User Defined Web Services

Prepared by:	Documentation Team
Prepared for:	Learning Resources
Module:	Business Intelligence
Date:	2017
Document Ref:	LMDSY0037
Version:	11.06

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

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

Contents

1 User Defined Webservices

What are web services? The easiest way to answer this question is to think of web services as a way of extracting and putting information from and to COINS.

It allows COINS to interact with other systems, this is not necessarily other ERP systems but companies such as HMRC.

For a long time, COINS have used web services, although it hasn't been as obvious in the UI as it is in the background, CIS, eRCT, RTI all use forms of SOAP (Simple Object Access Protocol) requests to contact and exchange information with a third party.

Web services works by exchanging the data in an xml format, this format ensures that each part of the data that you are sending/exchanging is encompassed by 'tags' that define where/what the system should be doing with it.

Each tag starts with and ends with the triangle brackets (< and >) to classify and denote an ending of a tag the forward slash is used; each tag will end up looking something similar to this:

<this is a tag> DATA </this is a tag>

This section 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:

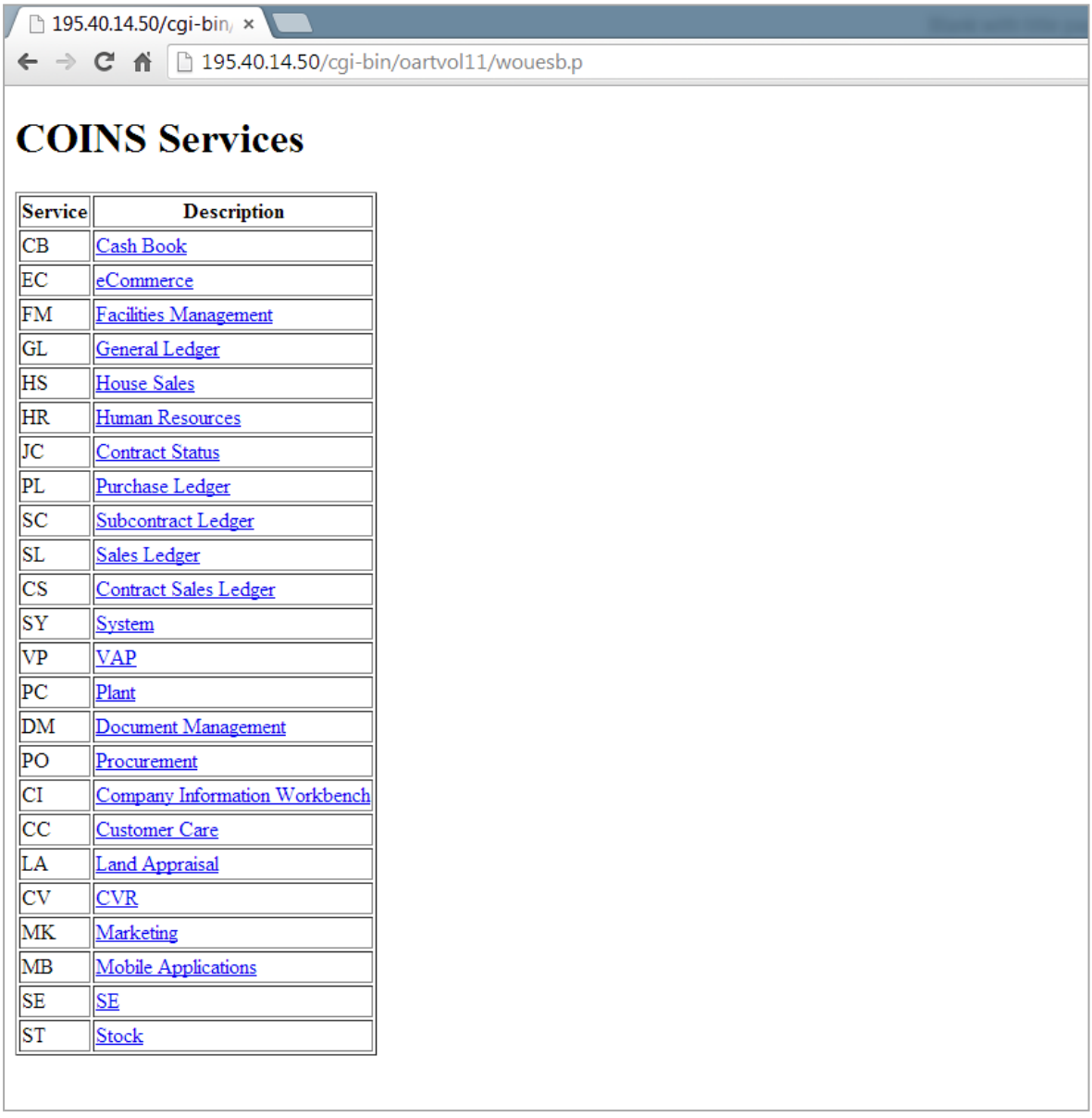
http://195.40.14.50/cgi-bin/oartvol11/wologin.p

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

For example:

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

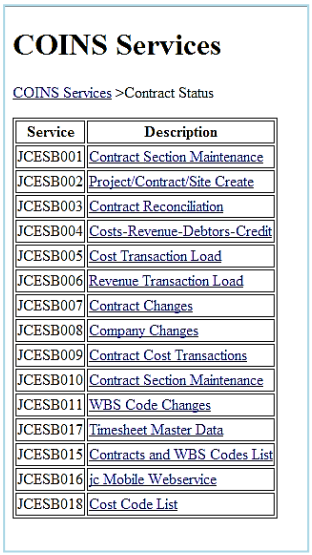
The following page should be displayed.



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 licensing 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 licensed to do so.

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

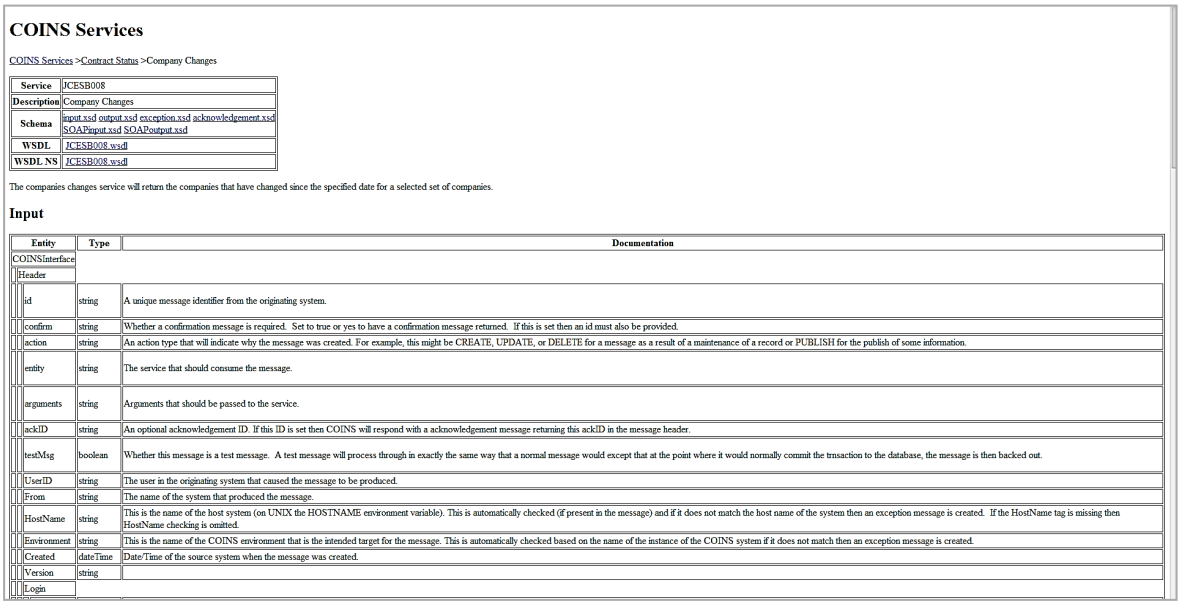


COINS Services

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	jc 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.wsdl

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 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.
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 Understanding COINS Centric XML

3.1 <Header>

The header portion of the XML is broken down into only two sections, the <header> tag and the <login> tag.

A lot of the fields here are used to determine where you are wanting to send the xml request to, fields such as <Userid> and <from> are not necessary and can be overlooked.

The header in its full form looks like this;

This can be amended to look like the following (this is not inclusive of data but only the necessary headings)

The data that needs to be in the header is relatively straight forward, the hostname is the actual host name of the server, this can be found either on the unix server or in RF under the customer information.

Example of a completed header is below, please note that the TestMsg flag is set to “false” this enables the web service request to be taken as a true request action.

The header information does not change regardless of which web service you use, the only change would be if the customer changes, i.e Taylor Wimpey instead of persimmons as their host name and login details would be different.

3.2 <Body>

The body information will always be different, these are the defined fields and data that you are wanting to either insert into coins or get out of coins.

In this example we’ve been referencing and using HSESB001 which is the “release plot for sale” web service.

A separate document will be put into place to ensure the understanding of Mass loading data, this is more to do with understanding how to import large quantities of data into one request and how that is handled.

Earlier on in this document we referenced header tags that do not complete straight away, instead fields appear underneath the start tag and aren't finished until the last field is referenced, an example being <body> </body> and <hs_transRow> </hs_transRow>

It is worth noting that the start of the xml will always start with <COINSInterface> and end with </COINSInterface> (first and last tag),

From a data migration perspective, think of each tag (start and end) as separate columns in a spreadsheet, within each of the tags exist the data. (Remember this for later on)

The Body section consists of the following, the starting <body> tag and the ending </body> tag. The Body tag is then in turn has an encapsulated <hs_transRow> start and end </hs_transRow>.

Fields such as kco are less important if you are importing into only one kco which I would recommend dealing with separately anyway.

The external ID tag is also irrelevant but provides a way to identify externally the different rows if required.

Other fields will look relatively familiar such as job_num (Development) and vwb_code (plot)

If you are ever unsure of the fields you can either search the web service list (wouesb.p) or navigate to within the UI (Coins OA) where the web service will be inserting records and check to see the fields and the field labels.

Once you are happy with the data, the completed form should look like the below, this is the header, body and the subsequent hs_transRow as well.

The most important parts of the XML is the testMsg being equal to false, the ending of each of the tags, and the rsp_action.

The rsp_action tells the web service what you are wanting to do with the web service,

There are three valid options

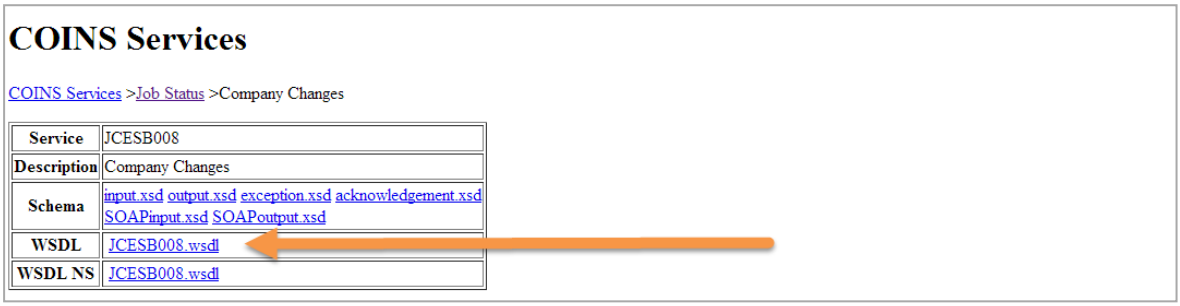
I = Insert – this inserts a row of data into the system.

U = Update – This updates information already present in the system.

D = Delete – This deletes a record that is in the system.

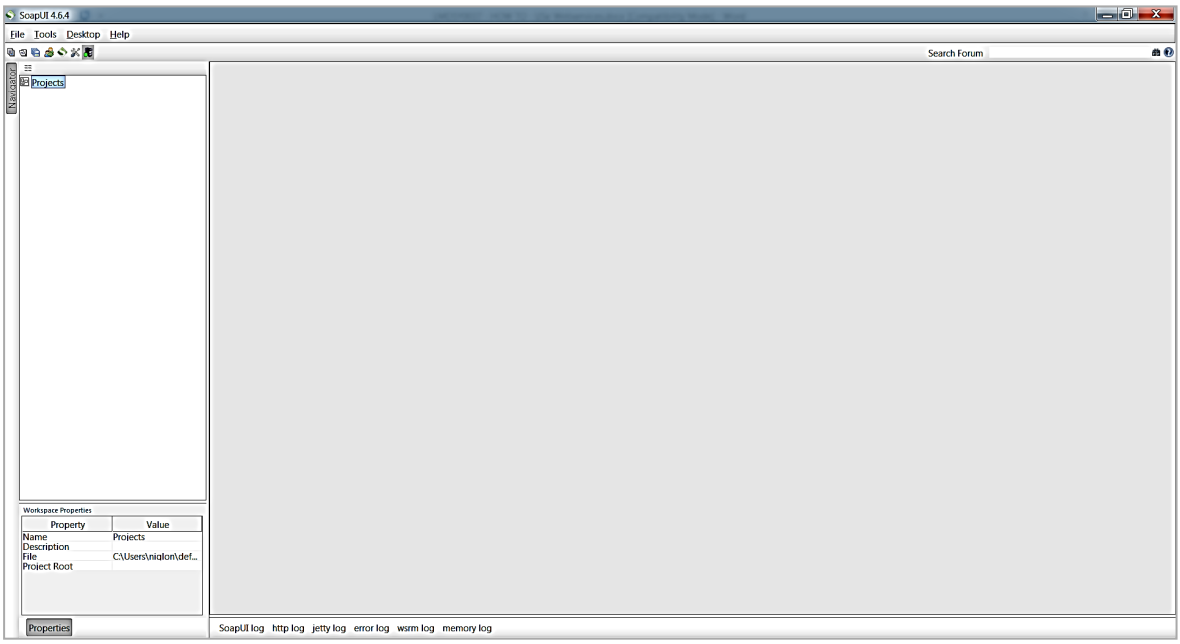
4 Testing a Service Connection (Using soapUI)

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.

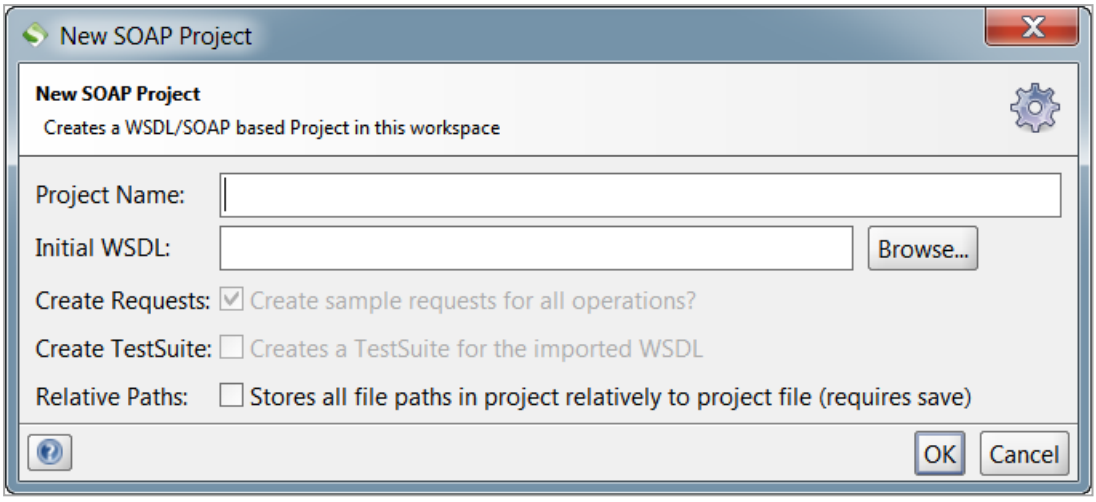


Right-click on this and select Copy Shortcut.

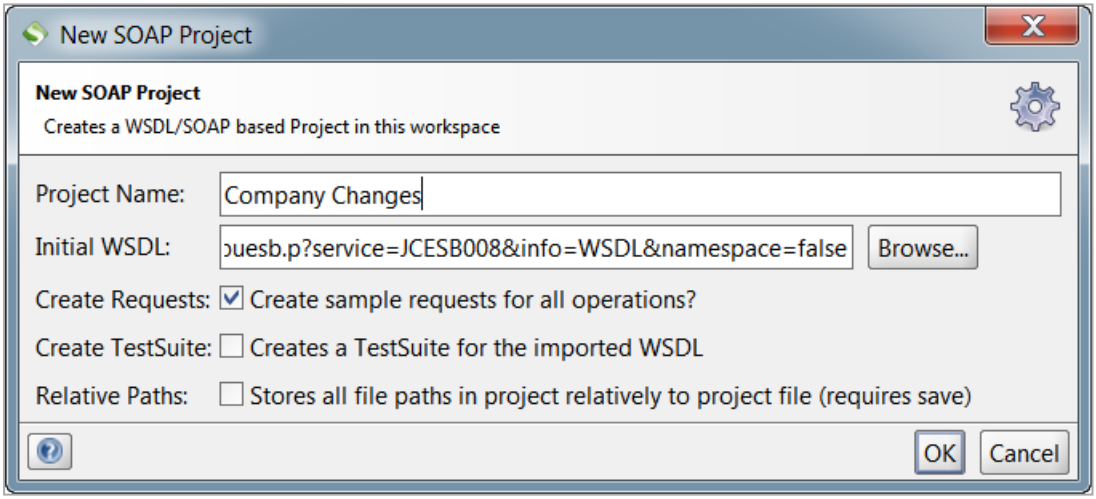
Launch the soapUI application.



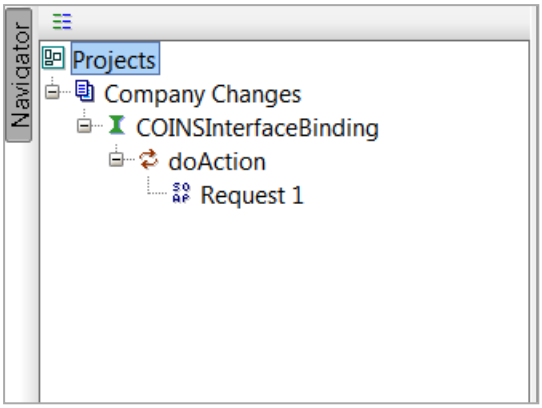
Select File/New SOAP Project.



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



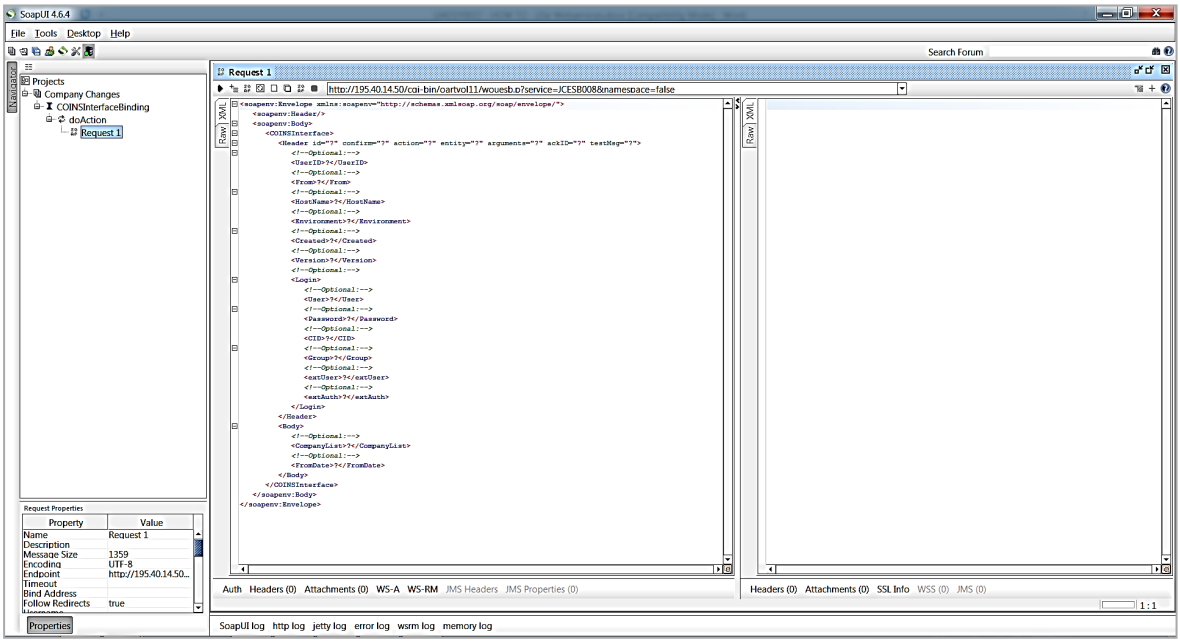
Click OK.



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.



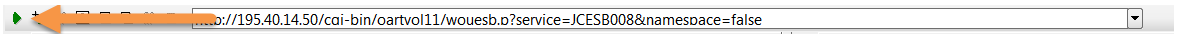
The message details (shown below left) are the same as those generated as Sample Output in the Services Browse (shown below right). The only difference is the version in the soapUI is more verbose with details such as optional fields indicated.

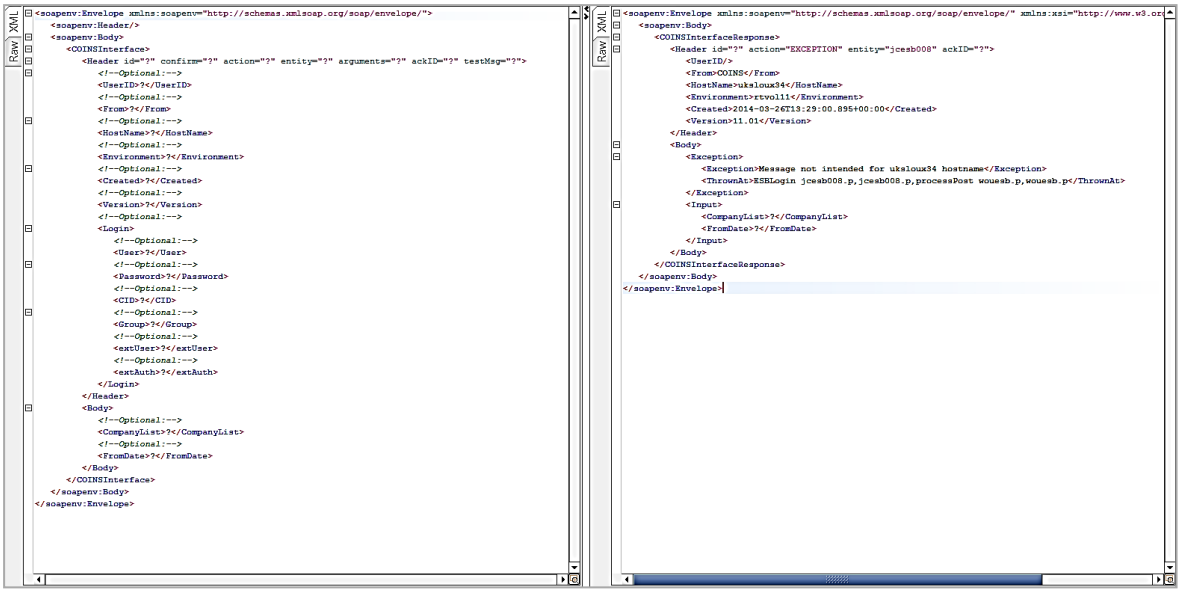
Sample

```
<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>
```

```
<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>
```

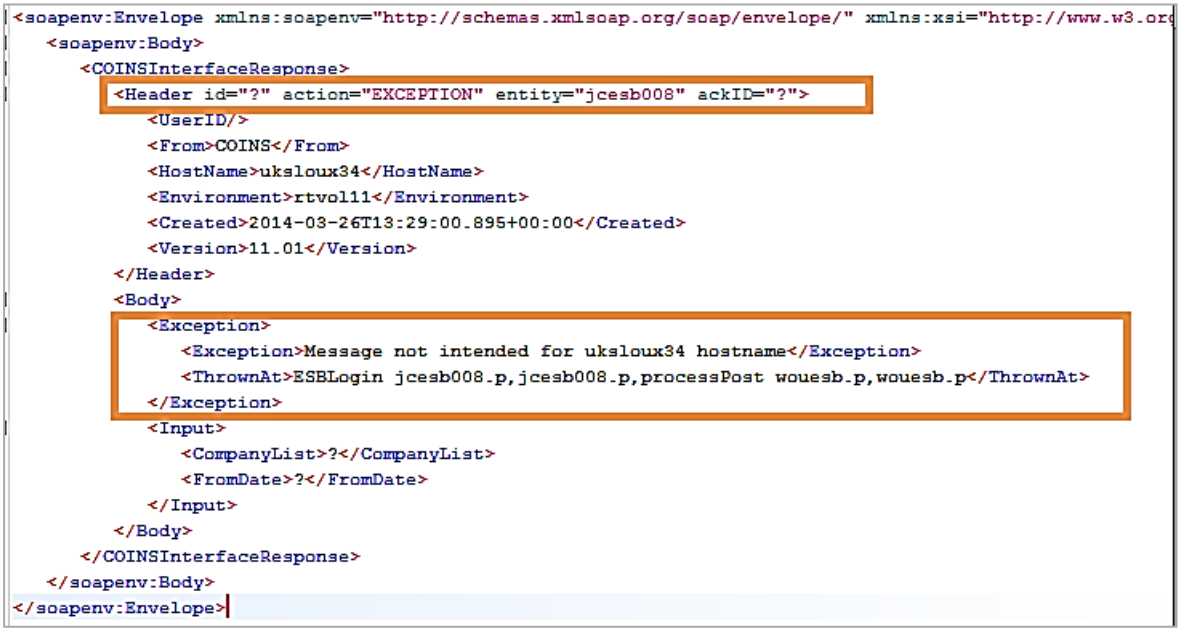
To submit the request message, click the green arrow to the left of the toolbar





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:



```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org
   <soapenv:Body>
      <COINSInterfaceResponse>
         <Header id="?" action="EXCEPTION" entity="jcesb008" ackID="?">
            <UserID/>
            <From>COINS</From>
            <HostName>uksloux34</HostName>
            <Environment>rtvol11</Environment>
            <Created>2014-03-26T13:29:00.895+00:00</Created>
            <Version>11.01</Version>
         </Header>
         <Body>
            <Exception>
               <Exception>Message not intended for uksloux34 hostname</Exception>
               <ThrownAt>ESBLogin jcesb008.p,jcesb008.p,processPost wouesb.p,wouesb.p</ThrownAt>
            </Exception>
            <Input>
               <CompanyList>?</CompanyList>
               <FromDate>?</FromDate>
            </Input>
         </Body>
      </COINSInterfaceResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

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/X
   <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>
               <thrownAt>ESBLogin jcesb008.p,jcesb008.p,processPost wouesb.p,wouesb.p</thrownAt>
            </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.

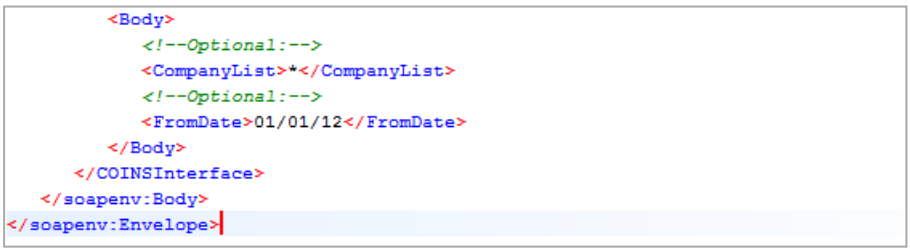
```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Header/>
   <soapenv:Body>
      <COINSInterface>
         <Header id="?" confirm="?" action="?" entity="?" arguments="?" ackID="?" testMsg="?"
            <!--Optional:-->
            <UserID>?</UserID>
            <!--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.

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:



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/X
   <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>&lt;P>1234567890 324567832&lt;/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>
```

We have now proved that:

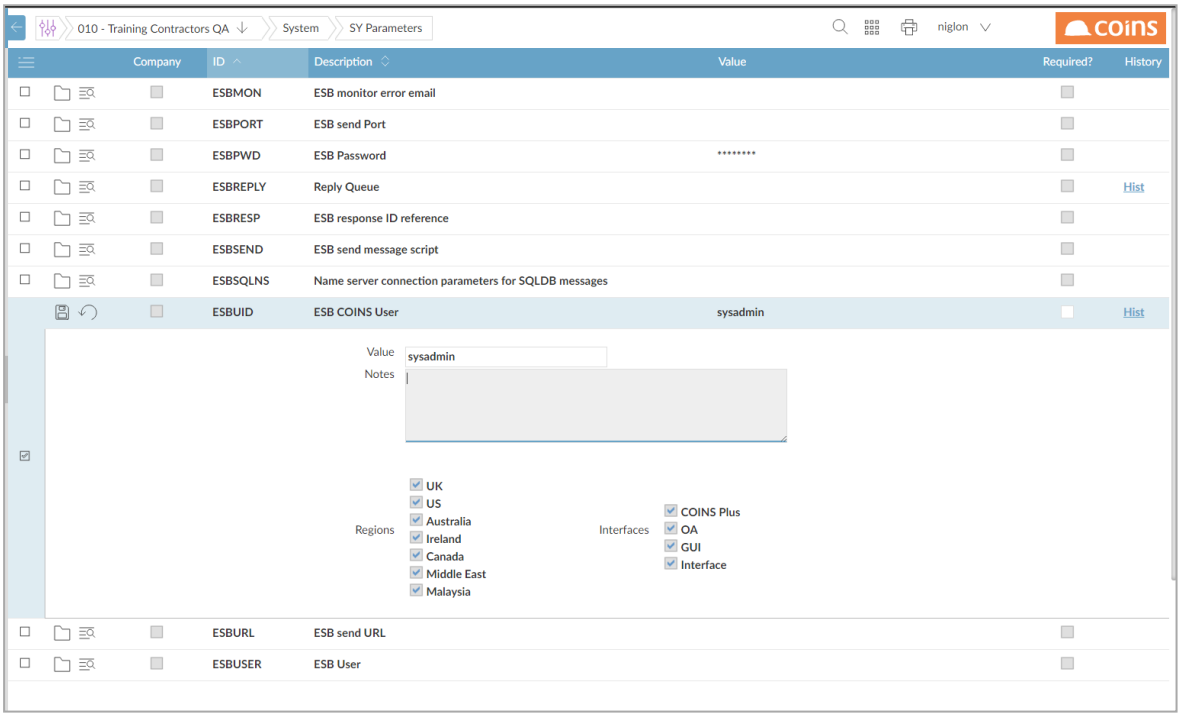
- the Services are working
- we can configure the messages appropriately
- we can call them and get a response

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.

5 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.



Parameter ESBUID allows you to specify a user id (licensed 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>
            <!--Optional:-->
            <CompanyList>*</CompanyList>
            <!--Optional:-->
            <FromDate>01/01/12</FromDate>
         </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/X
   <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>
```

6 soapUI

Yo can use the Open Source tool soapUI to demonstrate the interaction of Web Services. It is assumed that clients working with Web Services will have their own tools and applications, but if you wish to follow COINS Learning Materials directly, soapUI can be downloaded from.:

http://www.soapui.org

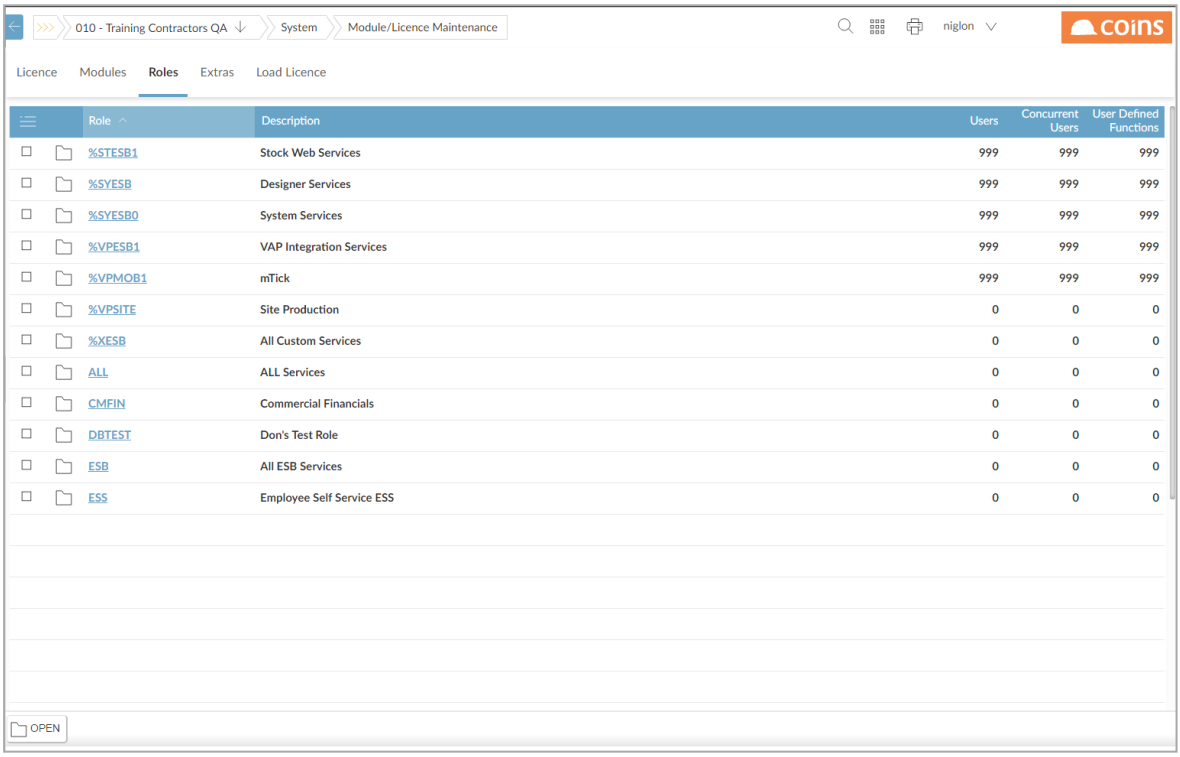
7 Licensing

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

7.1 Additional Licensing

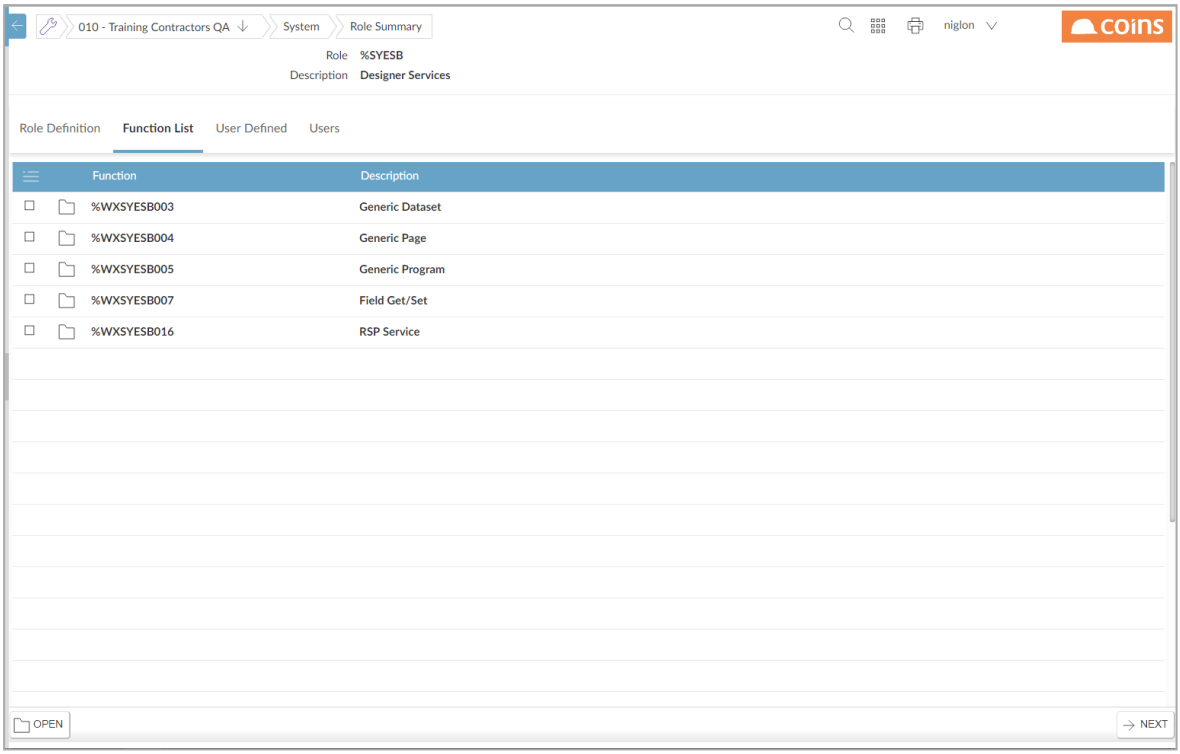
User Defined Services require an additional element of licensing.

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

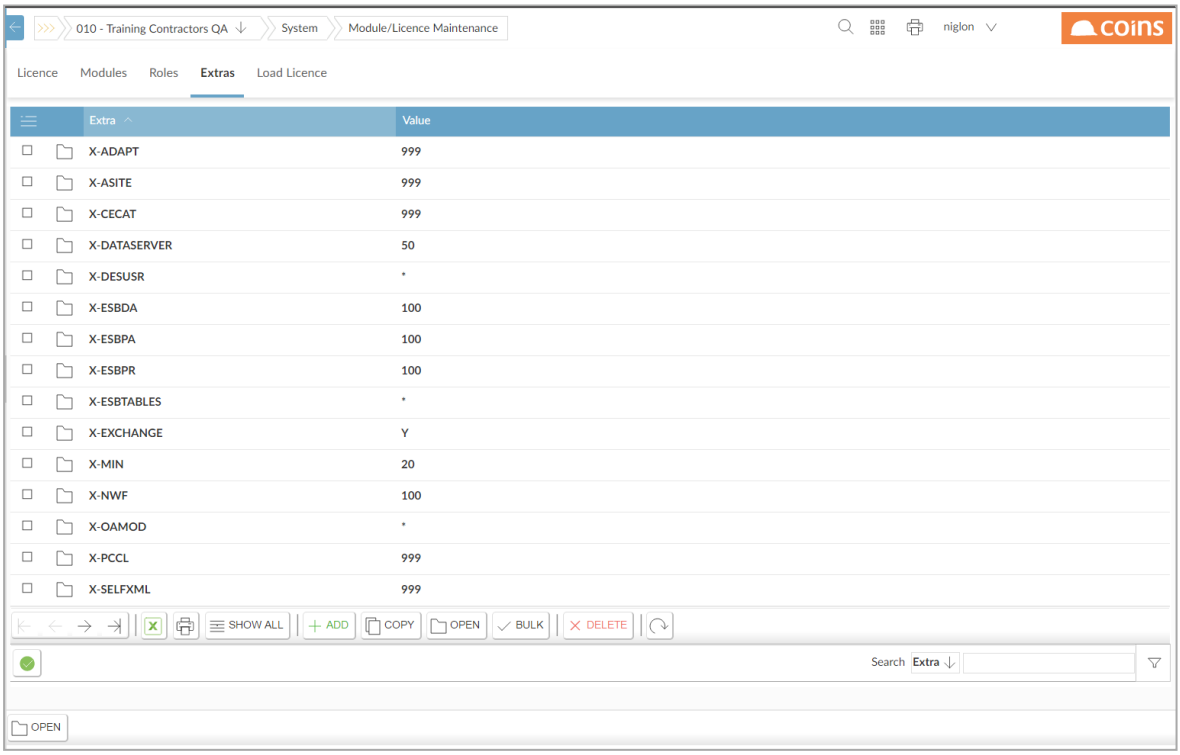


Role	Description	Users	Concurrent Users	User Defined Functions
%STESB1	Stock Web Services	999	999	999
%SYESB	Designer Services	999	999	999
%SYESB0	System Services	999	999	999
%VPESB1	VAP Integration Services	999	999	999
%VPMOB1	mTick	999	999	999
%VPSITE	Site Production	0	0	0
%XESB	All Custom Services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

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



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:



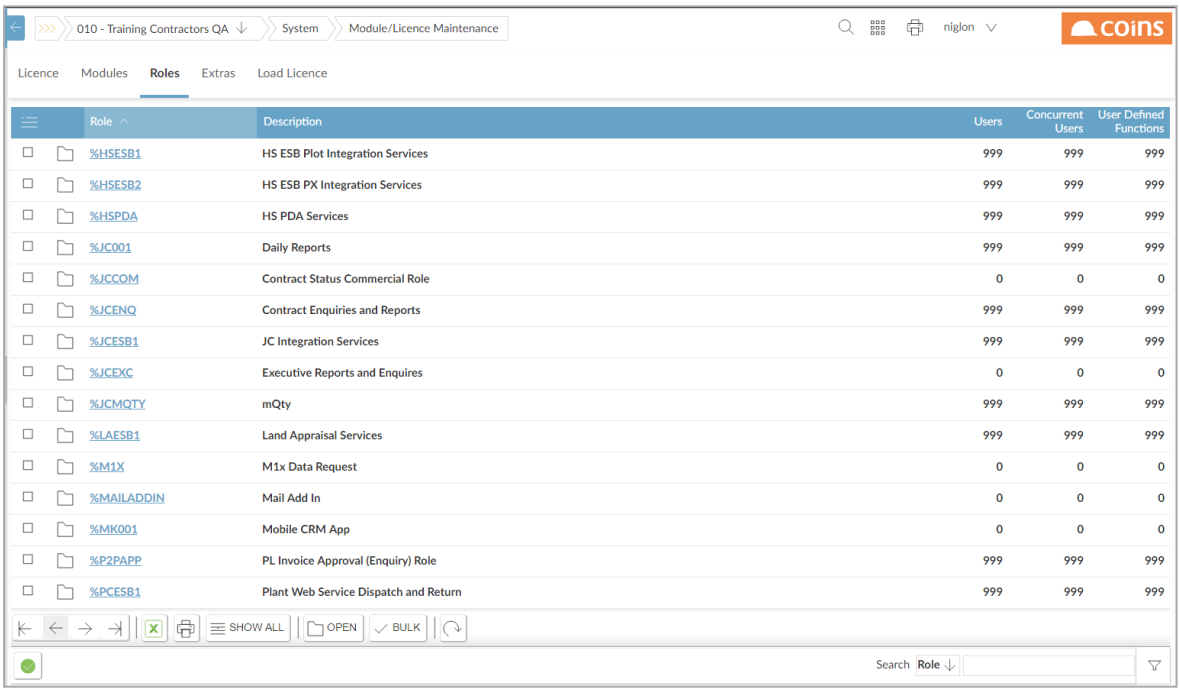
X-ESBDA	controls how many datasets can be created within user defined services.
X-ESBPA	controls the number of pages that can be built
X-ESBPR	controls how many user defined programs
X-ESBTABLES	controls how many GET-SET services may be defined.

7.2 Roles

Web Services are licensed using Roles. These may be found within System > Setup> Module Licence Maintenance..

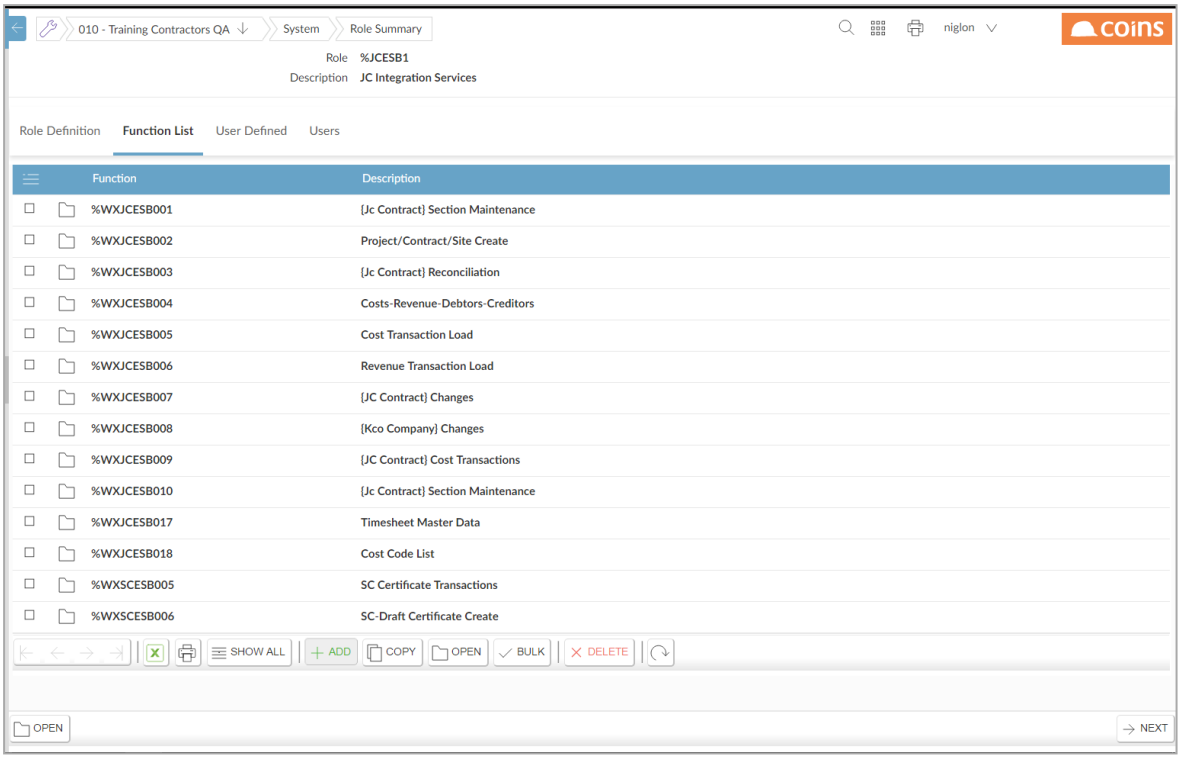
This function contains all the different types of roles associated which includes more than just the web services, it is worth noting that all the web service roles contain the letters 'ESB' so can easily be identified. If the role does not exist it will never appear on the user record under the 'groups' section, to import a role from Development please see the next section

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
%HSESB1	HS ESB Plot Integration Services	999	999	999
%HSESB2	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
%PCESB1	Plant Web Service Dispatch and Return	999	999	999

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


010 - Training Contractors QA
System
Role Summary
niglon
coins
Role %JCESB1
Description JC Integration Services
Role Definition
Function List
User Defined
Users
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
%WXSCESB005 SC Certificate Transactions
%WXSCESB006 SC-Draft Certificate Create
SHOW ALL
ADD
COPY
OPEN
BULK
DELETE
OPEN
NEXT

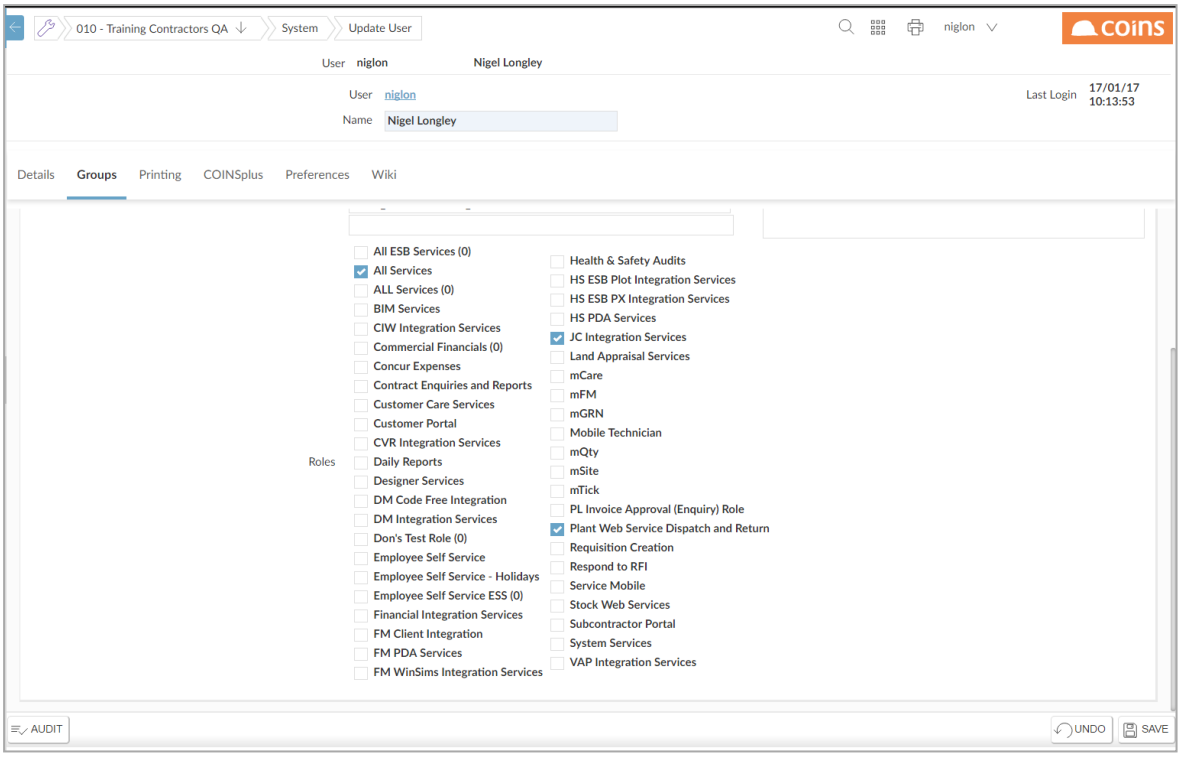
7.3 User Maintenance

In order for any of the web services to work (regardless of it being accessible and viewable) the role and the user has to be set up correctly. This depends on a few things, the role needs to be in the system, if it isn't you'd need to log into dev, export the role and import it into the customer environment. You MUST be logged in as sysadmin on the customer environment to be able to see the 'export/import role' function.

The user must also be set up to be able to use web services and must have the correct role assigned, please bear in mind that any new role used has a cost implication and should be discussed with the account manager/project manager.

The User profile needs only three things in order for it to work correctly, against the main 'details' page under the 'named user' section, there is a tick box 'web services' that needs to be ticked, a web services company needs to also be specified.

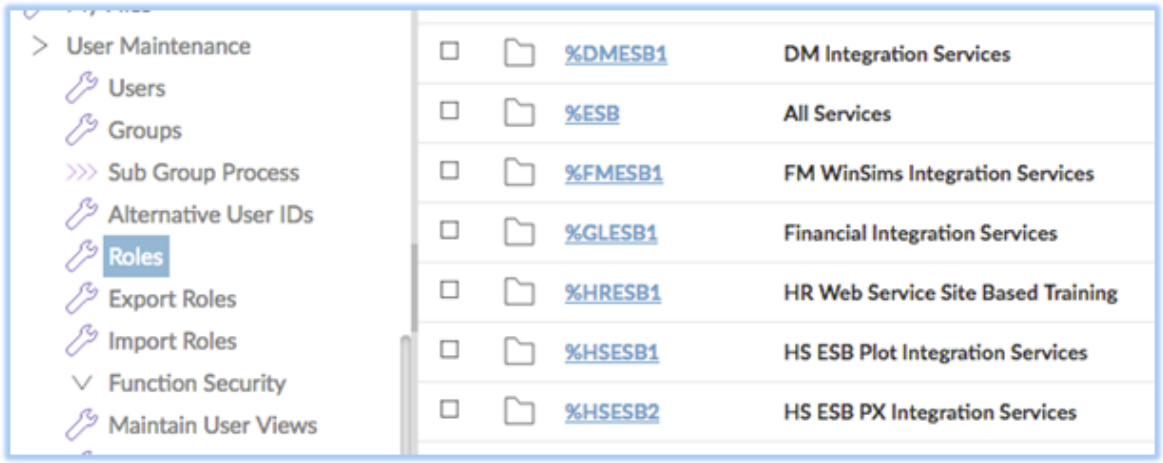
On the second tab 'GROUPS' the web service role needs to be selected, the only exception to the individual web service roles are when implementing and testing, this can be achieved by using the %ESB role which essentially gives access to ALL web services, very few customers will have this in place, only thelarger customers will have this.



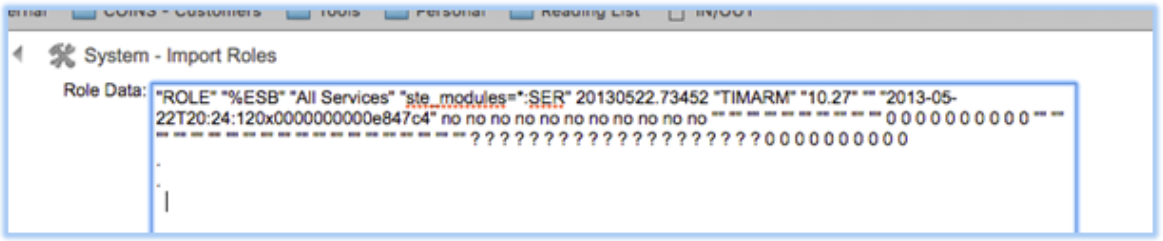
Tick the appropriate Roles – subject to the licensing granted to your company and click **SAVE**

7.4 Importing Roles

Log in as sysadmin and navigate to the SY module > User Maintenance; You will find the roles functions appear under the user maintenance folder, which will allow you to Import the role that you copied from the development environment.



Once you click on the 'import role' function you can then Paste in the role from the development environment export screen and press **NEXT**,



This will import the role and make it available if the licence defines more than zero against the role in question.

Under the User function, select the user that you are wanting to work with and 'open' the user record.

if the role is defined in the licence it will now appear under the groups record where you can tick the applicable role.

8 User Defined WebServices

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).

8.1 Datasets

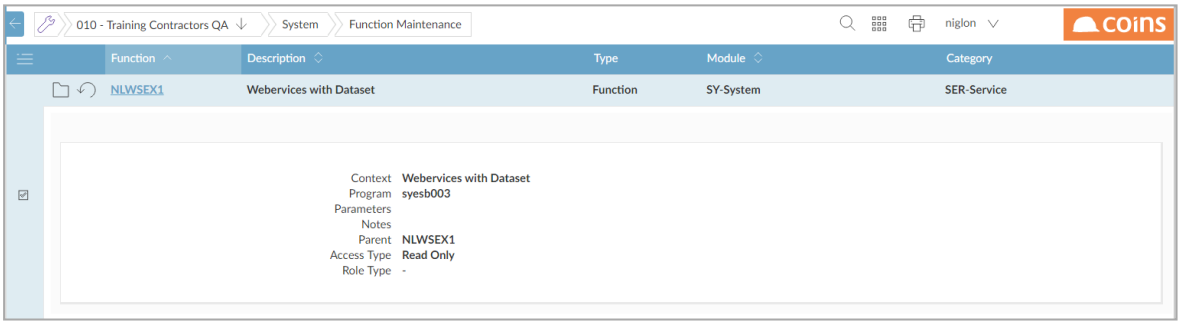
Three components are required for Datasets. A function, a page and the dataset itself.

> Note: 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.

8.1.1 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.



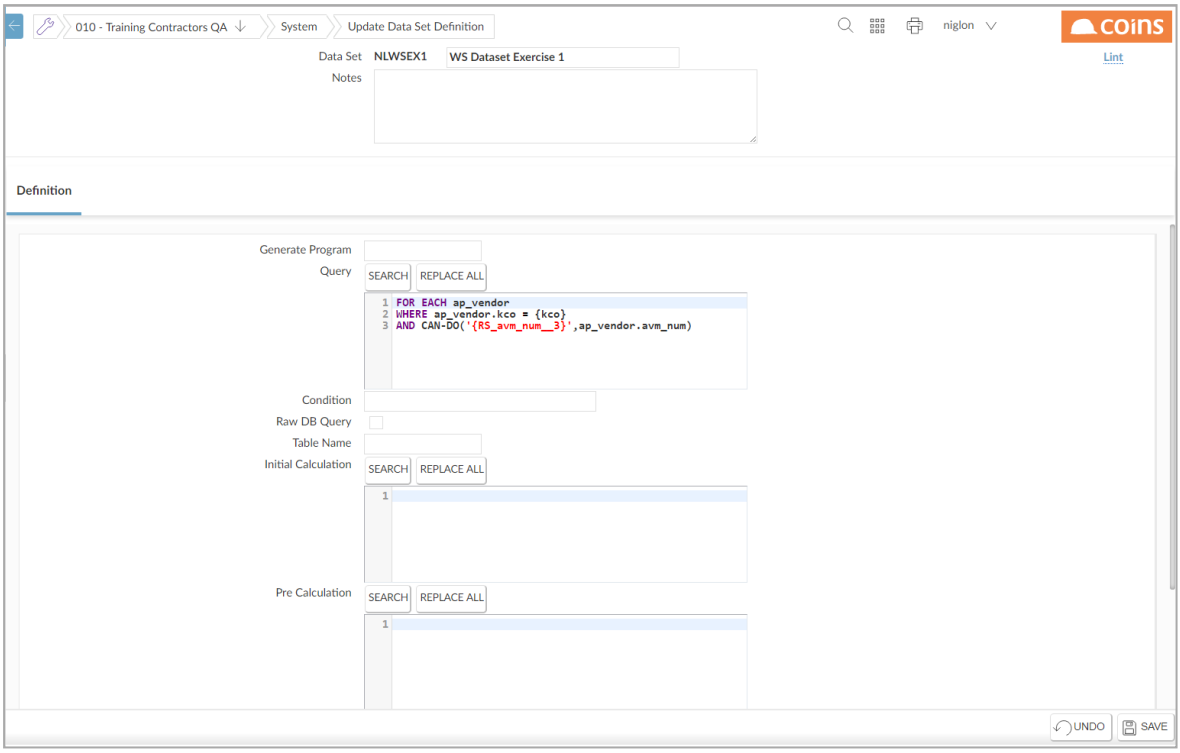
8.1.2 Dataset

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 

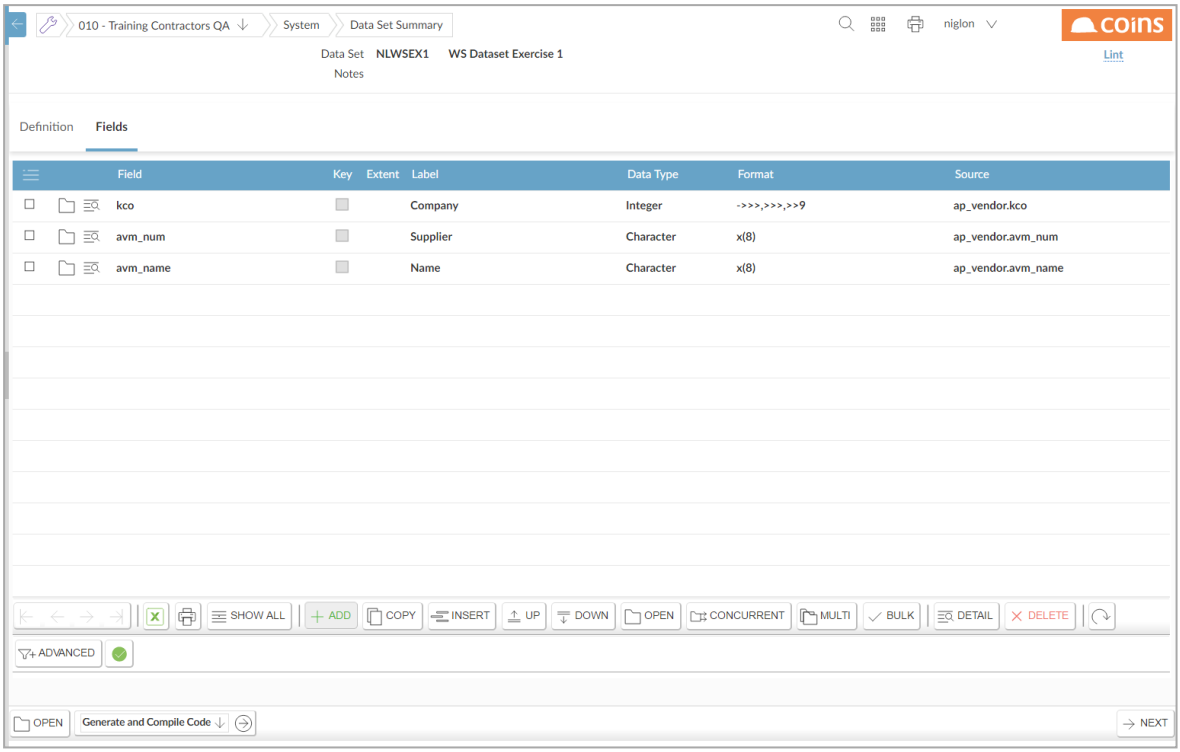
Add the fields

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.



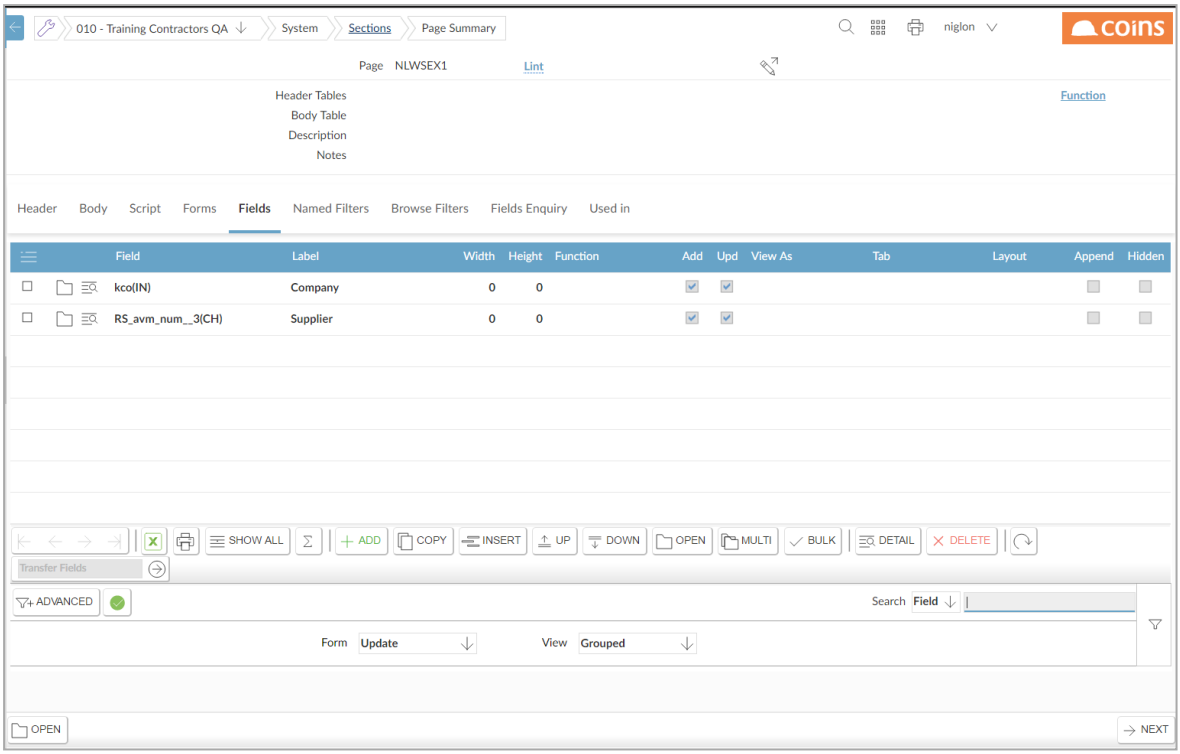
8.1.3 Page

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).

> Note: 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.

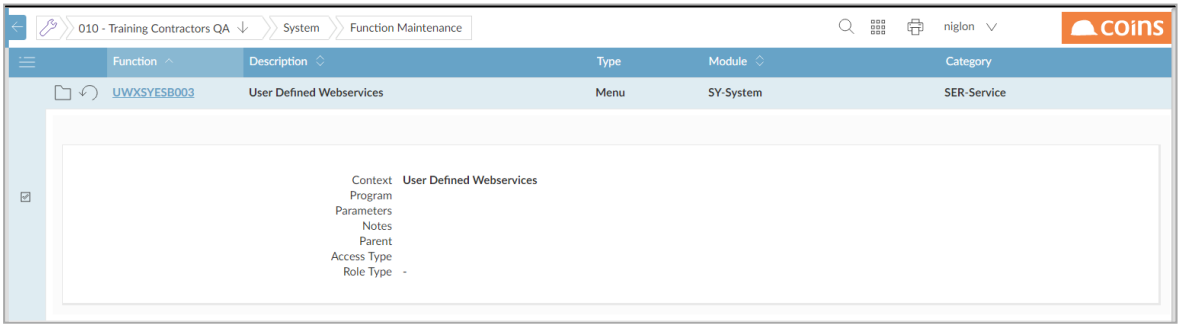
Field names can be followed by (XX) where XX is either IN,DE,DA,CH,LO meaning Integer, Decimal, Date, Character or Logical data type respectively. **If omitted then character data type is assumed.**



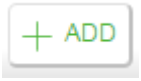
The page defines the INPUT schema for the BODY section of the message.

8.1.4 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.

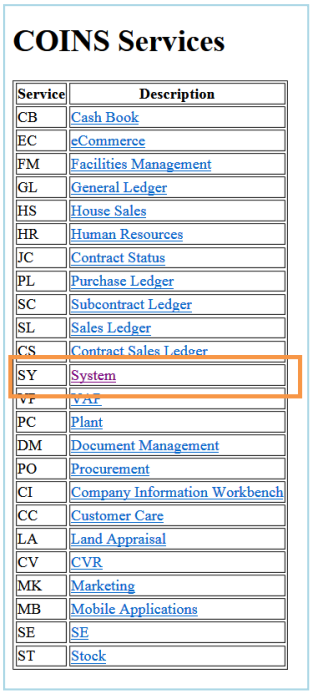


Open the Function and select the Menu Items Tab. If the function already exists, there may already be entries on this tab relating to Standard Used Defined Services issued by COINS (prefixed with %) or other User Defined Services created by your company.

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

Click  . The full details of the function are displayed.

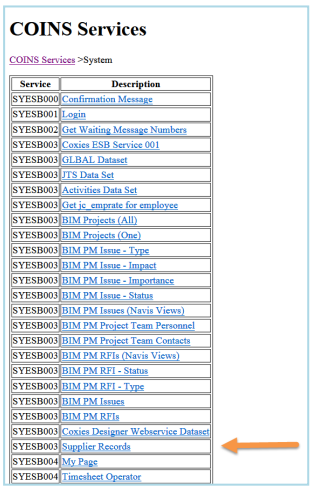
Navigate back to the COINS Services Menu and select the SY – System Hyperlink.



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. You new Function will be in this group.

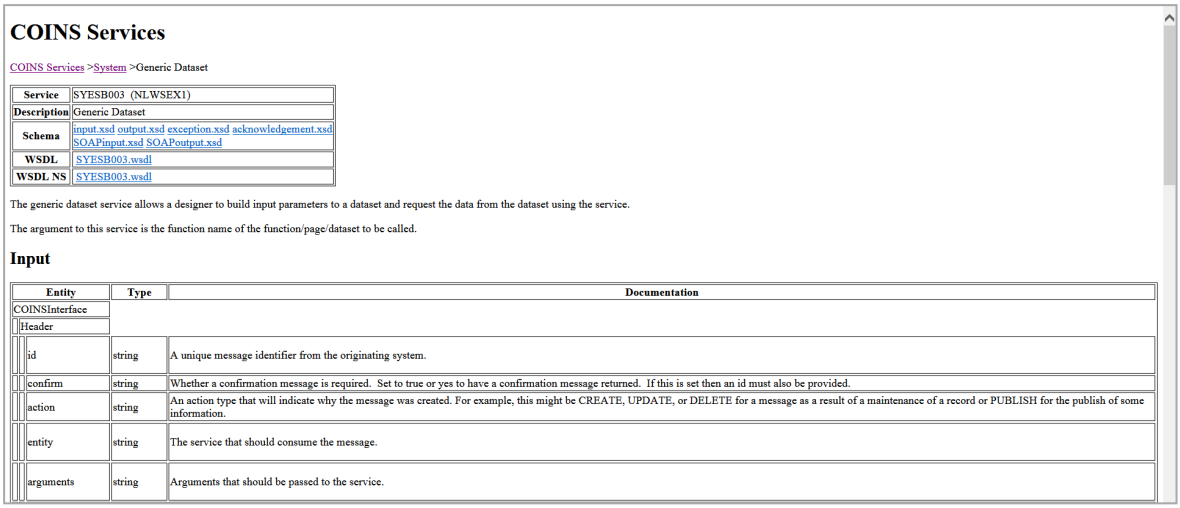


COINS Services

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 RFIs (Navis Views)
SYESB003	BIM PM RFI - Status
SYESB003	BIM PM RFI - Type
SYESB003	BIM PM Issues
SYESB003	BIM PM RFIs
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	SYESB003 (NLWSEX1)
Description	Generic Dataset
Schema	input.xsd output.xsd exception.xsd acknowledgement.xsd SOAPinput.xsd SOAPoutput.xsd
WSDL	SYESB003.wsdl
WSDL NS	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>
```

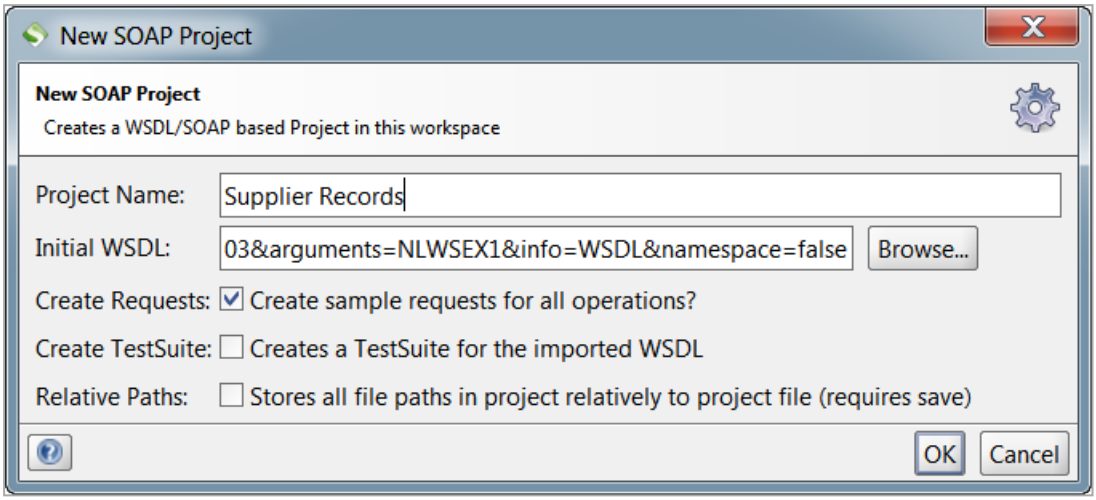
```
<Body>
  <Results>1</Results>
  <kco>99</kco>
  <RS_avm_num__3></RS_avm_num__3>
</Body>
</COINSInterface>
```

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>
```

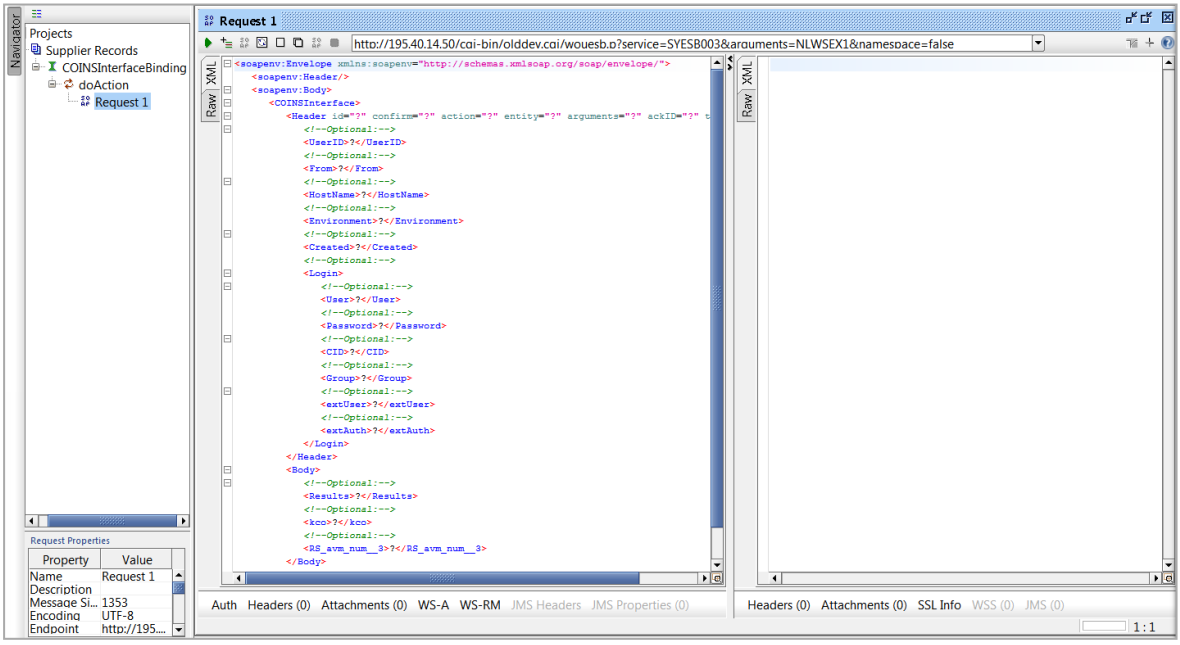
8.1.5 Testing the Service



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

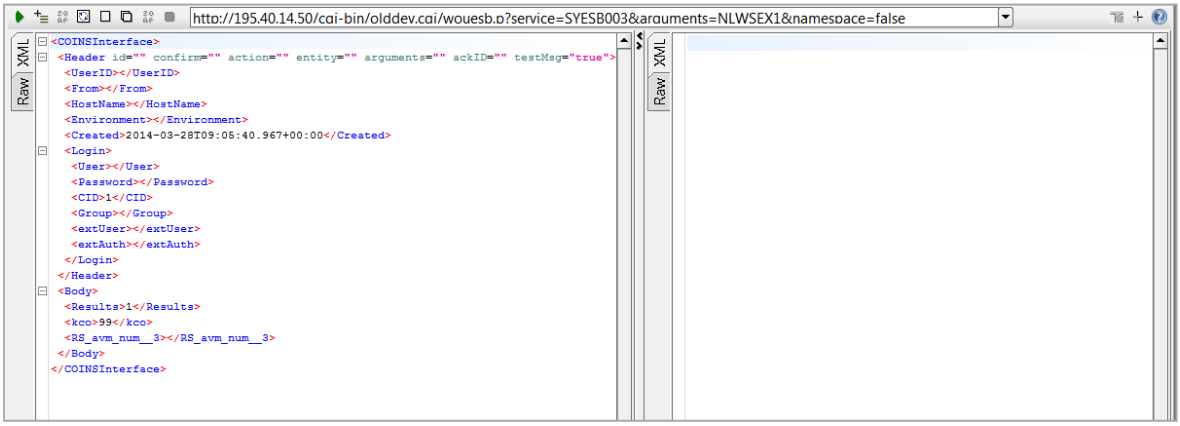
Sample

```
<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>
  <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>
```

Output

Entity	Type	Docur
COINSInterface		

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



Enter the hostname, Environment and username/password detail.

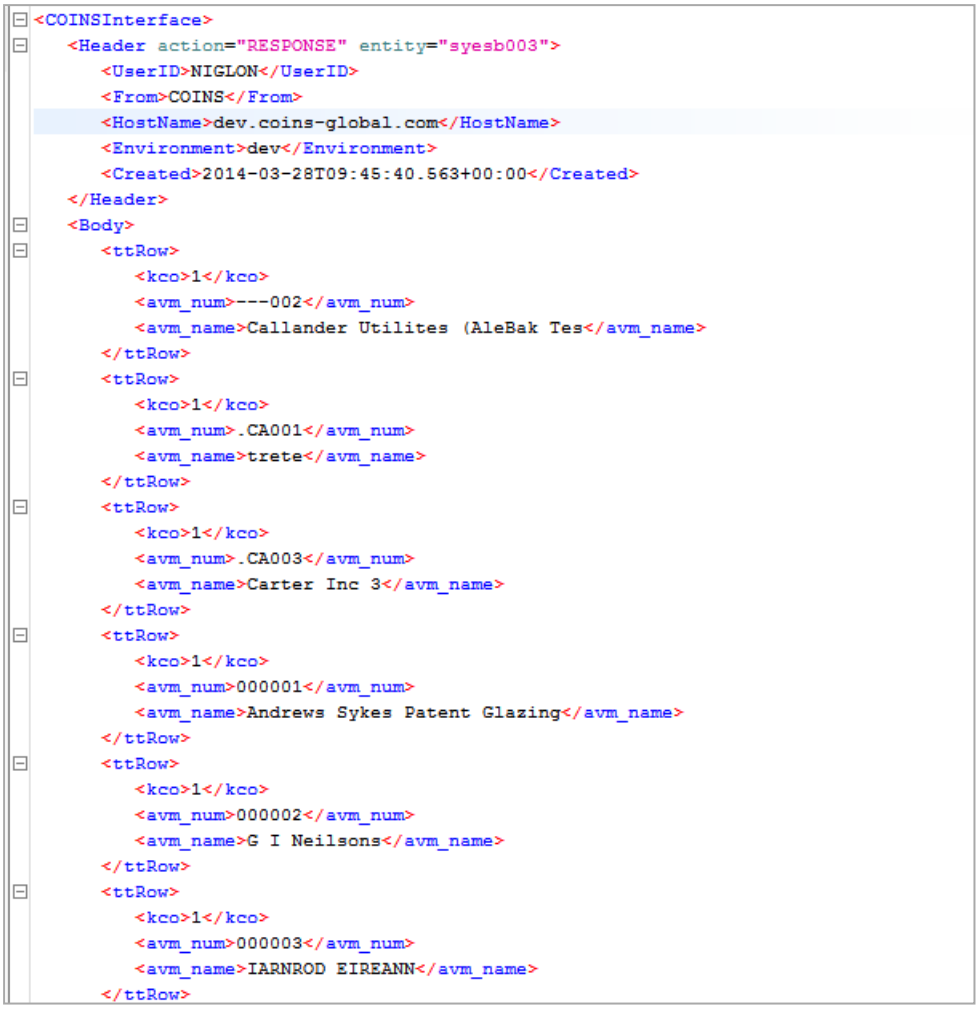
Delete the lines for Group, extUser and extAuth

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="syesb003">
      <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>
```

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.

8.2 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.

> Note: 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.

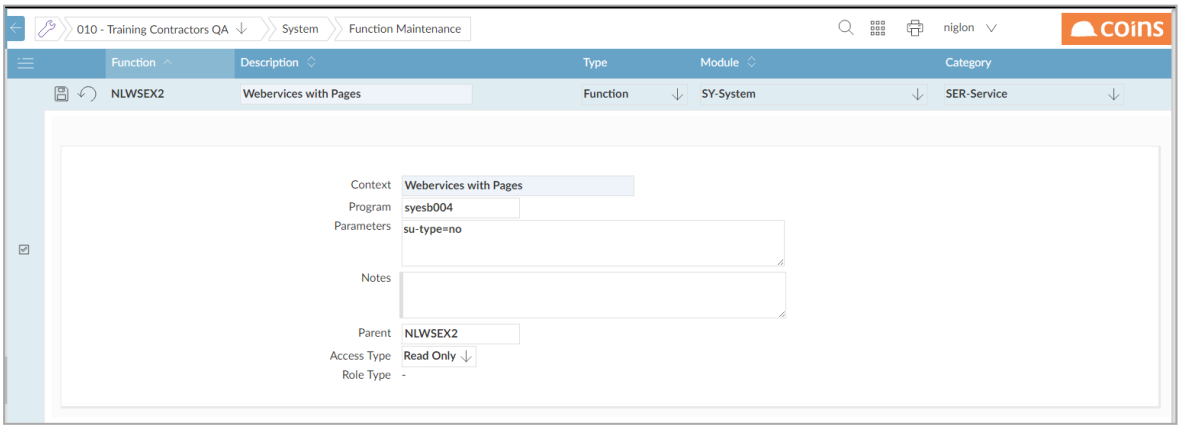
For a page webservice, two design components are required, the function(s) and a page. It is important that the function(s) and page have the same name.

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

8.2.1 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



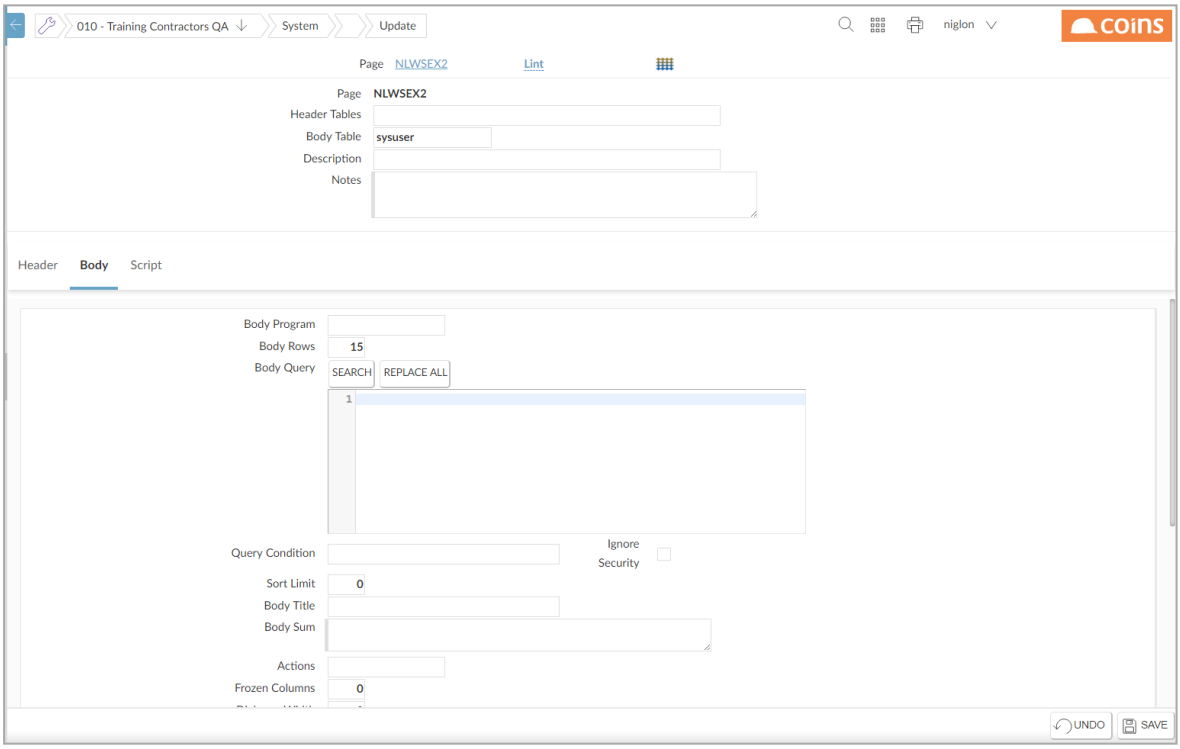
8.2.2 Page

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

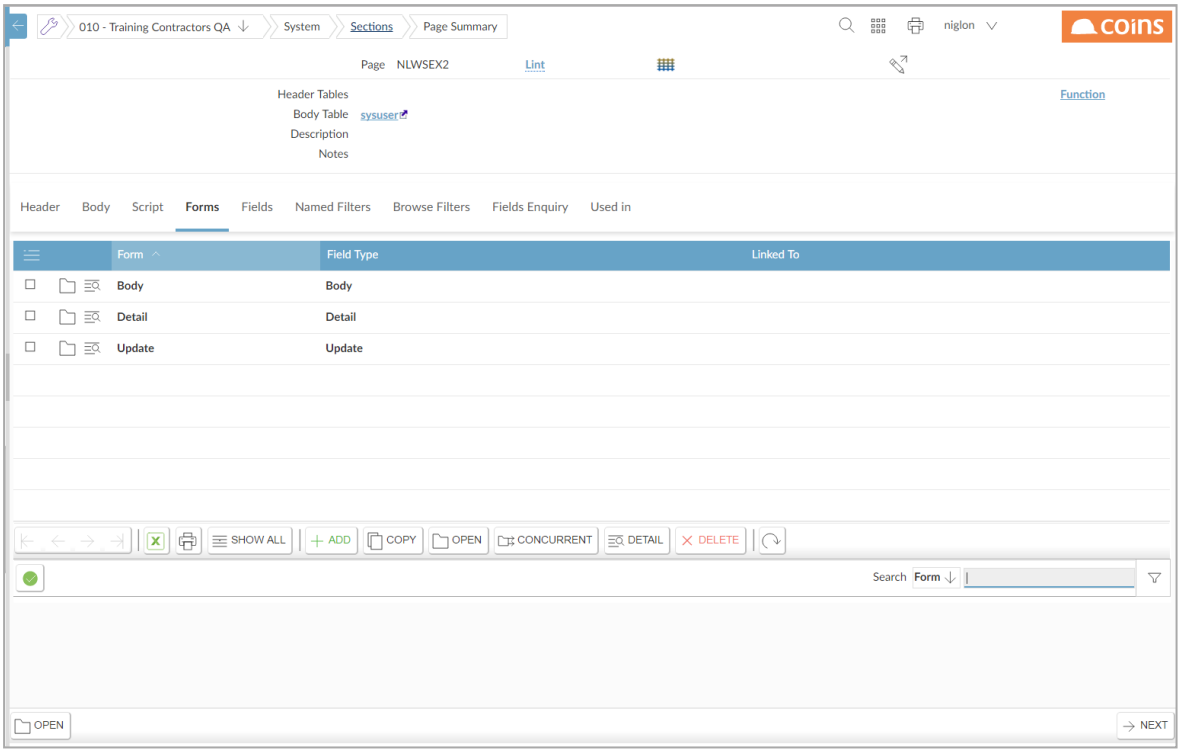
Body	Defines the fields that uniquely identify the record
Update	Defines the Input Schema
Detail	Defines the output Schema

Create new page called NLWSEX2.

> Note: 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

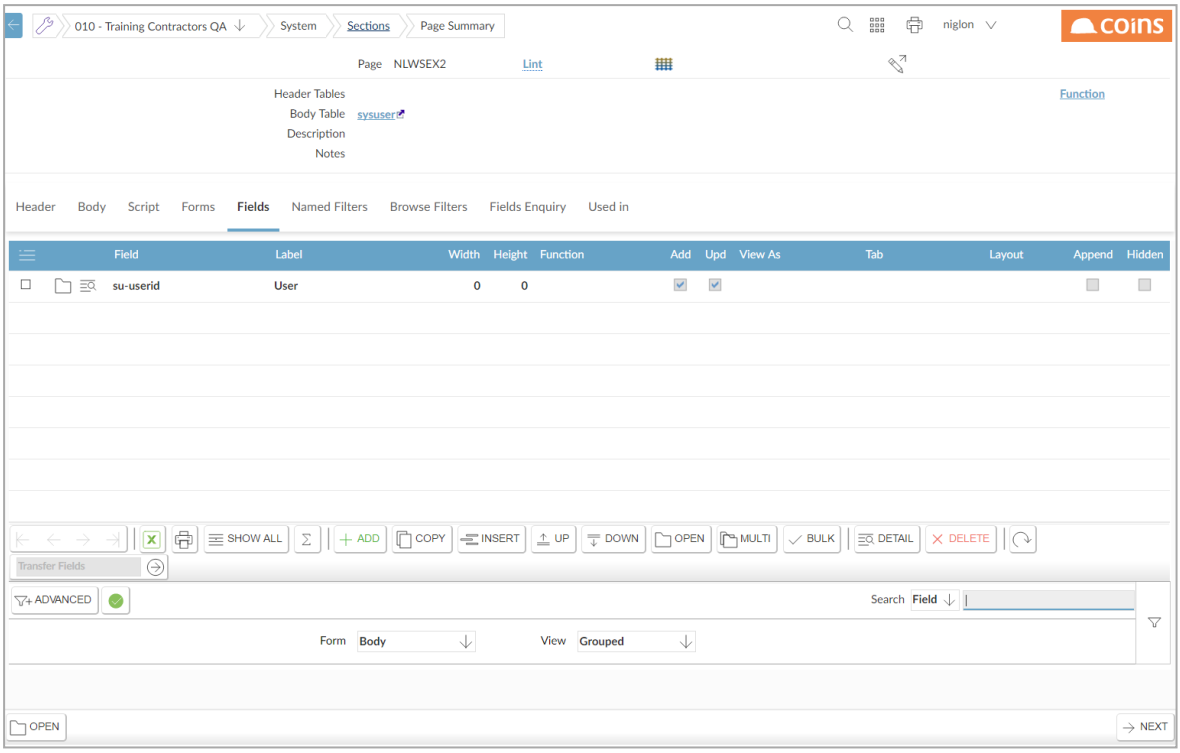


8.2.2.1 Body Form

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.

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



8.2.2.2 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

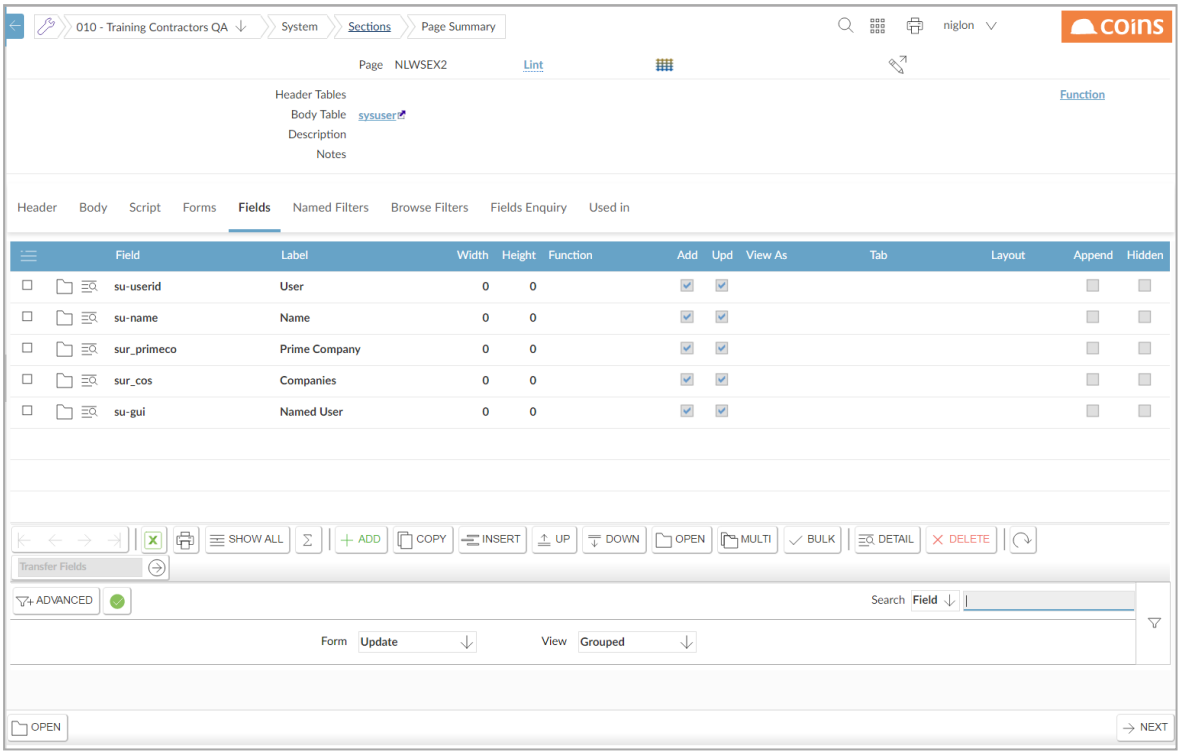
su-name

sur-primeco

su-cos

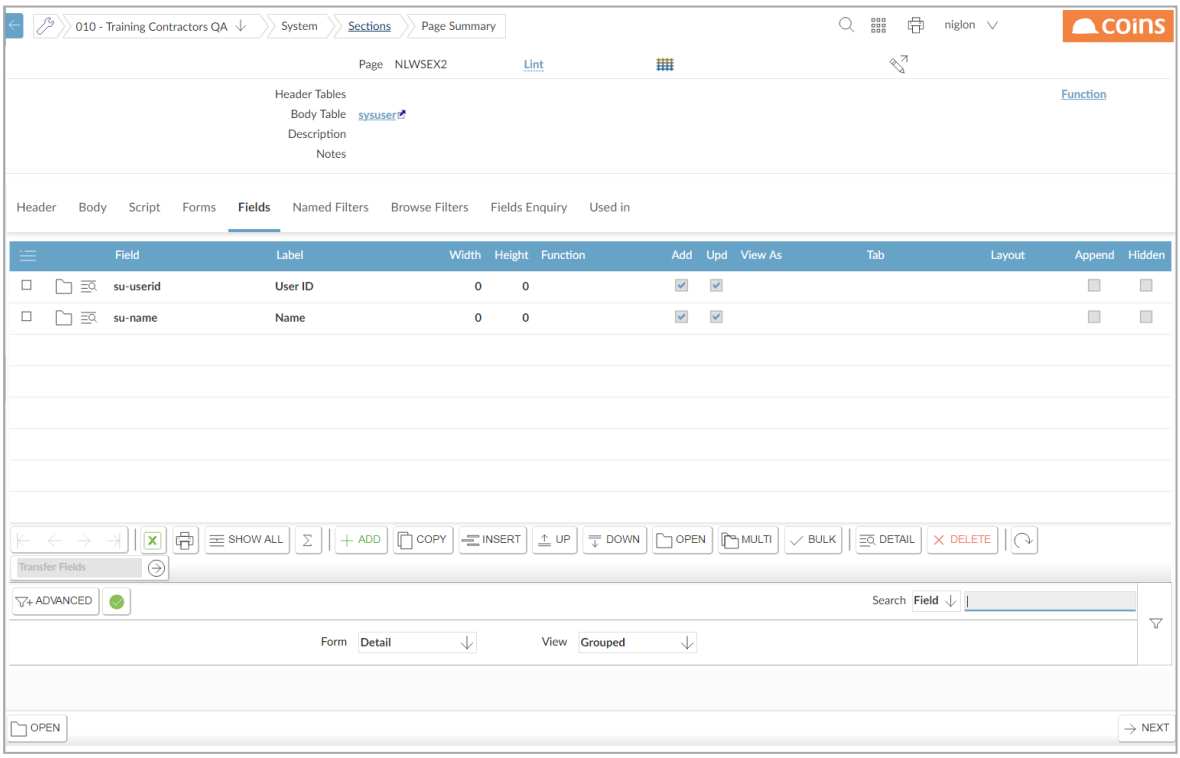
su-gui

Add these fields to the Update Form.



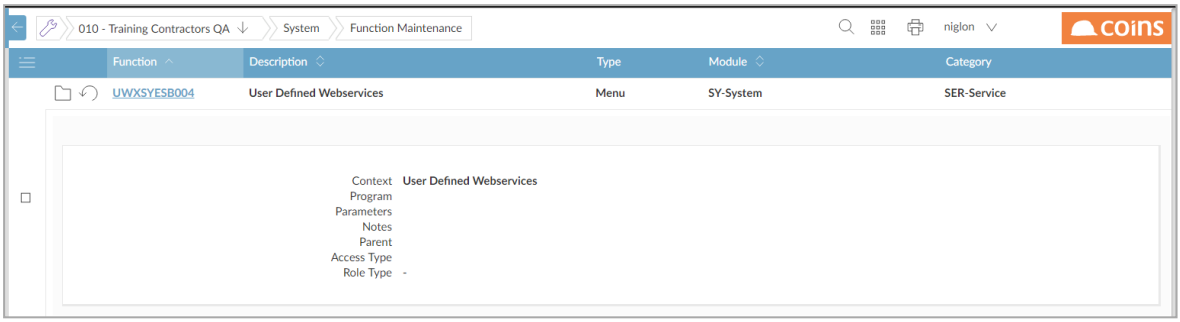
8.2.2.3 Detail Form

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



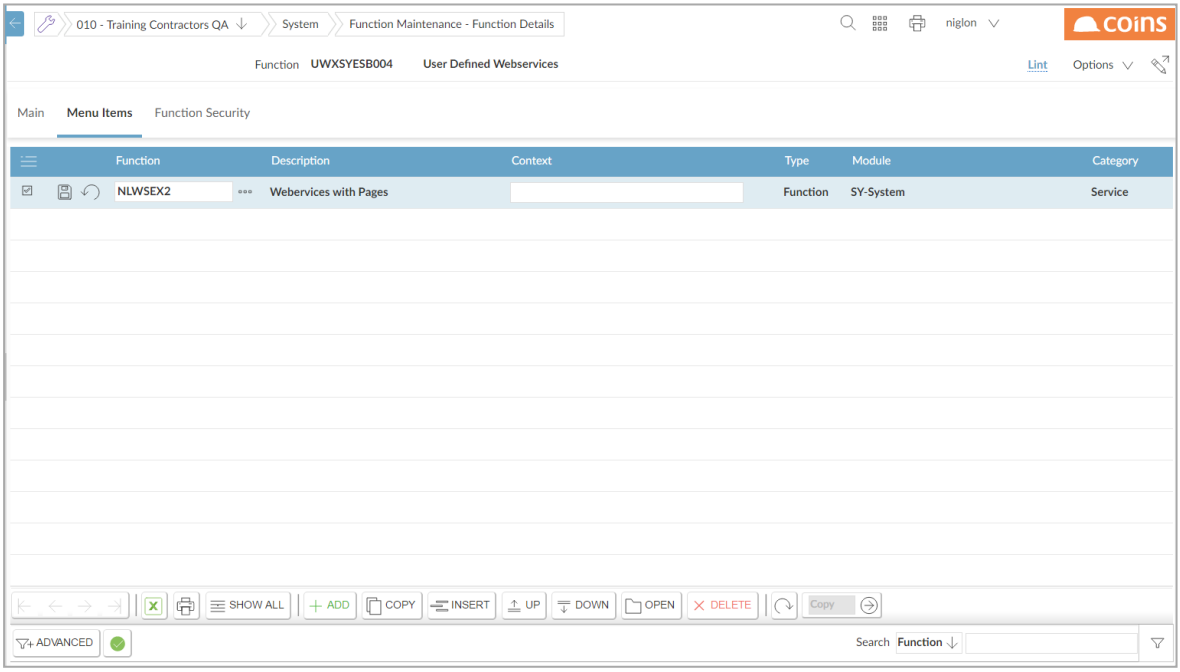
8.2.3 Adding the Function to the Web services Menu

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.

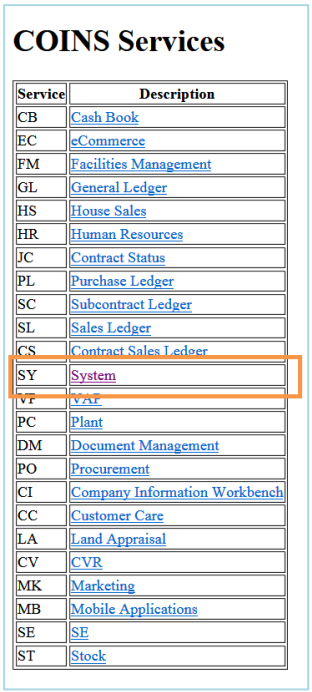


Open the Function and select the Menu Items Tab. If the function already exists, there may already be entries on this tab relating to Standard Used Defined Services issued by COINS (prefixed with %) or other User Defined Services created by your company.

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



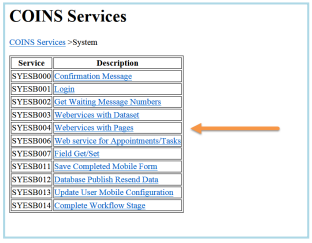
Navigate back to the COINS Services Menu and select the SY – System Hyperlink.



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
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 SYESB00 entries. You 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	Webervices with Dataset
SYESB004	Webervices 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

Select the Hyperlink to view the service schema.

COINS Services

COINS Services >System >Generic Page

Service	SYESB004 (NLWSEX2)
Description	Generic Page
Schema	input.xsd output.xsd exception.xsd acknowledgement.xsd SOAPinput.xsd SOAPoutput.xsd
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 trnsaction 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.
Login		
User	string	The COINS user to be used to consume the message. If ommitted then the user set in SY parameter ESBUID will be used. If user is specified then a password must also be given.
[illegible]	[illegible]	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:

Sample

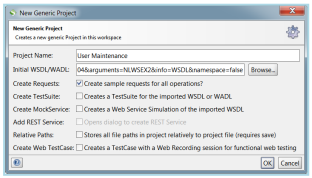
```
<COINSInterface>
 <Header id="" confirm="" action="" entity="" arguments="" ackID="" testMsg="true">
  <UserID></UserID>
  <From></From>
  <HostName></HostName>
  <Environment></Environment>
  <Created>2014-04-08T14:02:17.947+01:00</Created>
  <Login>
   <User></User>
   <Password></Password>
   <CID></CID>
   <Group></Group>
   <extUser></extUser>
   <extAuth></extAuth>
  </Login>
 </Header>
 <Body>
  <sysuserRow rsp_action="" id="">
   <su-userid></su-userid>
   <su-name></su-name>
   <sur_primecc>99</sur_primecc>
   <sur_cos></sur_cos>
   <su-quitrtue></su-qui>
  </sysuserRow>
 </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>2014-04-08T14:02:18.055+01:00</Created>
 </Header>
 <Body>
  <sysuserRow rsp_action="" id="">
   <su-userid></su-userid>
   <su-name></su-name>
  </sysuserRow>
 </Body>
</COINSInterface>
```

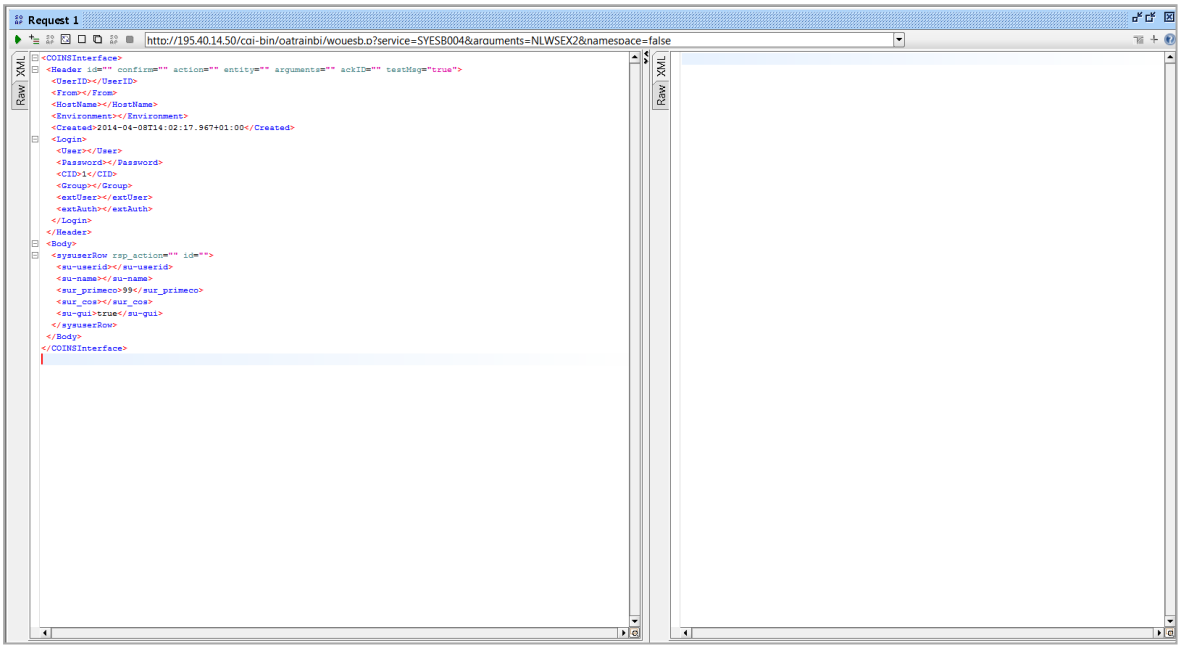
8.2.4 Testing the Service



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="">
```

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

A	Add
U	Update
D	Delete

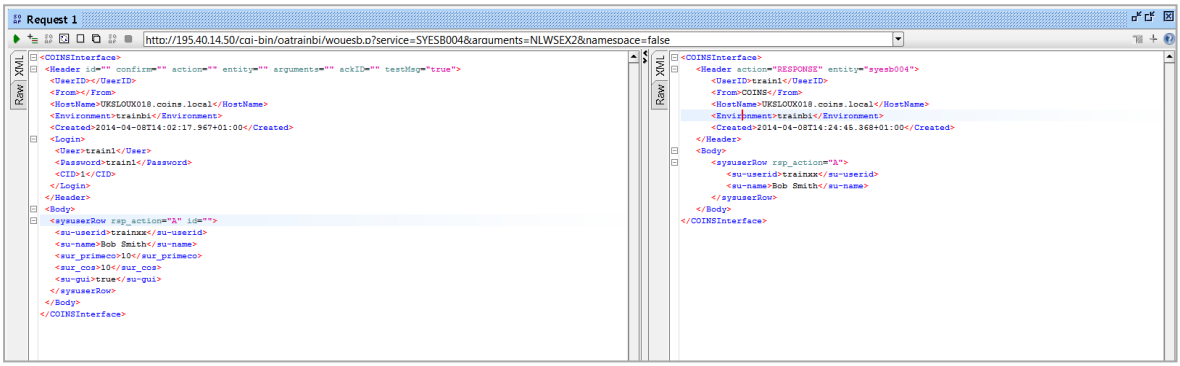
The id field can contain an id record that will be useful for identifying issues when updating more than one record. We will use this later in the exercise.

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

For example:

Send the message and correct any exceptions if they occur.

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



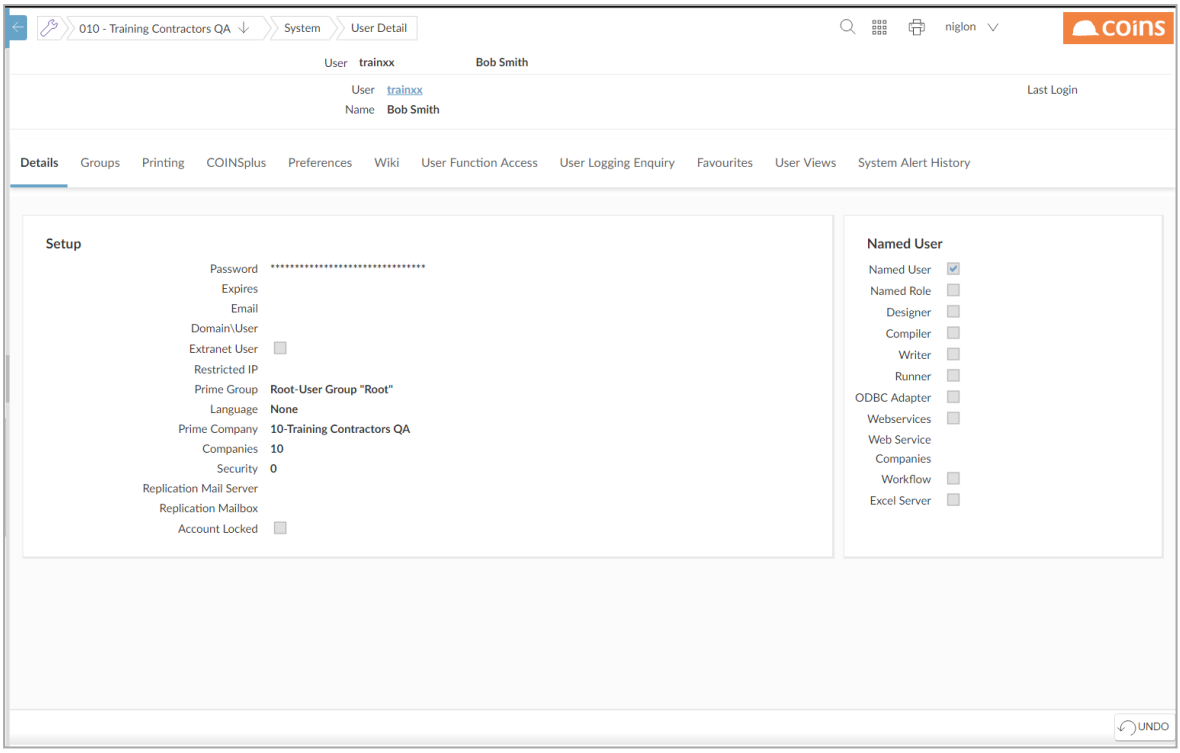
However, if we check the User records in User Maintenance, no new User has been created.

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

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:



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.

8.2.4.1 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 userid'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:

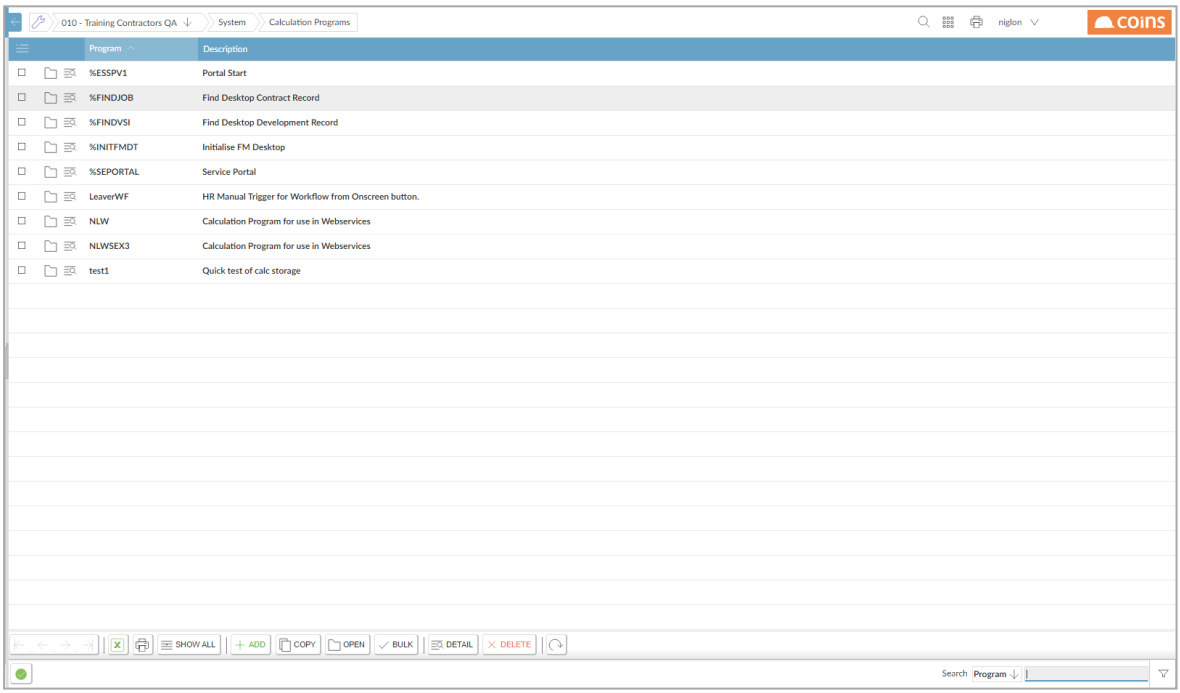
The exception will always give the row number that failed, but with an ID specified, the ID will also appear in the exception making it easier for me to identify the record that needs to be amended. Only when all records in the input message are correct will any changes be applied to the database.

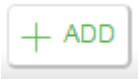
8.3 Calculation Programs

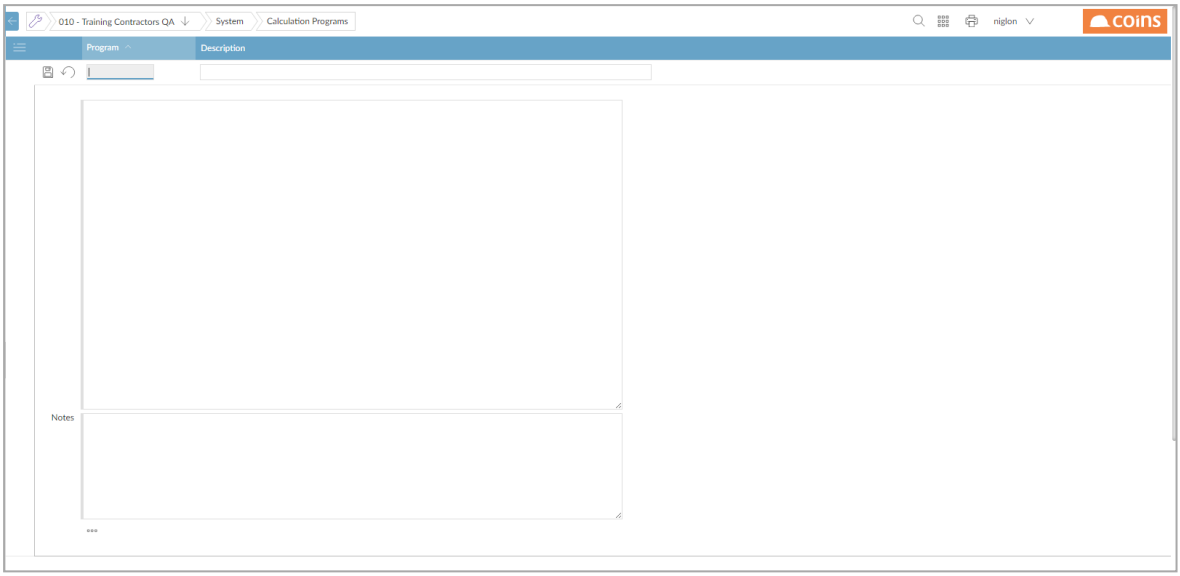
COINS provides the functionality for user-defined programs to be created and called by Web services offering greater flexibility for data manipulation and retrieval.

8.3.1 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}

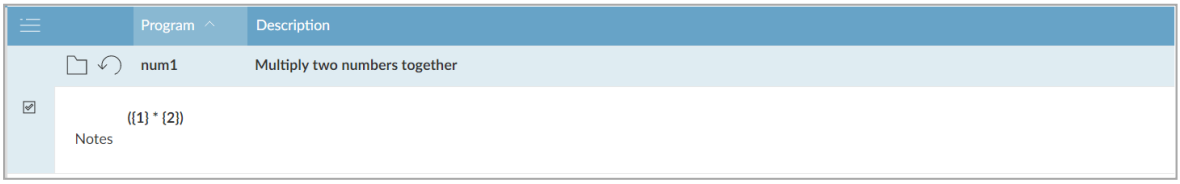
For example, if we wanted a calculation to multiply two different numbers, we would write this as:

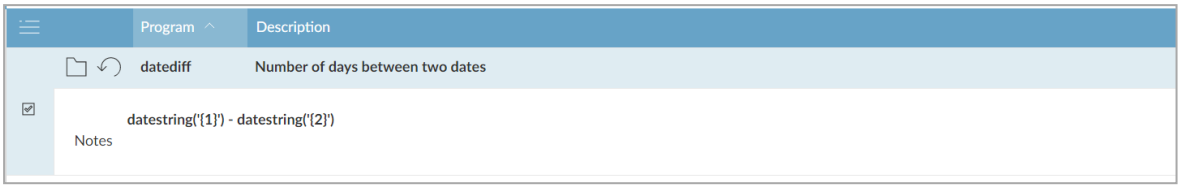
{1} * {2}

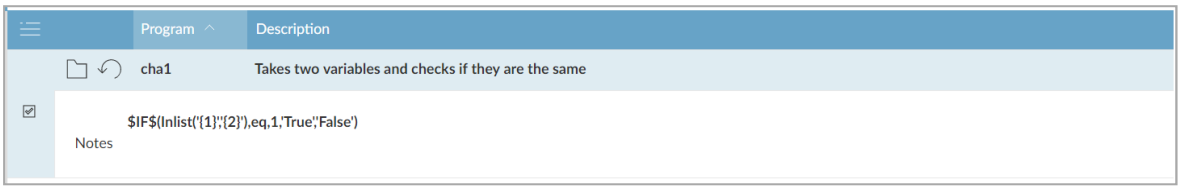
Once a variable is defined, it can be re-used in a calculation and therefore need only be passed into the calculation once.

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

Three examples of a calculation program definition are shown below:







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.

8.3.2 Testing Calculation Programs (run, run$)

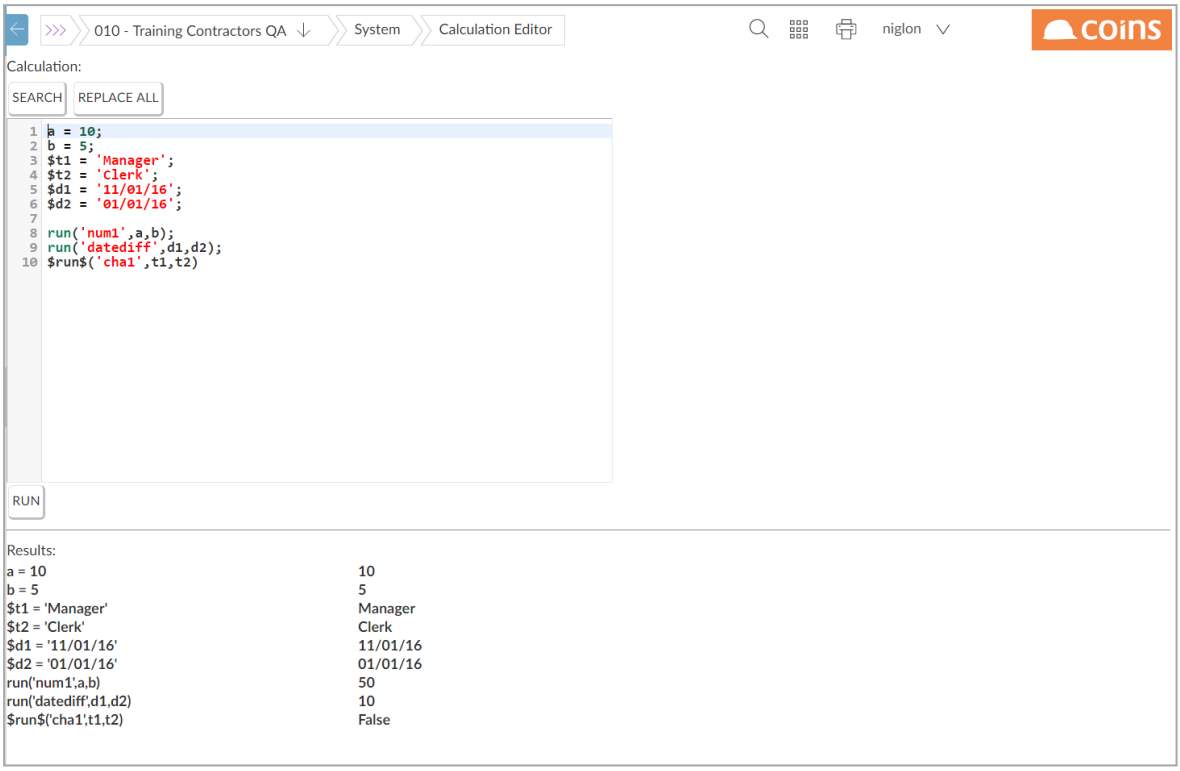
Once a calculation has been saved then it can be run by using:

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

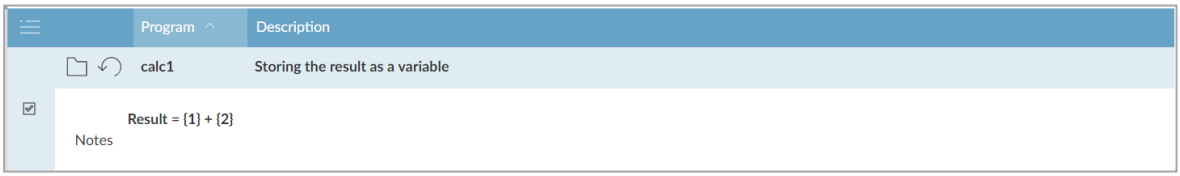
or

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

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



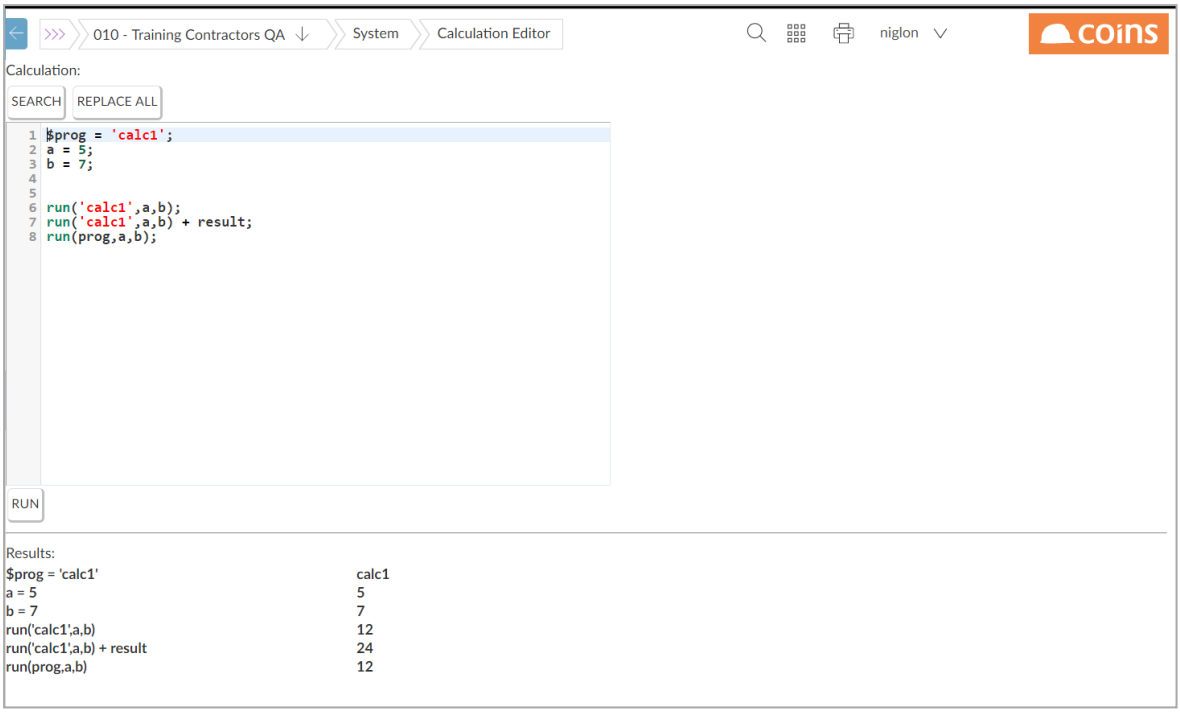
Calculation programs can be defined to both use and create variables. For example:



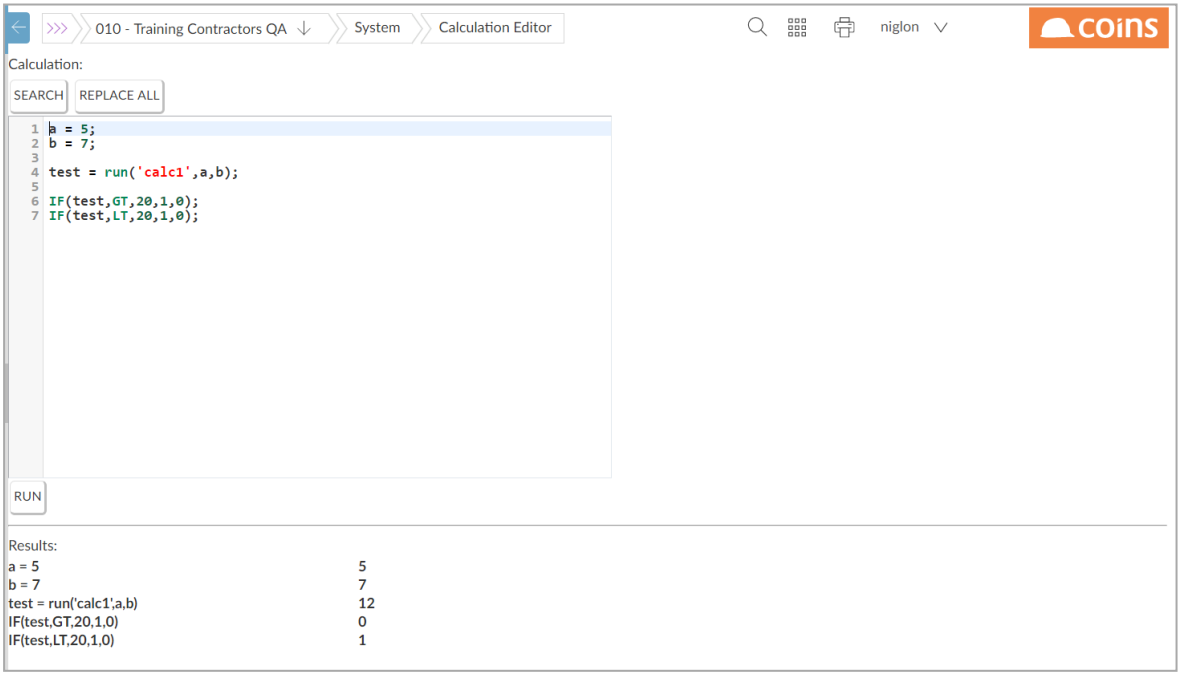
This will store the value of {1} + {2} in a variable called result. This can then be used directly in subsequent calculations.

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

Examples of these are shown below:

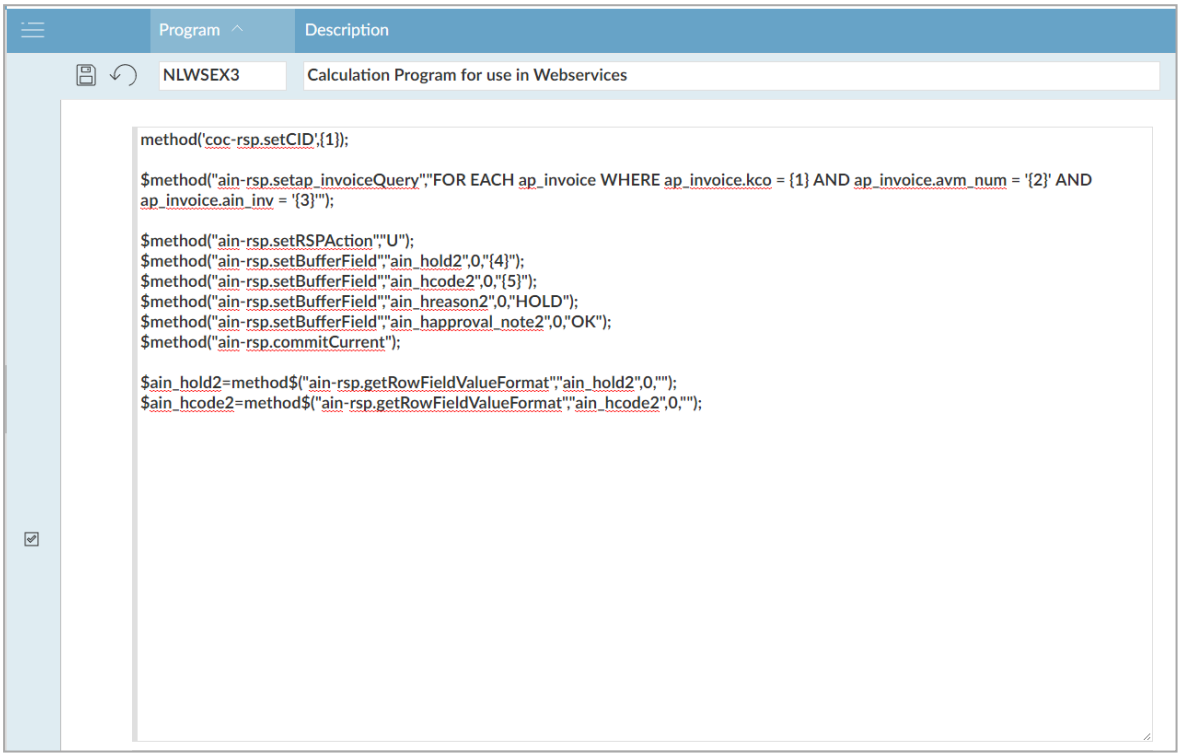


It is also possible to use calculation programs directly within other calculations:



8.3.3 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

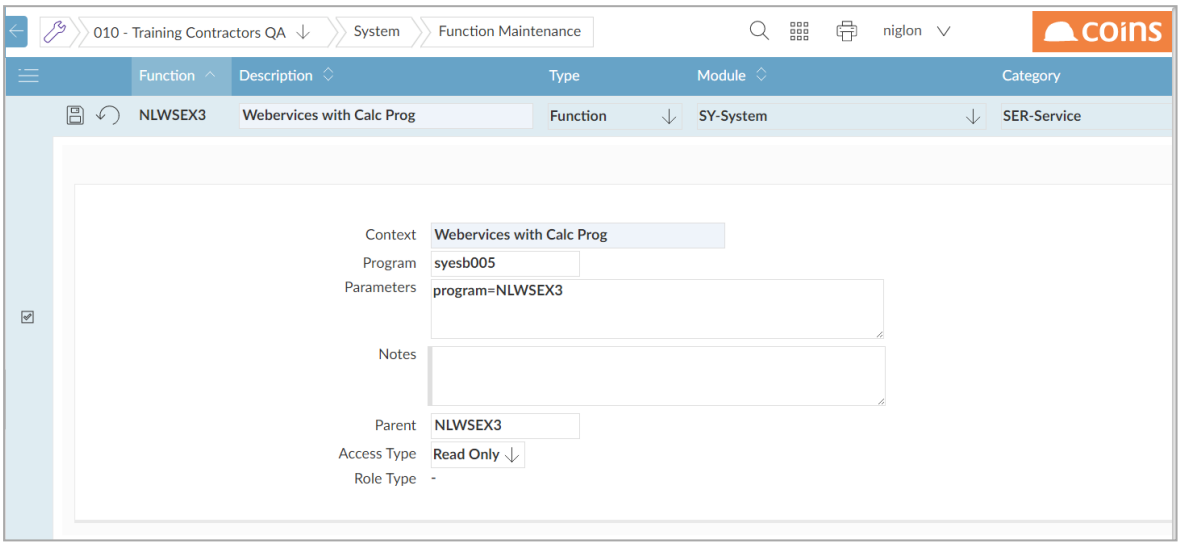
The calculation will look up the appropriate invoice, set the hold flag and code, and return two variables which will be used in the output message as confirmation of the changes.

8.3.3.1 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.

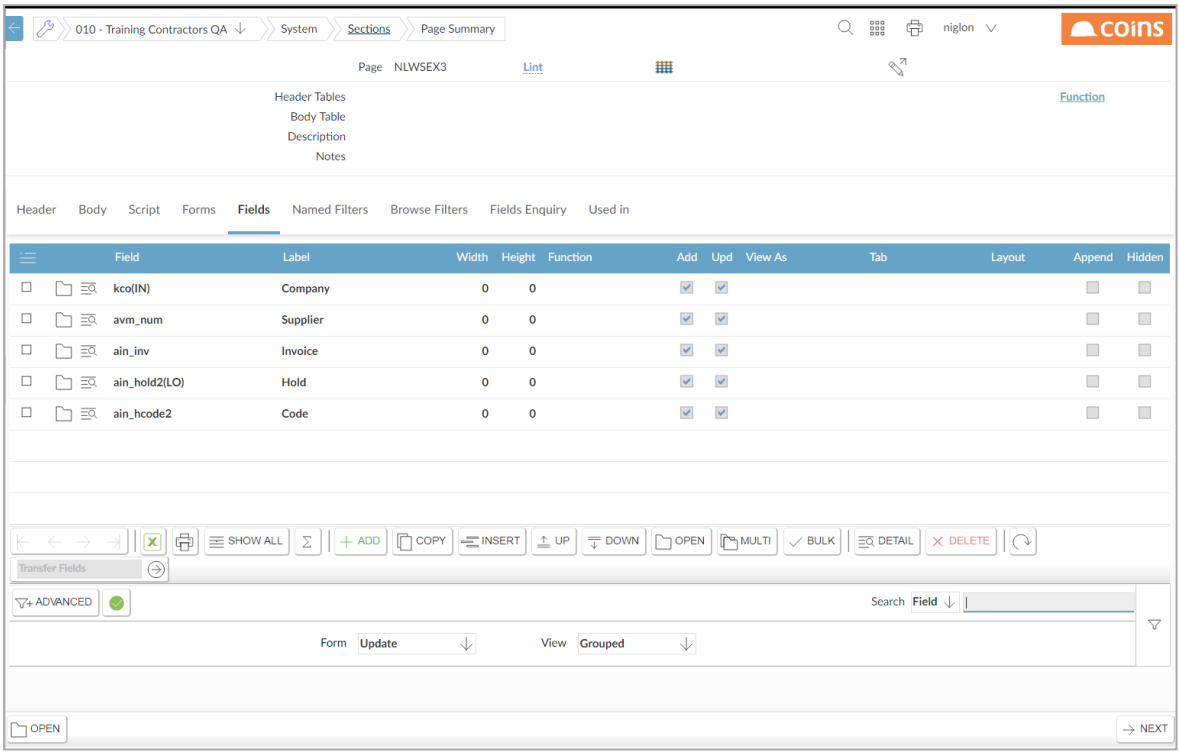
Add the parameter program=NLWSEX3



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.

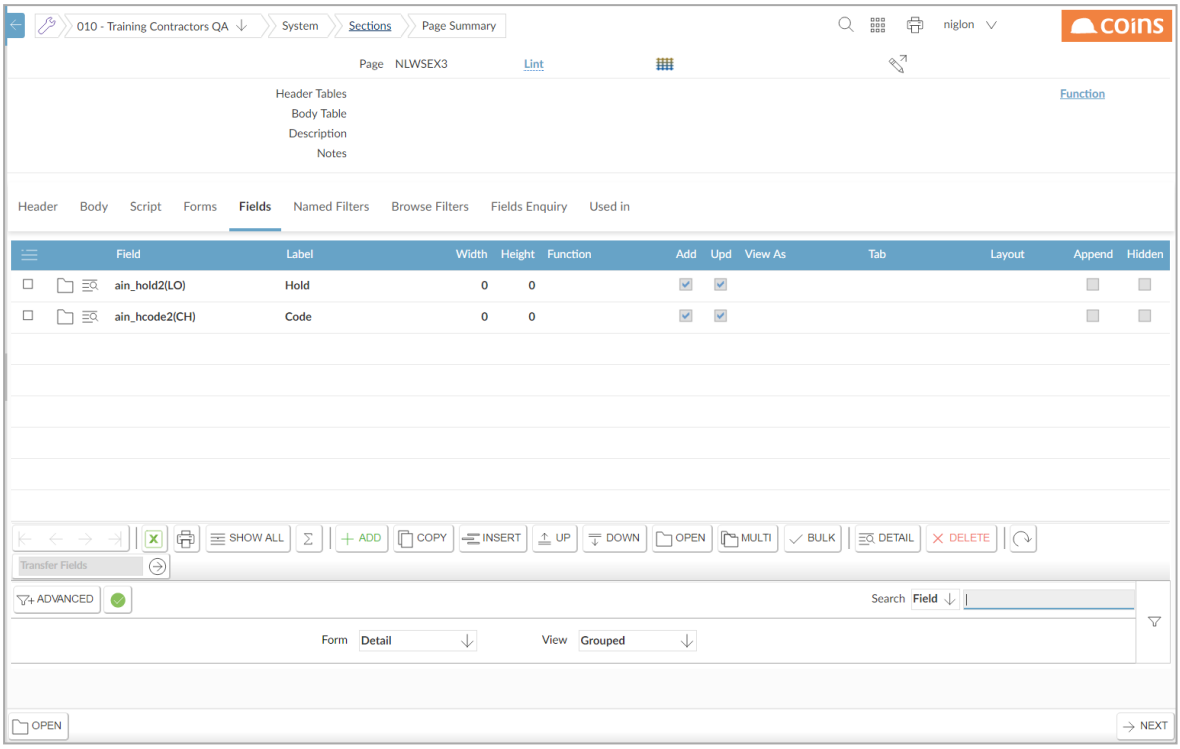
8.3.3.2 Create the Page

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



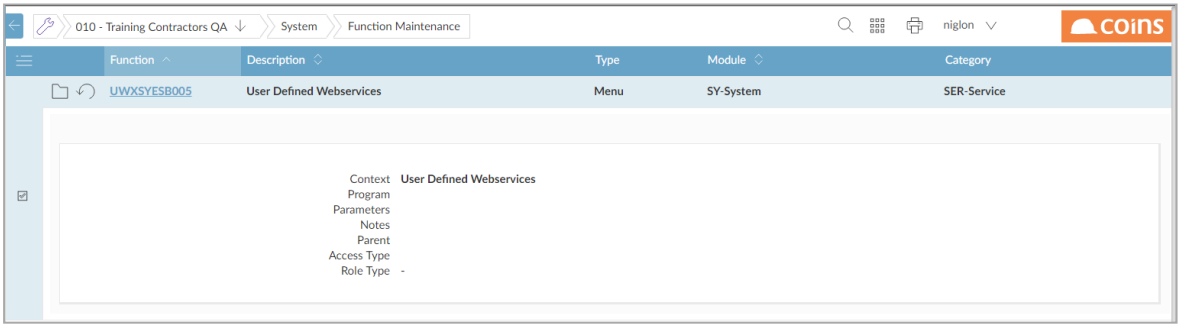
The fields will relate to the input parameters in the calculation program and should be defined in the sequence they are used {1}=kco, {2}=avm_num etc…

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

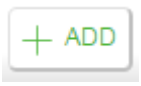


8.3.3.3 Adding the Function to the Web services Menu

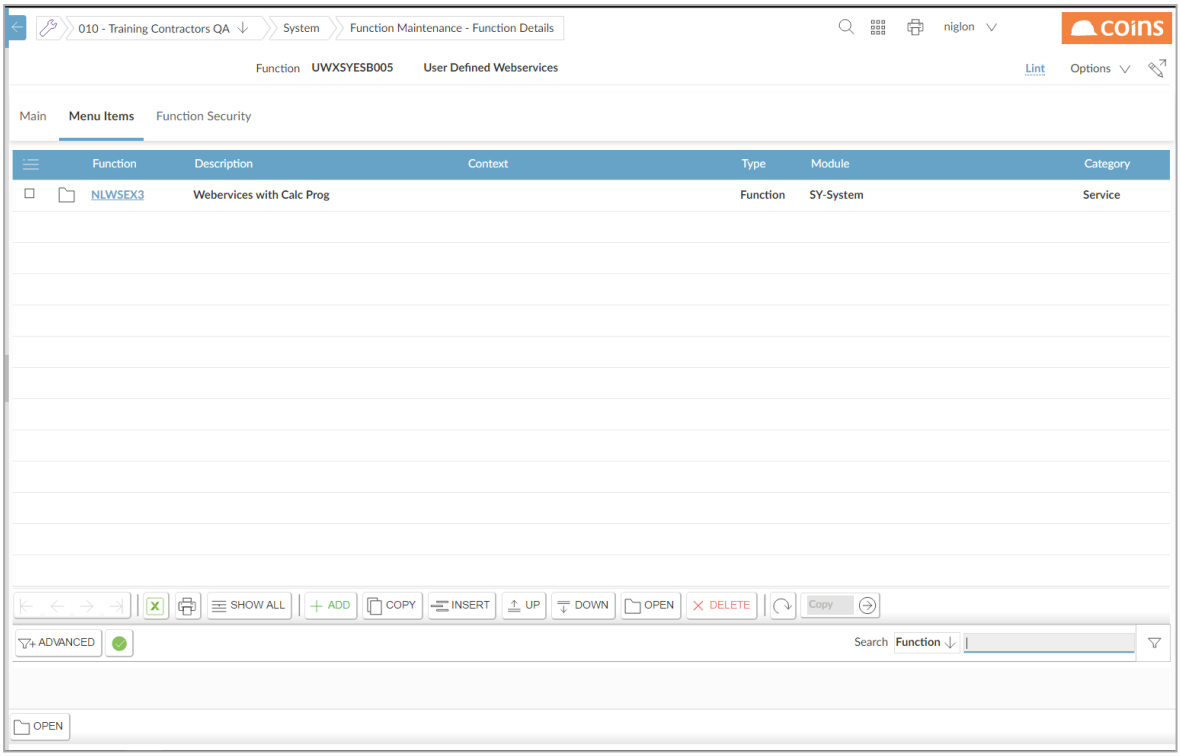
To run the Page Web Service, first check that the menu function UWXSYESB005 exists. If it does not exist, then you need to create it.



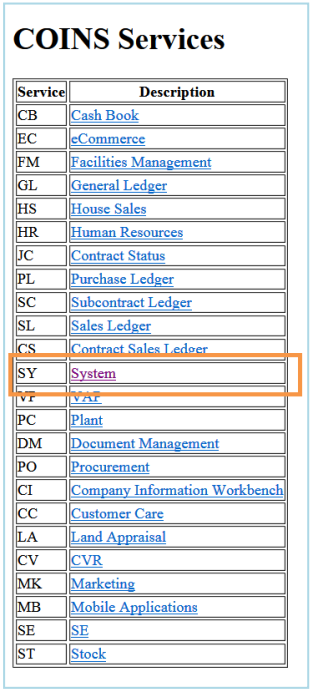
Open the Function and select the Menu Items Tab. If the function already exists, there may already be entries on this tab relating to Standard Used Defined Services issued by COINS (prefixed with %) or other User Defined Services created by your company.

Click 

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



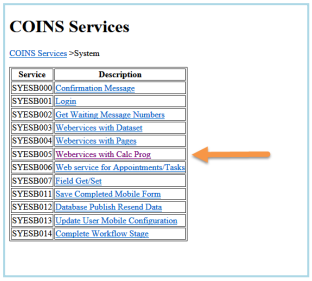
Navigate back to the COINS Services Menu and select the SY – System Hyperlink.



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 SYESB00 entries. You new Function will be in this group.

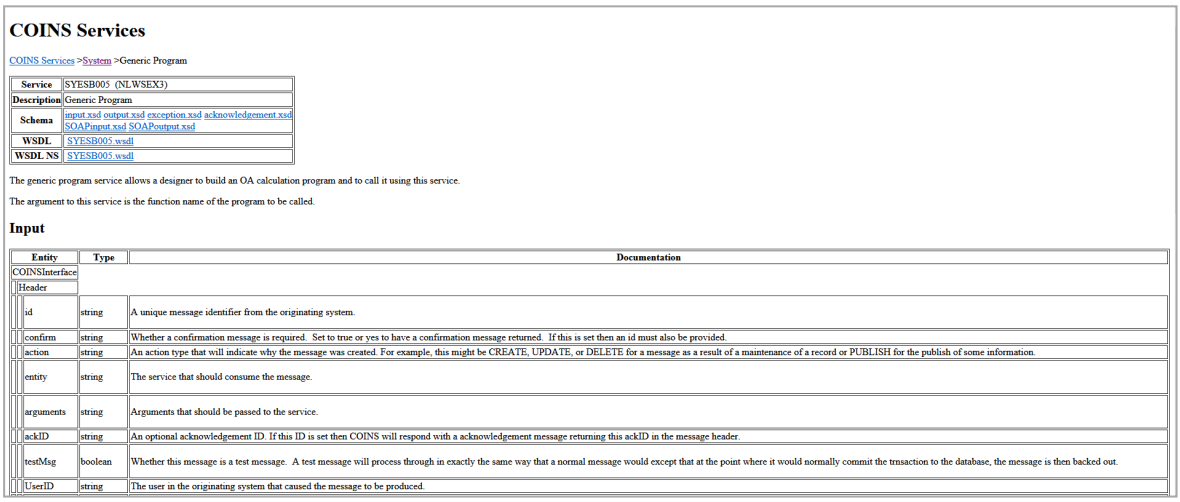


COINS Services

COINS Services >System

Service	Description
SYESB000	Confirmation Message
SYESB001	Login
SYESB002	Get Waiting Message Numbers
SYESB003	Webervices with Dataset
SYESB004	Webervices with Pages
SYESB005	Webervices with Calc Prog
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

Select the Hyperlink to view the service schema.



COINS Services

COINS Services >System >Generic Program

Service	SYESB005 (NLWSEX3)
Description	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 trnsaction 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

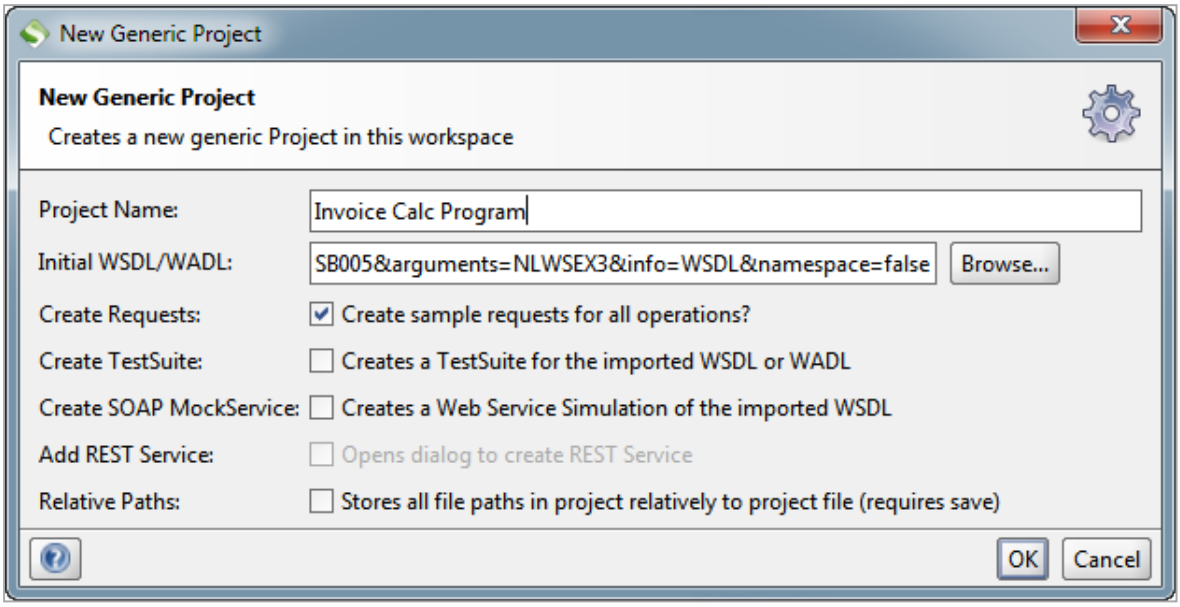
```
<COINSInterface>
 <Header id="" confirm="" action="" entity="" arguments="" ackID="" testMsg="true">
  <UserID></UserID>
  <From></From>
  <HostName></HostName>
  <Environment></Environment>
  <Created>2014-04-10T11:54:54.523+01:00</Created>
  <Login>
   <User></User>
   <Password></Password>
   <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>
```

8.3.4 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.

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.

Note: 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.

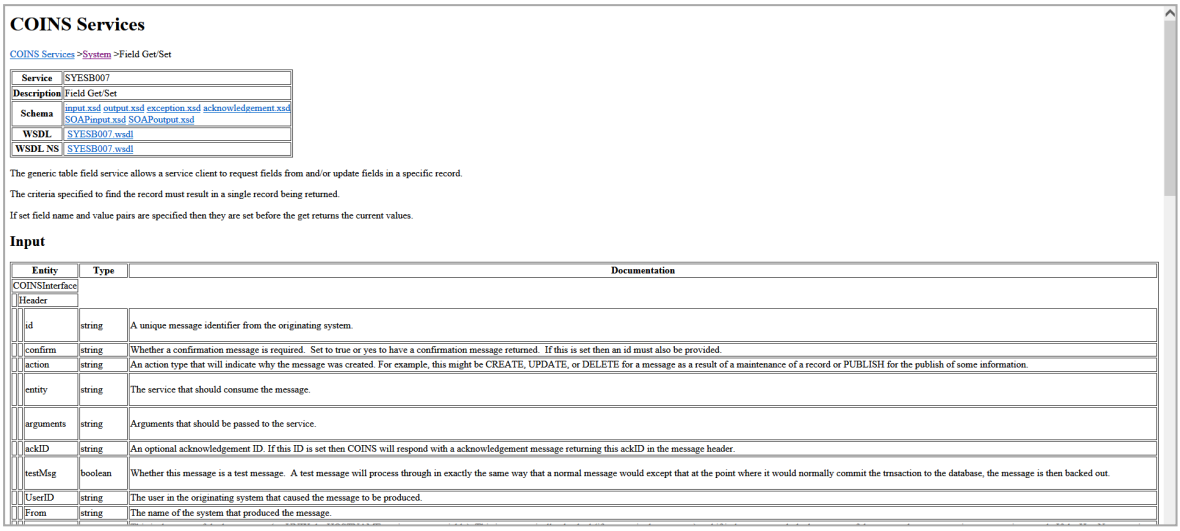
8.4 Get/Set

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.

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 SOAPinput.xsd SOAPoutput.xsd
WSDL	SYESB007.wsdl
WSDL NS	SYESB007.wsdl

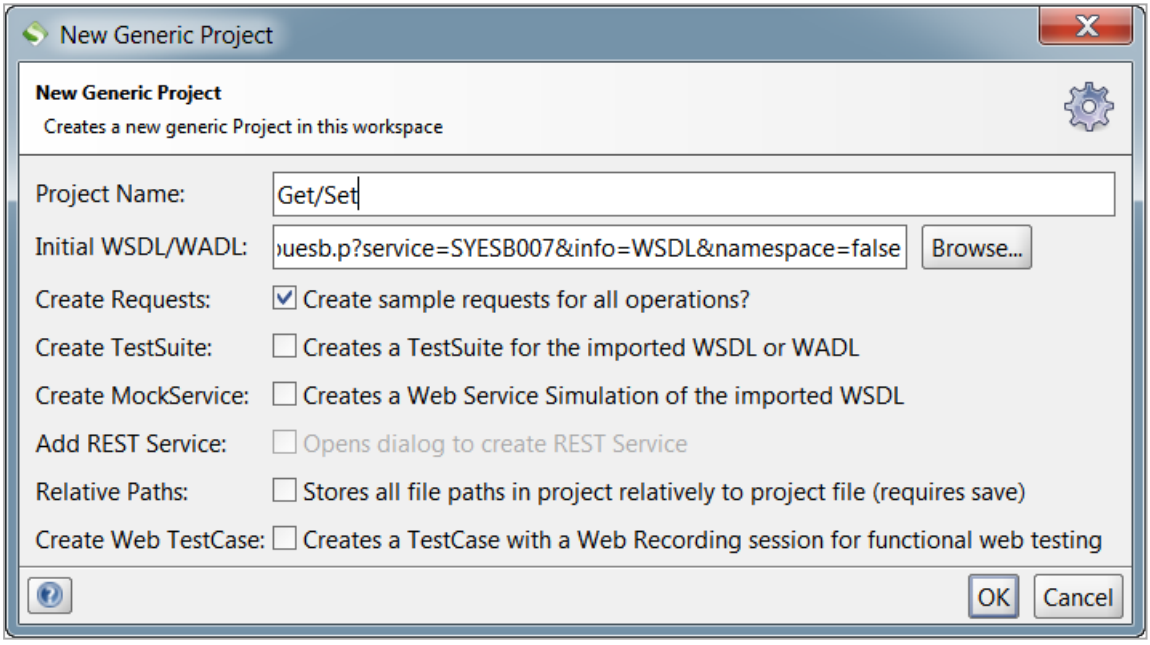
The generic table field service allows a service client to request fields from and/or update fields in a specific record.

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 trnsaction 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.

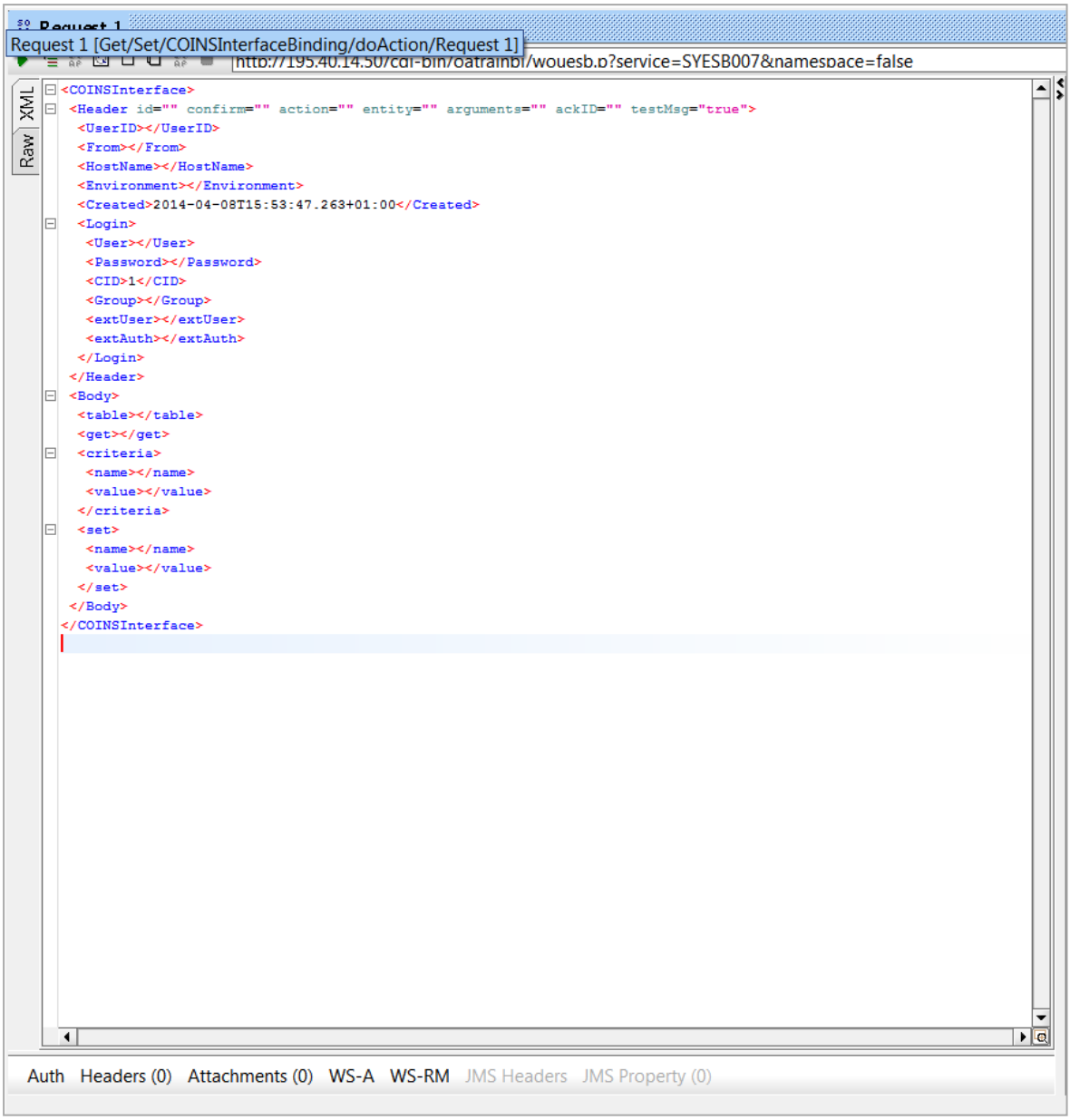


Create a new soapUI project and, from the service schema, copy the WSDL shortcut link into the initial WSDL field.

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 is should be set.

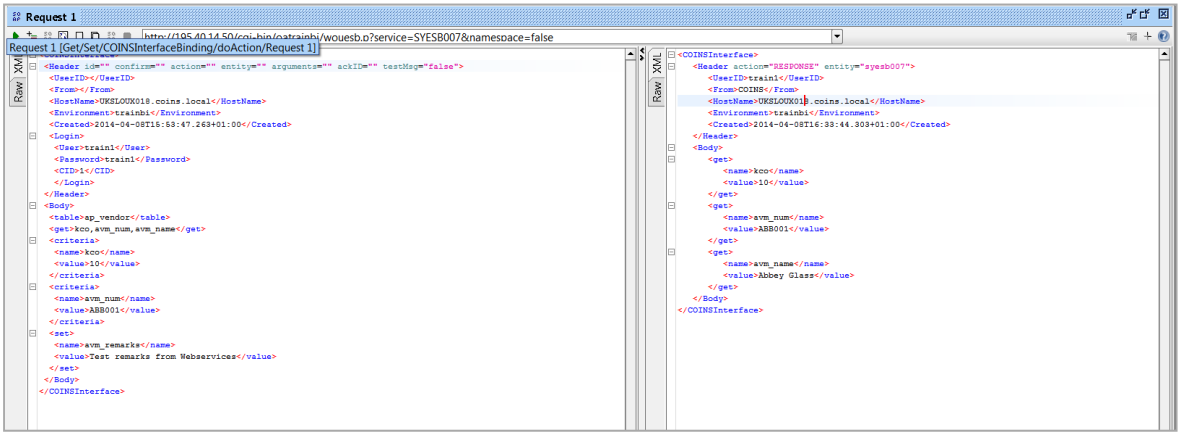
```
<set>
 <name></name>
 <value></value>
</set>
```

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.



Note: 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: