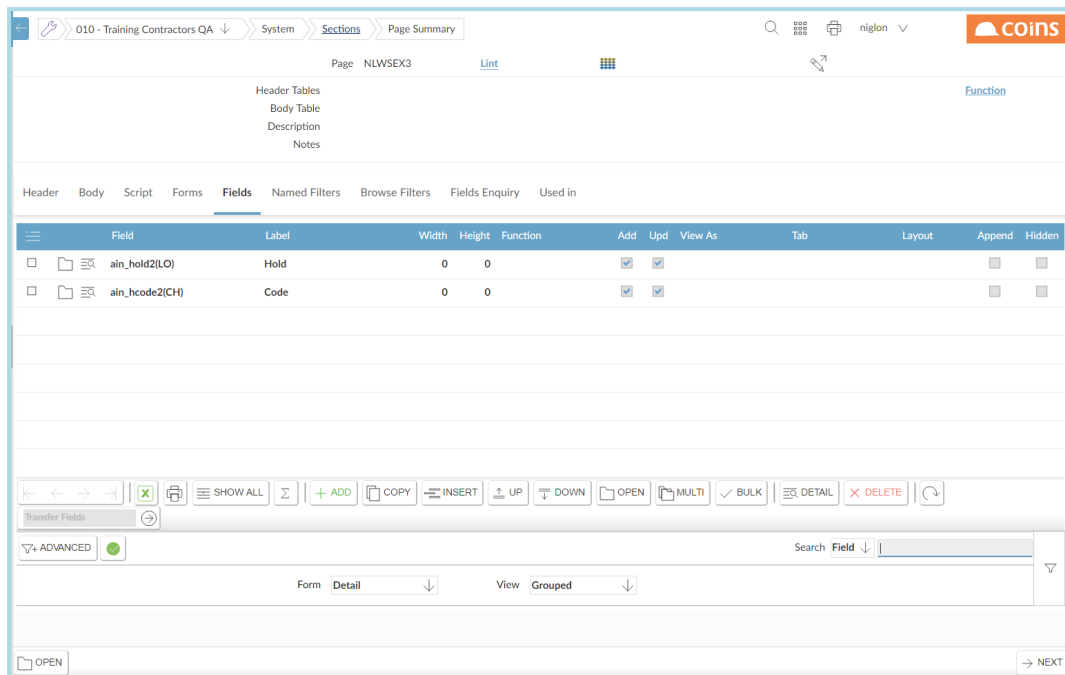
6.3.3.2 Create the Page

The Page will have two Forms. The Update Form will define the Input Message



The fields will relate to the input parameters in the calculation program and should be defined in the sequence they are used {1}=kco, {2}=avm_num etc...

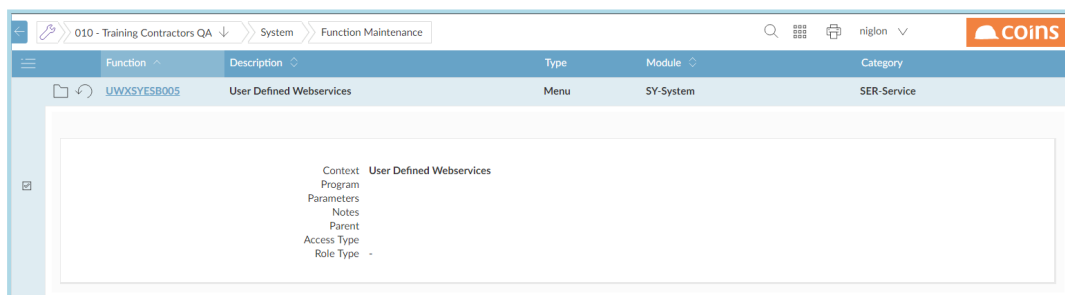
The Detail Form defines the Output message and will return the variables set by the Calculation Program.

Field	Label	Width	Height	Function	Add	Upd	View As	Tab	Layout	Append	Hidden
ain_hold2(LO)	Hold	0	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>				<input type="checkbox"/>	<input type="checkbox"/>
ain_hcode2(CH)	Code	0	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>				<input type="checkbox"/>	<input type="checkbox"/>

6.3.3.3 Adding the Function to the Web services Menu

To run the Page Web Service, first check that the menu function UWXSYESB005 exists. If it does not exist, then you need to create it.



Function	Description	Type	Module	Category
UWXSYESB005	User Defined Webservices	Menu	SY-System	SER-Service

Context: User Defined Webservices

Program: -

Parameters: -

Notes: -

Parent: -

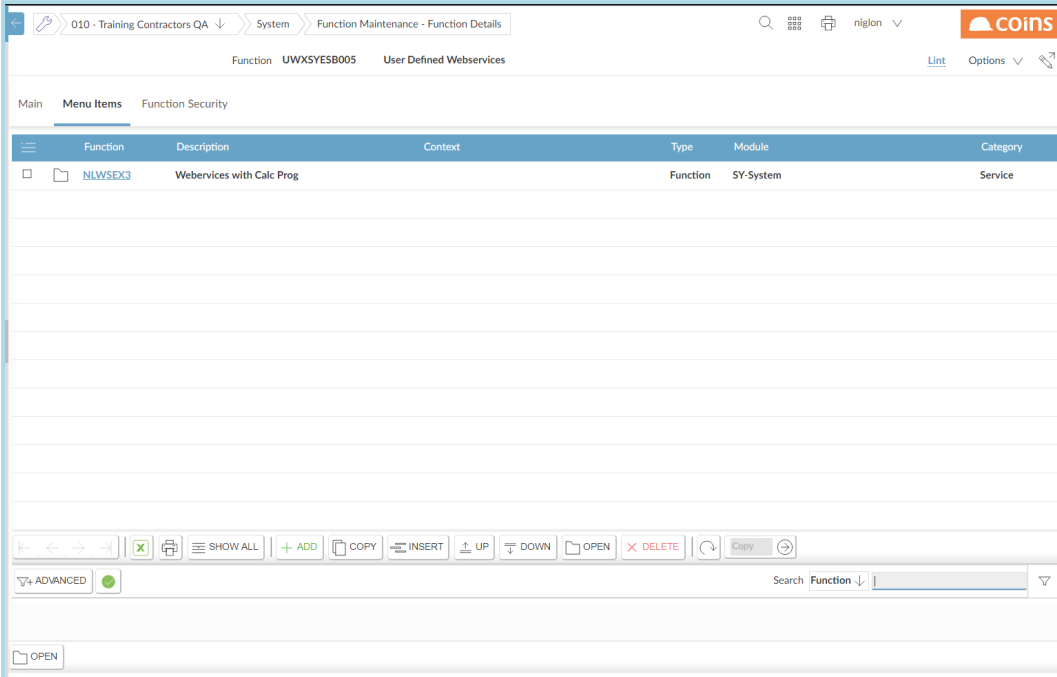
Access Type: -

Role Type: -

Open the Function and select the Menu Items Tab. If the function already exists, there may already be entries on this tab relating to Standard Used Defined Services issued by COINS (prefixed with %) or other User Defined Services created by your company.



Click **Add** and enter the name of your new User Defined Services Function (in our example NLWSEX3).

Function	Description	Context	Type	Module	Category
NLWSEX3	Webervices with Calc Prog		Function	SY-System	Service

Navigate back to the COINS Services Menu and select the SY - System Hyperlink.

COINS Services

Service	Description
CB	Cash Book
EC	eCommerce
FM	Facilities Management
GL	General Ledger
HS	House Sales
HR	Human Resources
JC	Contract Status
PL	Purchase Ledger
SC	Subcontract Ledger
SL	Sales Ledger
CS	Contract Sales Ledger
SY	System
VT	VAT
PC	Plant
DM	Document Management
PO	Procurement
CI	Company Information Workbench
CC	Customer Care
LA	Land Appraisal
CV	CVR
MK	Marketing
MB	Mobile Applications
SE	SE
ST	Stock

Locate the SYESB00 entries. Your new Function will be in this group.



COINS Services

COINS Services >System

Service	Description
SYESB000	Confirmation Message
SYESB001	Login
SYESB002	Get Waiting Message Numbers
SYESB003	WebServices with Dataset
SYESB004	WebServices with Pages
SYESB005	WebServices with Calc Prog
SYESB006	Web service for Appointments/Tasks
SYESB007	Print Out Set
SYESB011	Save Completed Mobile Form
SYESB012	Database Publish Renewal Data
SYESB013	Update User Mobile Configuration
SYESB014	Complete Workflow Stage

Select the Hyperlink to view the service schema.

COINS Services

COINS Services >System >Generic Program

Service	Description
SYESB005 (NLWSEX)	Generic Program

Schema [input.xsd output.xsd exception.xsd acknowledgement.xsd](#)
[SOAPInput.xsd SOAPOutput.xsd](#)

WSDL [SYESB005.wsdl](#)

WSDL NS [SYESB005.wsdl](#)

The generic program service allows a designer to build an OA calculation program and to call it using this service.
 The argument to this service is the function name of the program to be called.

Input

Entity	Type	Documentation
COINSInterface		
Header		
id	string	A unique message identifier from the originating system.
confirm	string	Whether a confirmation message is required. Set to true or yes to have a confirmation message returned. If this is set then an id must also be provided.
action	string	An action type that will indicate why the message was created. For example, this might be CREATE, UPDATE, or DELETE for a message as a result of a maintenance of a record or PUBLISH for the publish of some information.
entity	string	The service that should consume the message.
arguments	string	Arguments that should be passed to the service.
ackID	string	An optional acknowledgement ID. If this ID is set then COINS will respond with a acknowledgement message returning this ackID in the message header.
testMsg	boolean	Whether this message is a test message. A test message will process through in exactly the same way that a normal message would except that at the point where it would normally commit the transaction to the database, the message is then backed out.
UserID	string	The user in the originating system that caused the message to be produced.

The Input Schema - defined by the Update form of the page will look something like this:

Sample

```
<COINSInterface>
  <Header id="" confirm="" action="" entity="" arguments="" ackID="" testMsg="true">
    <UserID></UserID>
    <From></From>
    <HostName></HostName>
    <Environment></Environment>
    <Created>2014-04-10T11:54:54.523+01:00</Created>
    <Login>
      <User></User>
      <Password></Password>
      <CID>1</CID>
      <Group></Group>
      <extUser></extUser>
      <extAuth></extAuth>
    </Login>
  </Header>
  <Body>
    <kco>99</kco>
    <avm_num></avm_num>
    <ain_inv></ain_inv>
    <ain_hold2>true</ain_hold2>
    <ain_hcode2></ain_hcode2>
  </Body>
</COINSInterface>
```

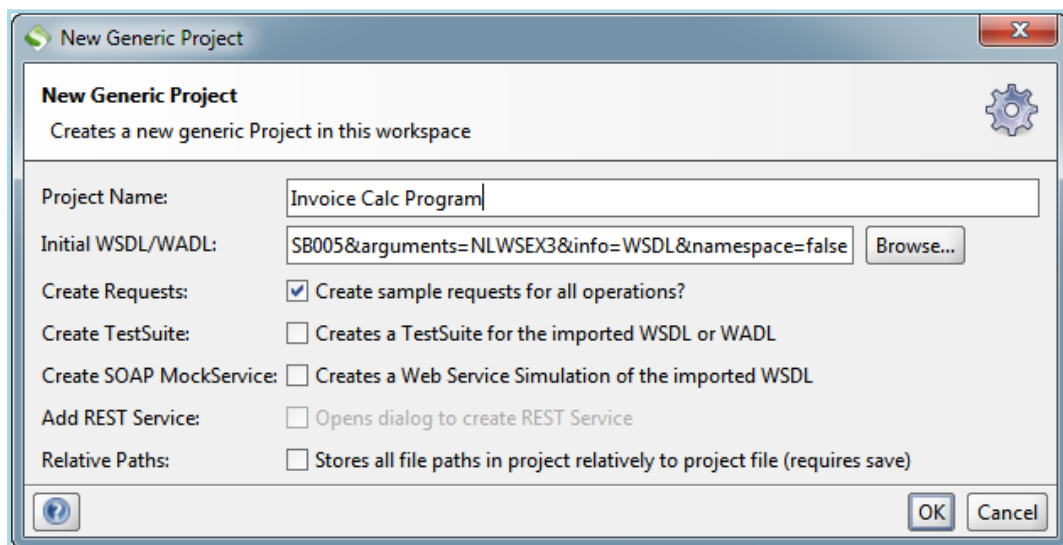


The Output Schema - defined by the Detail Form of the page will look something like this:

Sample

```
<COINSInterface>
  <Header id="" confirm="" action="" entity="" ackID="">
    <UserID></UserID>
    <From></From>
    <HostName></HostName>
    <Environment></Environment>
    <Created>2014-04-10T11:54:54.605+01:00</Created>
  </Header>
  <Body>
    <ain_hold2>true</ain_hold2>
    <ain_hcode2></ain_hcode2>
  </Body>
</COINSInterface>
```

6.3.4 Testing the Service



Create a new soapUI project and from the Service Schema copy the WSDL shortcut link and paste this into the Initial WSDL field

Click OK

Once the project has been created, run Request 1 in the editor. Replace the Input message with the simplified one from the Input Schema from the Web service.



Request 1

http://195.40.14.50/cgi-bin/oatrainbi/wouesb.p?service=SYESB005&arguments=NLWSEX3&namespace=false

```
<COINSInterface>
<Header id="" confirm="" action="" entity="" arguments="" ackID="" testMsg="true">
  <UserID></UserID>
  <From></From>
  <HostName></HostName>
  <Environment></Environment>
  <Created>2014-04-10T11:54:54.523+01:00</Created>
  <Login>
    <User></User>
    <Password></Password>
    <CID>1</CID>
    <Group></Group>
    <extUser></extUser>
    <extAuth></extAuth>
  </Login>
</Header>
<Body>
  <kco>99</kco>
  <avm_num></avm_num>
  <ain_inv></ain_inv>
  <ain_hold2>true</ain_hold2>
  <ain_hcode2></ain_hcode2>
</Body>
</COINSInterface>
```

Enter the hostname, Environment and username/password details.

Delete the lines for Group, extUser and extAuth

Fill in the appropriate Company, Supplier, Invoice and Hold code details for a suitable test invoice. For example:

```
<COINSInterface>
<Header id="" confirm="" action="" entity="" arguments="" ackID="" testMsg="true">
  <UserID></UserID>
  <From></From>
  <HostName>UKSLOUX018.coins.local</HostName>
  <Environment>trainbi</Environment>
  <Created>2014-04-10T09:54:42.130+01:00</Created>
  <Login>
    <User>train1</User>
    <Password>train1</Password>
    <CID>1</CID>
  </Login>
</Header>
<Body>
  <kco>10</kco>
  <avm_num>ABB001</avm_num>
  <ain_inv>11090085</ain_inv>
  <ain_hold2>yes</ain_hold2>
  <ain_hcode2>A2</ain_hcode2>
</Body>
</COINSInterface>
```

Send the message and correct any exceptions if they occur.



If all is correct, a RESPONSE message should be returned.

```
<COINSInterface>
  <Header id="" confirm="" action="" entity="" arguments="" sockID="" testMsg="true">
    <UserID>/UserID</UserID>
    <From>/From</From>
    <HostName>WS2LOD0018.coins.local</HostName>
    <Environment>train1</Environment>
    <Created>2014-04-10T09:54:42.130+01:00</Created>
  </Header>
  <Login>
    <User>train1</User>
    <Password>train1</Password>
    <CID>1</CID>
  </Login>
  </Header>
  <Body>
    <kc>10</kc>
    <svm_num>AMB001</svm_num>
    <ain_inv>11090085</ain_inv>
    <ain_hold1>yes</ain_hold1>
    <ain_hold2>22</ain_hold2>
  </Body>
</COINSInterface>

<COINSInterface>
  <Header action="" RESPONSE="" entity="" type="005">
    <UserID>train1</UserID>
    <From>COINS</From>
    <HostName>WS2LOD0018.coins.local</HostName>
    <Environment>train1</Environment>
    <Created>2014-04-10T12:14:39.888+01:00</Created>
  </Header>
  <Body>
    <ain_hold1>yes</ain_hold1>
    <ain_hold2>22</ain_hold2>
  </Body>
</COINSInterface>
```



Once you are happy with the response, you can set the testMsg to “false” to apply the change to the database.

Check the invoice in COINS to verify the changes are taking place as expected.

6.4 Get/Set

Unlike the previous services, Get/Set services do not require functions or pages to be set up as their operation is pre-defined. As suggested by the name, the purpose of a Get/Set service is to retrieve a specific record and set values within it. Get/Set cannot be used to add records, only retrieve/amend existing ones.

From the COINS Services Menu, select the SY - System hyperlink.

COINS Services

[COINS Services](#) > System

Service	Description
SYESB000	Confirmation Message
SYESB001	Login
SYESB002	Get Waiting Message Numbers
SYESB003	Webservices with Dataset
SYESB004	Webservices with Pages
SYESB006	Web service for Appointments/Tasks
SYESB007	Field Get/Set
SYESB011	Save Completed Mobile Form
SYESB012	Database Publish Resend Data
SYESB013	Update User Mobile Configuration
SYESB014	Complete Workflow Stage



Locate the SYESB007 - Field Get/Set entry

Select the hyperlink to view the service schema.



COINS Services

COINS Services > System > Field Get/Set

Service	SYESB007
Description	Field Get/Set
Schema	input.xsd output.xsd exception.xsd acknowledgement.xsd
WSDL	SYESB007.wsdl
WSDL NS	SYESB007.wsdl

The generic field service allows a service client to request fields from and/or update fields in a specific record.
The criteria specified to find the record must result in a single record being returned.
If set field name and value pairs are specified then they are set before the get returns the current values.

Input

Entity	Type	Documentation
COINSInterface		
Header		
id	string	A unique message identifier from the originating system.
confirm	string	Whether a confirmation message is required. Set to true or yes to have a confirmation message returned. If this is set then an id must also be provided.
action	string	An action type that will indicate why the message was created. For example, this might be CREATE, UPDATE, or DELETE for a message as a result of a maintenance of a record or PUBLISH for the publish of some information.
entity	string	The service that should consume the message.
arguments	string	Arguments that should be passed to the service.
ackID	string	An optional acknowledgement ID. If this ID is set then COINS will respond with a acknowledgement message returning this ackID in the message header.
testMsg	boolean	Whether this message is a test message. A test message will process through in exactly the same way that a normal message would except that at the point where it would normally commit the transaction to the database, the message is then backed out.
UserID	string	The user in the originating system that caused the message to be produced.
From	string	The name of the system that produced the message.

New Generic Project

Creates a new generic Project in this workspace

Project Name:

Initial WSDL/WADL:

Create Requests: ☒ Create sample requests for all operations?

Create TestSuite: ☐ Creates a TestSuite for the imported WSDL or WADL

Create MockService: ☐ Creates a Web Service Simulation of the imported WSDL

Add REST Service: ☐ Opens dialog to create REST Service

Relative Paths: ☐ Stores all file paths in project relatively to project file (requires save)

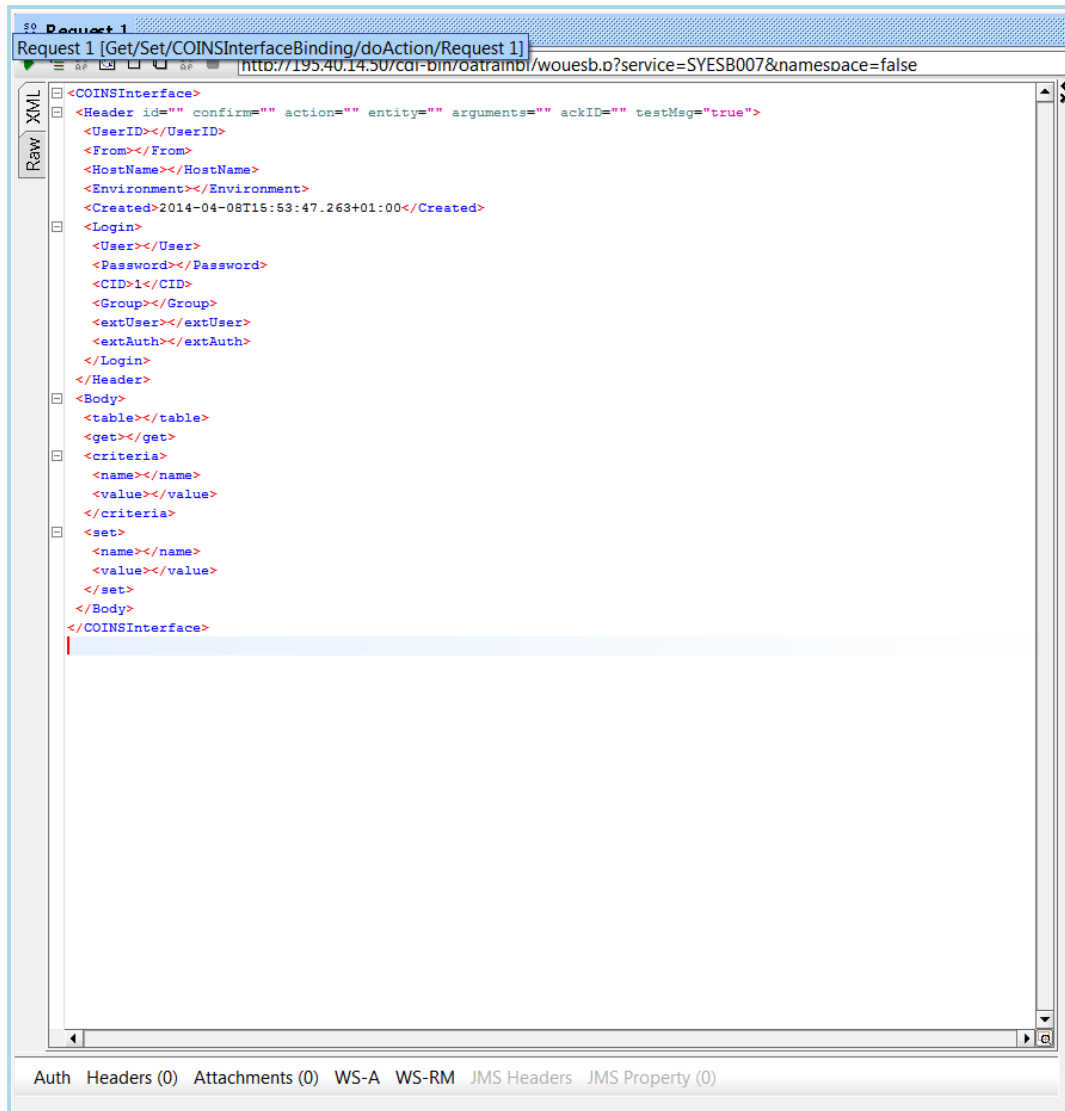
Create Web TestCase: ☐ Creates a TestCase with a Web Recording session for functional web testing

Create a new soapUI project and, from the service schema, copy the WSDL shortcut link into the initial WSDL field.

Click OK

Once the project has been created, run Request1 in the editor.

Replace the Input message in the editor with the simplified version from the Input Message Schema



Enter the hostname, Environment and username/password detail.

Delete the lines for Group, extUser and extAuth

In the Body section of the Input Message, the first entries are:

```
<table></table>
<get></get>
```

These allow you to specify the table the record will come from and the fields that will be returned in the Output Message. The get statement is a comma separated list of fields.



The next lines relate to the criteria which will identify the record to be retrieved. The entries require the name of the key field and the value to be used to identify the record.

```
<criteria>  
  <name></name>  
  <value></value>  
</criteria>
```

If there is more than one key field which identifies a record, you will need to copy the Criteria block for each required field. For example, for a table with two key fields:

```
<criteria>  
  <name></name>  
  <value></value>  
</criteria>  
<criteria>  
  <name></name>  
  <value></value>  
</criteria>
```

Finally, the Set block of lines allow you to specify the field to be changed and the value to which it should be set.



```
<set>

  <name></name>

  <value></value>

</set>
```

As with the Criteria block, if more than one field is to be changed, there should be a block of set lines for each field.

In the following example, a supplier record in the ap_vendor table will be retrieved and the remarks field in the supplier record will be updated.

```
<COINSInterface>
<Header id="" confirm="" action="" entity="" arguments="" ackID="" testMsg="false">
  <UserID></UserID>
  <From></From>
  <HostName>UKSLOUX018.coins.local</HostName>
  <Environment>trainbi</Environment>
  <Created>2014-04-08T15:53:47.263+01:00</Created>
  <Login>
    <User>train1</User>
    <Password>train1</Password>
    <CID>1</CID>
  </Login>
</Header>
<Body>
  <table>ap_vendor</table>
  <get>kco,avm_num,avm_name</get>
  <criteria>
    <name>kco</name>
    <value>10</value>
  </criteria>
  <criteria>
    <name>avm_num</name>
    <value>ABB001</value>
  </criteria>
  <set>
    <name>avm_remarks</name>
    <value>Test remarks from Webservices</value>
  </set>
</Body>
</COINSInterface>
```

Sending this message should produce the following return with the Output message showing the three fields we requested in the get entry of the Input Message.



```

Request 1
Request 1 [Get/Set/COINSInterfaceBinding/doAction/Request 1]
<?xml version="1.0" encoding="UTF-8" ?>
<Header id="" confirm="" action="" entity="" arguments="" ackID="" testMsg="false">
  <DevID></DevID>
  <From></From>
  <HostName>UWSL000018.coins.local</HostName>
  <Environment>trainbit</Environment>
  <Created>2014-04-02T15:59:47.369+01:00</Created>
  <Login>
    <DevID>trainbit</DevID>
    <Password>trainbit</Password>
  </Login>
  <CID></CID>
</Header>
<Body>
  <table>
    <tr>
      <td>get_vendor</td>
      <td>get</td>
      <td>get_kco, avm_num, avm_name</td>
    </tr>
  </table>
  <criteria>
    <name>kco</name>
    <value>id</value>
  </criteria>
  <criteria>
    <name>avm_num</name>
    <value>ABB001</value>
  </criteria>
  <set>
    <name>avm_remarks</name>
    <value>Test remarks from Webservices</value>
  </set>
</Body>
</COINSInterface>
  
```



Remember that leaving the Test Message entry set to “true” will process the messages but will not apply the changes to the database. The test message must be set to “false” for database changes to occur.

If the database change is successful our example will have the following effect on the supplier record:

Main	Terms	Configuration	Payment Method	Analysis Sets	Insurance	Notes
<p>Supplier Account: ABB001</p> <p>Payee Name: Abbey Glass Name: Abbey Glass</p> <p>Address: 42 Bramall Lane Sheffield</p> <p>Postcode: S25 4DL (SHEFFIELD)</p> <p>Search Name: Abbey Glass Short Name: Abbey</p> <p>Telephone: 01642 887766</p> <p>Default Print Method: Fax: 01642 887767 Email: tim.drake@coins-global.com</p> <p>Contact: Supplier's Code for Us:</p> <p>Country: GB-United Kingdom VAT-Registration-No: 762070208 - VAT- D4 Purchase-Standard Rate @4.7.6%</p> <p>Remarks: Test remarks from Webservices</p> <p>Account Currency: GBP-Sterling Allow Other Currencies: <input type="checkbox"/></p> <p>Account Closed: <input type="checkbox"/> Account Dormant: <input type="checkbox"/> Factor: <input type="checkbox"/></p>						