Web Services explained

Richard Perrins

30/03/2022

Contents

Chapter 1
What are Web Services

What are Web Services?

"*A web service is a functionality proposed by an application as a "service" to other applications through the internet. It is independent from underlying platforms or development languages used by client applications.*"

Web Services provide a way for:

1) External software to interact with Sage X3's data without the direct use of X3's Browser Interface

2) X3 to interact with External Sites to obtain or update data

As such, they enable Developers to extend the reach of X3 to interact with the outside world.

For the purposes of this session, there are three classes of Web Service:

Classic SOAP Web Services

This class of Web Services is a generic sort of Web Service which can be used to extract, insert or update information in the X3 database via calls to X3 code through a Classic SOAP Pool.

There are two sorts of Classic SOAP Web Service:

1) Object Web Services (the only verified/validated Objects are BPC, ITM and SOH)

2) Sub-program Web Services (e.g. AOWSIMPORT)

ADC Handhelds

This class of Web Services is used to interface X3 with Handheld devices – typically, for Warehouse Management or other Distribution and Manufacturing functions.

Historically, these were configured via the SafeX3 Console, but this class of Web Service has been deprecated for some time. As such, we'll not cover it in this piece.

However, a new version of ADC has been developed for Version 12 – known as "New ADCs" – we'll have a very brief look at these later…

RESTful Web Services

This class of Web Services allows X3 to interact with external sites via calls to a built-in API library within X3's 4GL and allows external sites to interact with X3 itself via X3's Representations.

Chapter 2
Meet Classic SOAP Web Services

Chapter 2.1
Setting up Classic SOAP Web Pools

Classic SOAP Web Service calls launch X3 4GL programs and are made through adonix/sadoss process pairs which communicate with X3 via dedicated Syracuse processes.

The number of Syracuse processes set up to handle Web Service calls across the X3 Solutions is set up in the Administration > Administration > Servers > Hosts option.

The Syracuse processes can be seen in Web Pool configuration and matched against the actual processes in Task Manager:

State

Host	Port	Process	Pid	Status	Endpoint	User login	Locale	Last action	Action By	Ws Qu
X3ERPV12SQLVM	0	W0	7128	Started	X3ERPV12_SEED	WSUSR	en-GB	AutoStart		
X3ERPV12SQLVM	0	W1	7496	Started	X3ERPV12_SEED	WSUSR	en-GB	AutoStart		
X3ERPV12SQLVM	0	W2	2144	Started	X3ERPV12_SEED	WSUSR	en-GB	AutoStart		

Task Manager

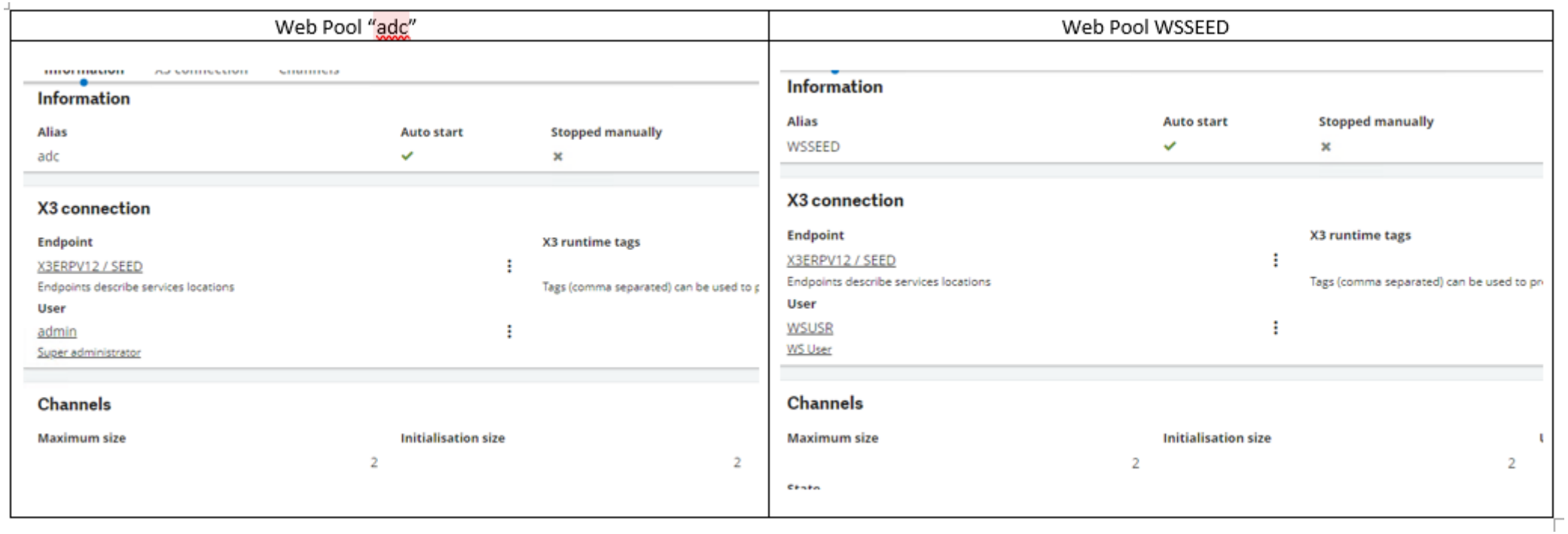
File Options View

Processes Performance Users Details Services

Name	PID	Status	Username	CPU	Memory (a...	Command line
node.exe	2144	Running	x3run	00	133,712 K	D:\Sage\SafeX3\SyraSrv\syracuse\bin\win32_x64\node.exe . W2 30 ""
node.exe	7496	Running	x3run	00	133,156 K	D:\Sage\SafeX3\SyraSrv\syracuse\bin\win32_x64\node.exe . W1 30 ""
node.exe	7140	Running	x3run	00	174,800 K	D:\Sage\SafeX3\SyraSrv\syracuse\bin\win32_x64\node.exe . N1 30 ""
node.exe	7128	Running	x3run	00	140,844 K	D:\Sage\SafeX3\SyraSrv\syracuse\bin\win32_x64\node.exe . W0 30 ""

The adonix/sadoss process-pairs are associated with Web Service Pools configured in the Administration > Administration > Web Services > Classic SOAP pools configuration option.

The number of pairs running at startup of a Pool, and the maximum number of pairs in a Pool are configurable – for example, two Web Pools called “adc” and “WSSEED”:



The results of setting up two Syracuse Child processes and two Web Pools, one called “adc” running under admin and the other called “WSSEED” running under wsuser:

Web Pool “adc”

Host	Port	Process	Pid	St
X3ERPV12SQLVM	0	W0	7128	St

X3 Session	X3 Pid	Status	X3 host	X3 port	X3 user login	X3 lo
6816	684	Available	x3erpv12sqlvm	50012	admin	en-U
6818	11844	Available	x3erpv12sqlvm	50012	admin	en-U

X3ERPV12SQLVM	0	W1	7496	St

X3 Session	X3 Pid	Status	X3 host	X3 port	X3 user login	X3 lo
6822	11896	Available	x3erpv12sqlvm	50012	admin	en-U
6827	5708	Available	x3erpv12sqlvm	50012	admin	en-U

X3ERPV12SQLVM	0	W2	2144	St

X3 Session	X3 Pid	Status	X3 host	X3 port	X3 user login	X3 lo
6826	11832	Available	x3erpv12sqlvm	50012	admin	en-U
6829	2208	Available	x3erpv12sqlvm	50012	admin	en-U

Web Pool “WSSEED”

X3ERPV12SQLVM	0	W0	7128	S

X3 Session	X3 Pid	Status	X3 host	X3 port	X3 user login	X3
6815	6592	Available	x3erpv12sqlvm	50012	wsusr	en-
6817	11792	Available	x3erpv12sqlvm	50012	wsusr	en-

X3ERPV12SQLVM	0	W1	7496	S

X3 Session	X3 Pid	Status	X3 host	X3 port	X3 user login	X3
6821	11244	Available	x3erpv12sqlvm	50012	wsusr	en-
6823	10528	Available	x3erpv12sqlvm	50012	wsusr	en-

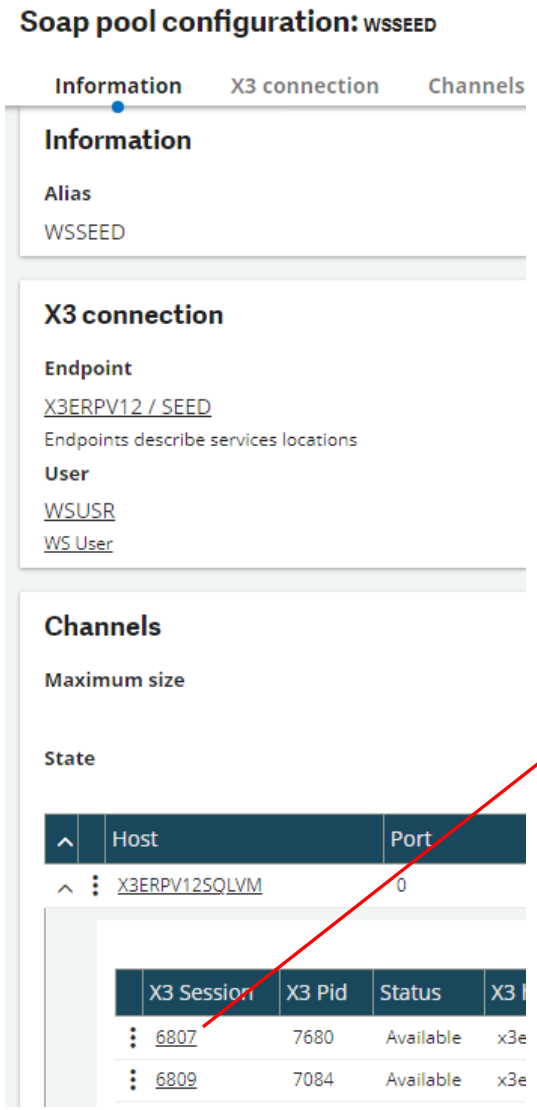
X3ERPV12SQLVM	0	W2	2144	S

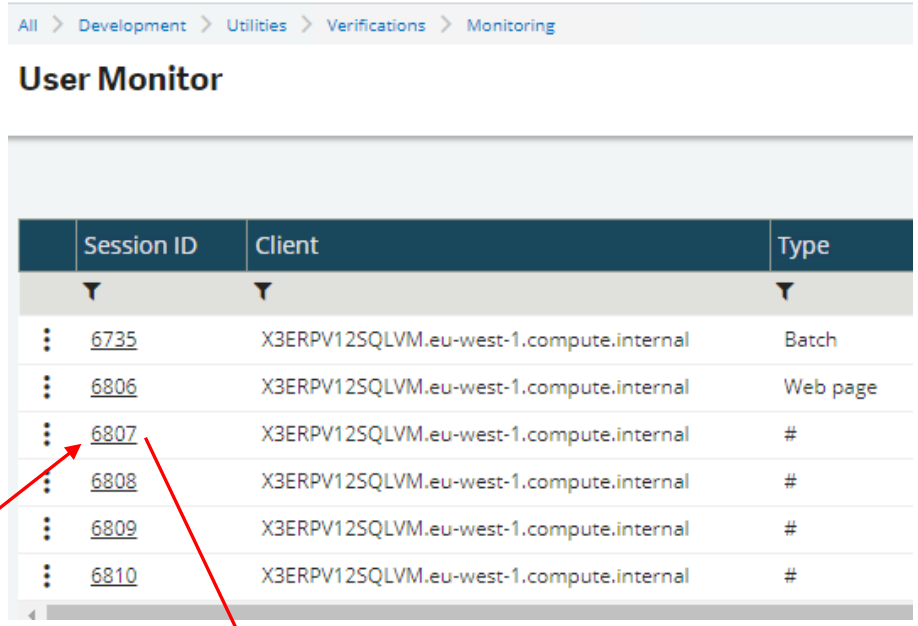
X3 Session	X3 Pid	Status	X3 host	X3 port	X3 user login	X3
6828	10404	Available	x3erpv12sqlvm	50012	wsusr	en-
6830	3892	Available	x3erpv12sqlvm	50012	wsusr	en-

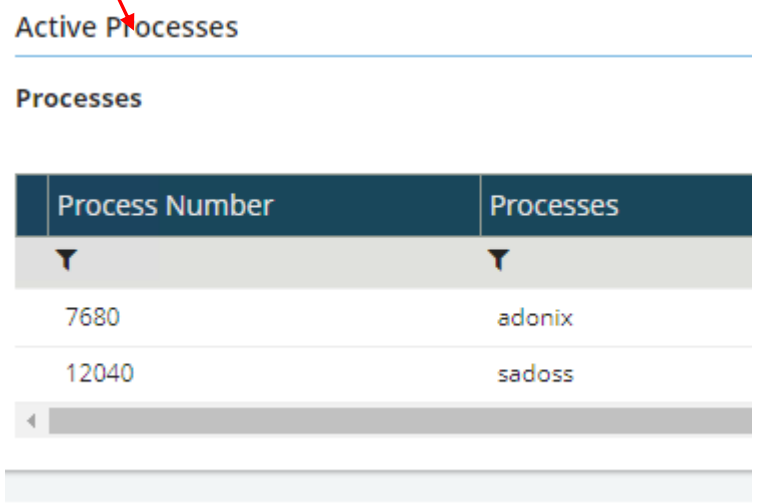
Classic SOAP Pool Configuration

The channel sessions can be seen in Administration > Administration > Web Services > Classic SOAP pools configuration and the adonix/sadoss pair can be seen in Development > Utilities > Verifications > Monitoring > User Monitor listed under the Session ID which is displayed in Classic SOAP pools configuration.

Classic SOAP Pool Configuration



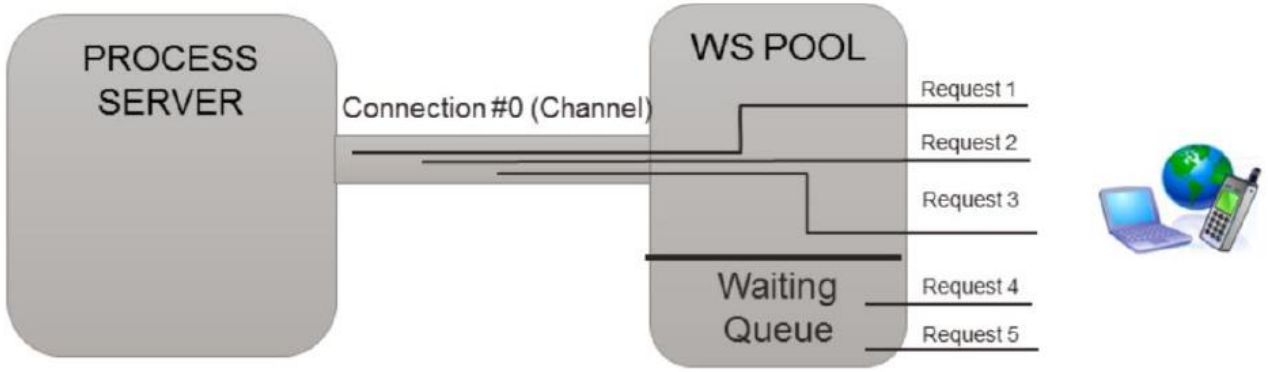




Classic SOAP Pool Configuration

This option allows you to configure a Web Service Pool with a set of Channels for requests to X3.

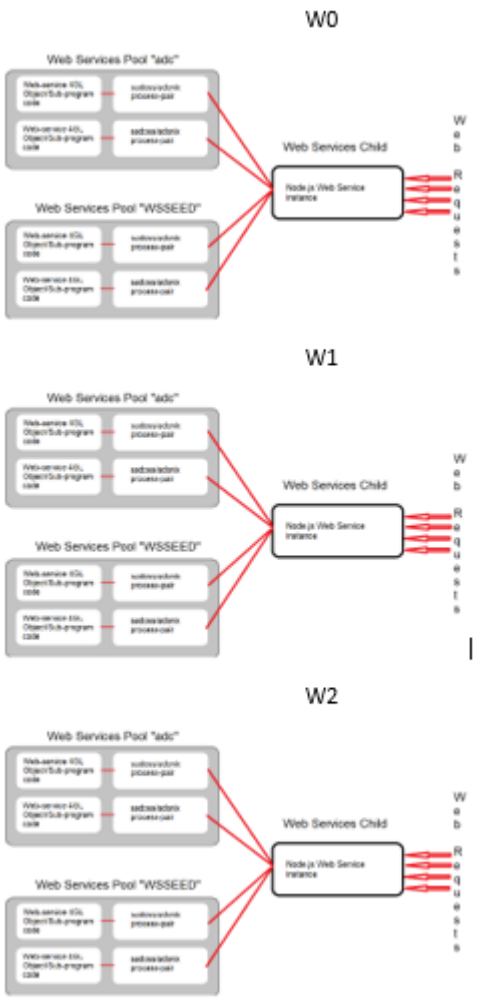
Multiple Pools can be set up for each Folder as required, and these can be used for different purposes – for example Custom Interfaces, ADC etc.

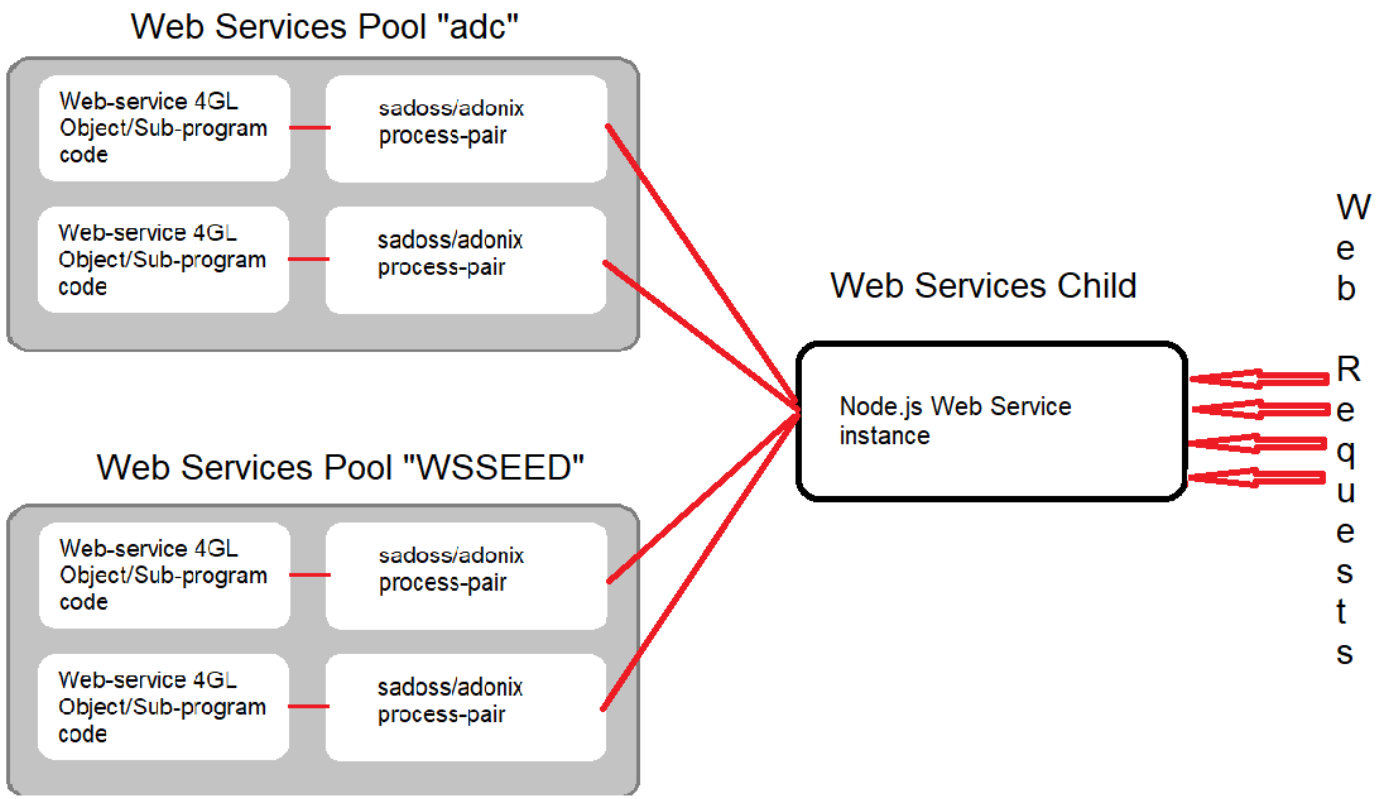


- Classic SOAP Pool Configuration https://online-help.sageerpx3.com/erp/12/staticpost/classic-soap-pools-configuration/?highlight=Classic+SOAP+
- Classic SOAP Web Services https://online-help.sageerpx3.com/erp/12/staticpost/classic-soap-pools-configuration/?highlight=Classic+SOAP+

Classic SOAP Pool Configuration

Each Web Pool, there will be X Channels in each of the Web Service Child Processes





Chapter 2.2
Types of Classic Web Service

Types of Classic Web Service

The actual Web Services are defined in Development > Script Dictionary > Scripts > Web Services (GESAWE) function.

Classic SOAP Web Services can be based on one of two types of entity:

1) Object

   This type of Web Service interacts with X3 via the Program/Script associated with the Object.

   The Web Service is basically “driving” the population of Screens associated with the underlying Program/Script which encapsulates the Business Logic relating to the Object.

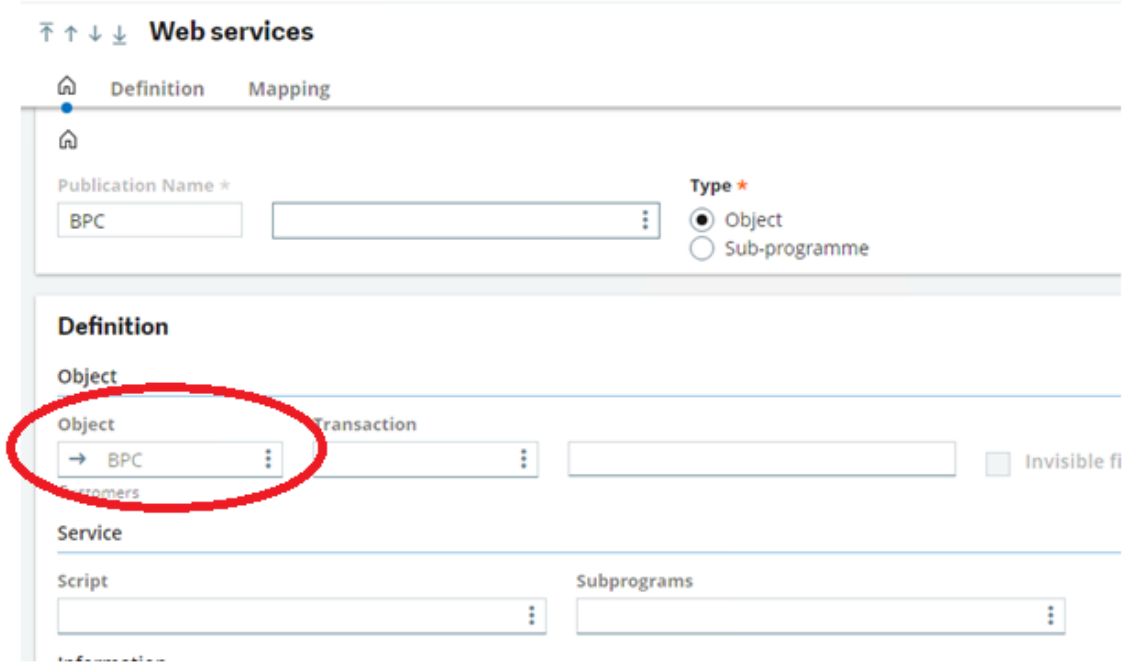
2) Sub-program

   This class of Web Service is interacting with a Sub-program which is not normally accessible to the User, but it does have parameters which need to be passed to the Sub-program.

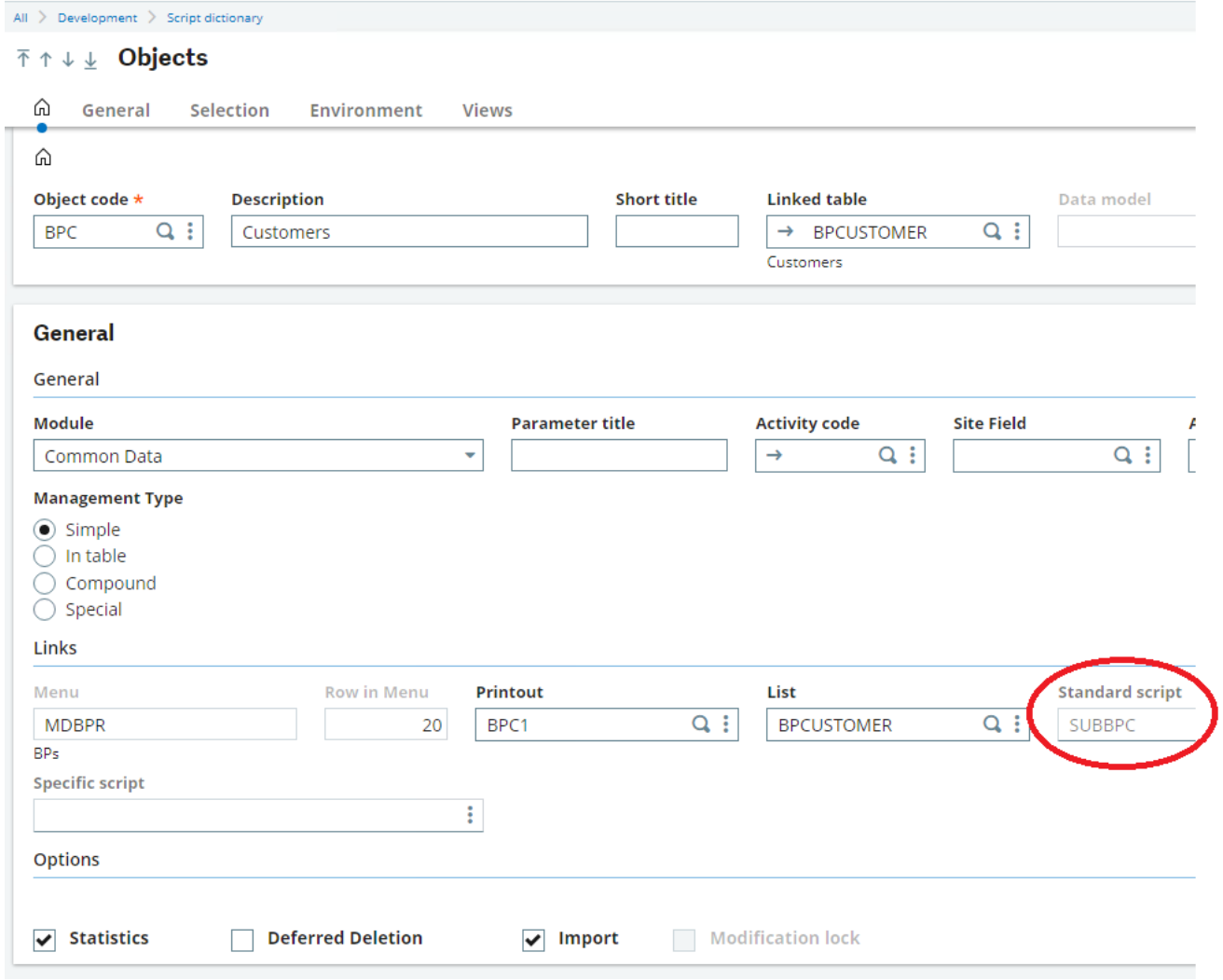
Simple Object Classic Web Services

a) Simple

A simple Object Web Service where data is referenced by multiple functions - for example, the standard Web Service BPC makes calls to SUBBPC to interact with the BPCUSTOMER entity as described in the BPC Object defined in Development > Script Dictionary > Objects (GESAOB)



Simple Object Classic Web Services

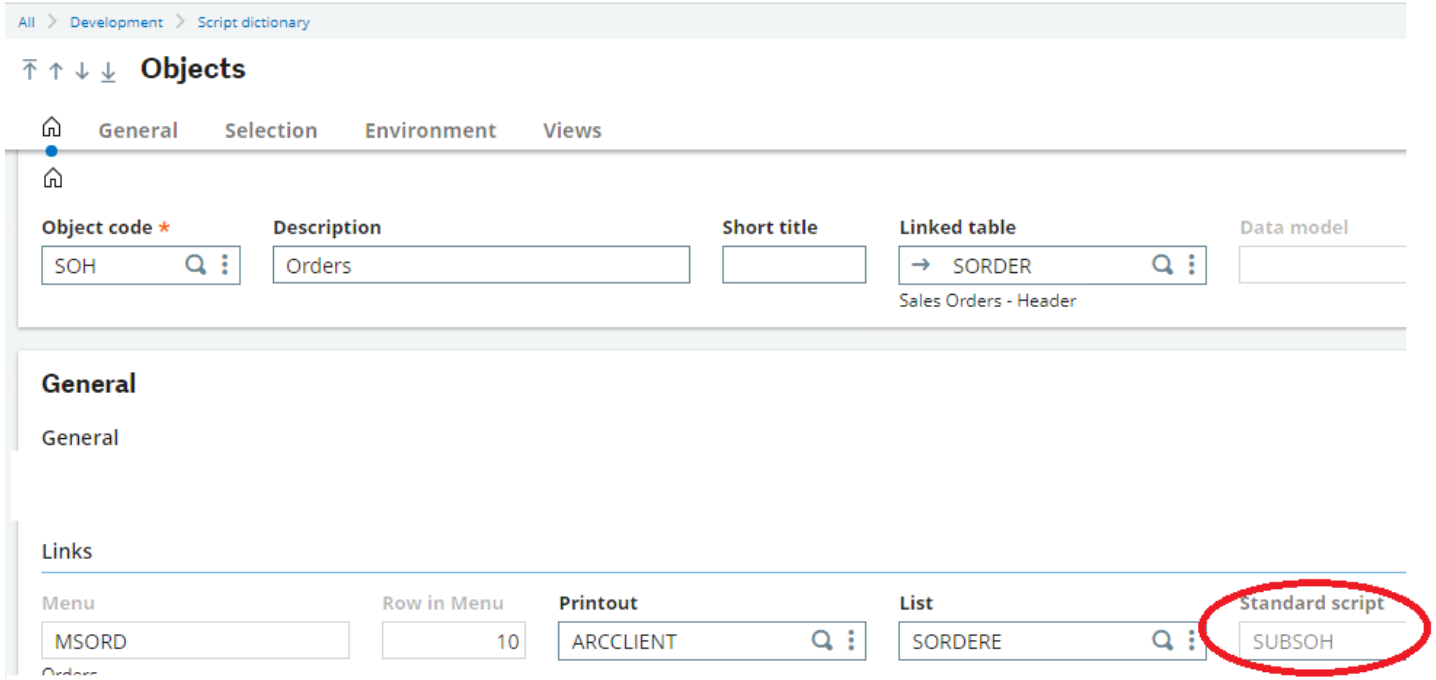


Transactional Object Classic Web Services

b) Transactional

A more complex Object Web Service class relates to data making up Transactions – for example, one which creates Sales Orders (SOH Object).

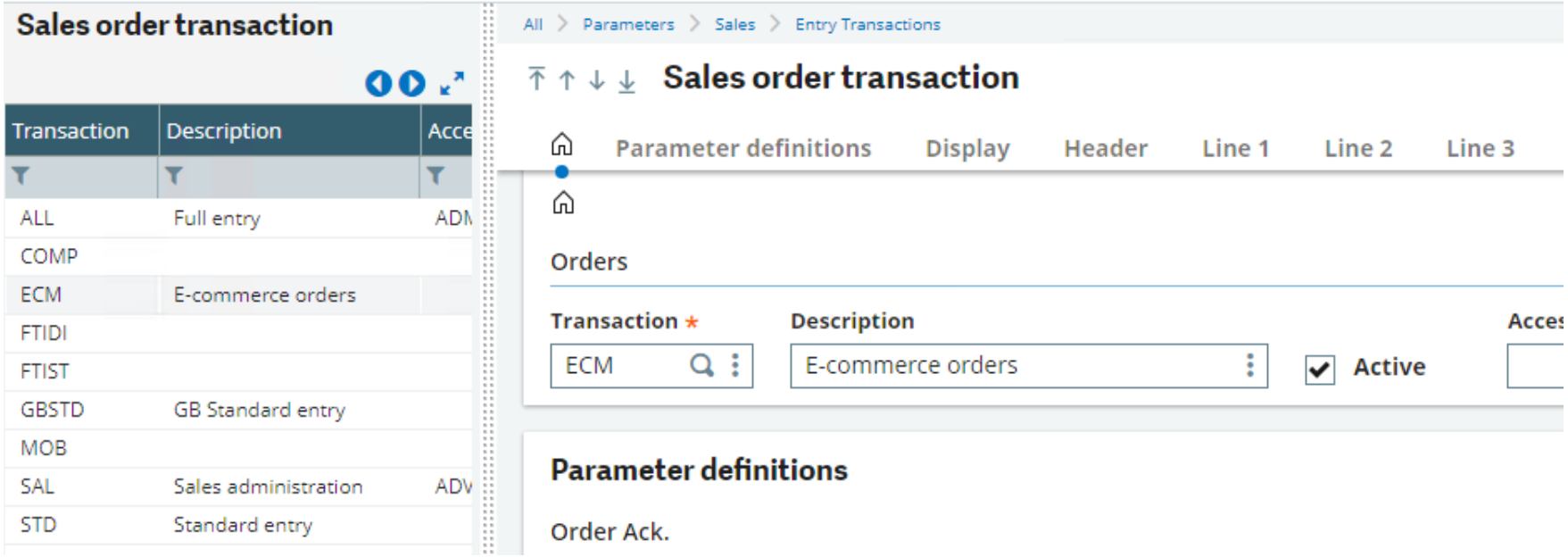
Again, the Web Service will interact with X3 via the Object's Program/Script – in this case, SUBSOH.



Transactional Object Classic Web Services

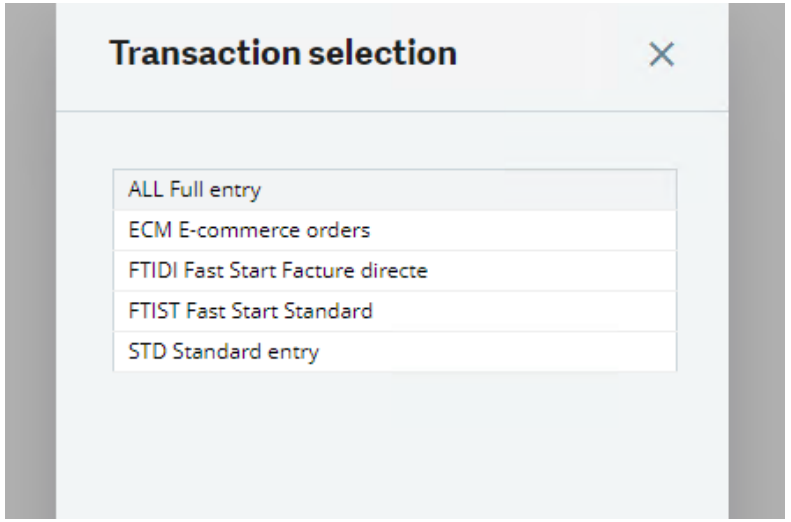
Some interesting points to highlight:

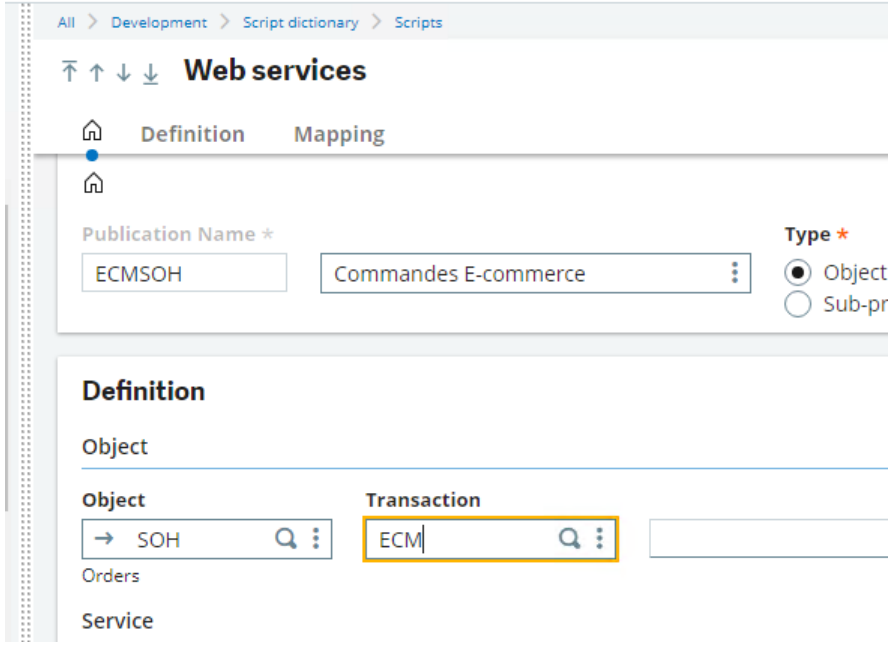
For Transactional Web Services, the Object can be viewed through the lens of different Entry Transactions – for example, in the above case ECMSOH will act as though it were interacting with X3 via the ECM Sales Order Transaction Type configured in Parameters > Sales > Entry Transactions > Orders (GESSLC)



Transactional Object Classic Web Services

When using the X3 Client, the User can choose which Entry Transaction to use when entering a particular sort of Transaction. In Web Services, it is specified when creating the Web Service.



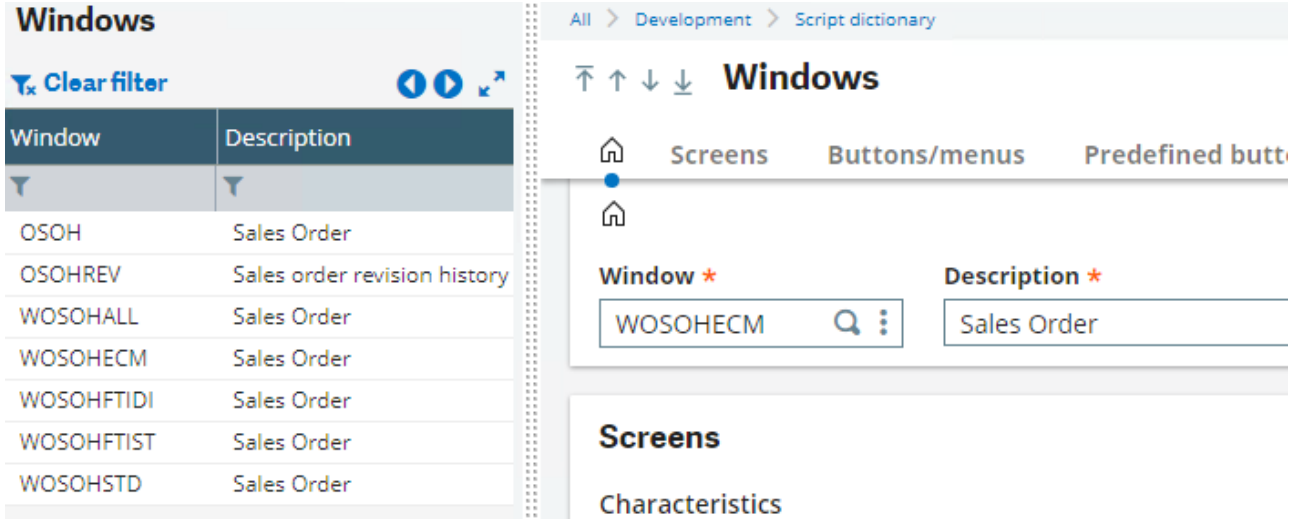


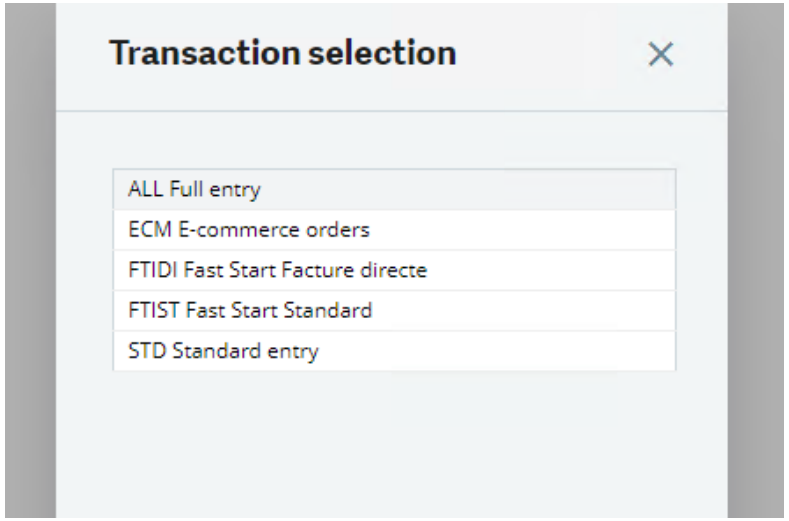
Tip: Centre of Excellence advises Developers to use Custom Entry Transactions to ensure that your Web Service is not impacted by standard Patches.

Transactional Object Classic Web Services

This mechanism means fields which aren't relevant for a particular type of Transaction can be removed/hidden so the Users don't have to see them.

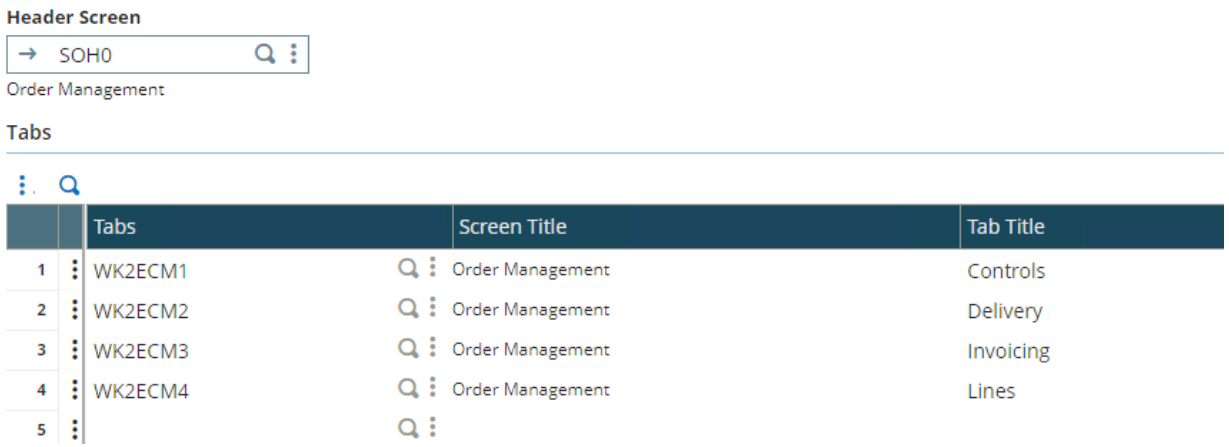
When an Entry Transaction is created/updated, it generates a set of Screens based on the Object's Entry Transaction Window – in this example, OSOH is the Template Window for the SOH Object and the Entry Transactions are generated from that Object Window – WOSOHECM in this case – visible in Development > Script dictionary > Windows (GESAWI).





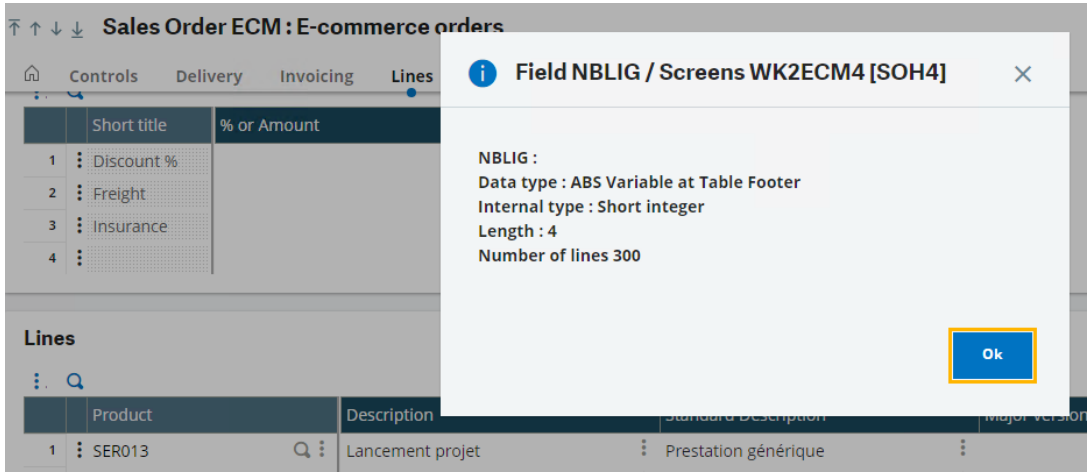
Transactional Object Classic Web Services

And this Window has a set of Screens specific to the Entry Transaction:



	Tabs	Screen Title	Tab Title
1	WK2ECM1	Order Management	Controls
2	WK2ECM2	Order Management	Delivery
3	WK2ECM3	Order Management	Invoicing
4	WK2ECM4	Order Management	Lines
5			

This can be confirmed by doing ESC-F6:



Transactional Object Classic Web Services

As the Web Service is “driving” the data-entry process via the Screens, it has to have a list of an Object’s fields which are accessible – this is the purpose of the Mapping tab in Development > Script Dictionary > Scripts > Web Services (GESAWE) :

Mapping

171 Records Page size: 100 1 2

	Selection	Group	L...	Code	Description	Internal name
1	☑	SOH0_1	1	SALFCY	Sales Site	SOH0_1~SALFCY
2	☑	SOH0_1	1	ZSALFCY		SOH0_1~ZSALFCY
3	☑	SOH0_1	1	SOHTYP	Type	SOH0_1~SOHTYP
4	☑	SOH0_1	1	ZSOHTYP		SOH0_1~ZSOHTYP
5	☑	SOH0_1	1	SOHNUM	Number	SOH0_1~SOHNUM
6	☑	SOH0_1	1	REVNUM	Revision	SOH0_1~REVNUM
7	☑	SOH0_1	1	CUSORDREF	Reference	SOH0_1~CUSORDREF

Input/Output Group

Field Name

Screen/Block/Field

Transactional Object Classic Web Services

In this example, Group SOH0_1 is the first Screen (header Screen SOH0) and Block 1 – all the fields are mapped from Code to Internal name – if “Invisible fields” is ticked, then all the fields are shown, even if they are set as “Hidden” Entry-mode. Otherwise, only those have “Display” or “Entry” Entry-mode are listed.

Web Services Mapping tab

1	SALFCY	Sales Site	SOH0_1~SALFCY
1	ZSALFCY		SOH0_1~ZSALFCY
1	SOHTYP	Type	SOH0_1~SOHTYP
1	ZSOHTYP		SOH0_1~ZSOHTYP
1	SOHNUM	Number	SOH0_1~SOHNUM
1	REVNUM	Revision	SOH0_1~REVNUM
1	CUSORDREF	Reference	SOH0_1~CUSORDREF
1	SOHNUMEND	Final number	SOH0_1~SOHNUMEND
1	ORDDAT	Date	SOH0_1~ORDDAT
1	BPCORD	Sold-to	SOH0_1~BPCORD
1	BPCNAM		SOH0_1~BPCNAM
1	CUR	Currency	SOH0_1~CUR
1	ZCUR		SOH0_1~ZCUR
1	WSOHCAT		SOH0_1~WSOHCAT
1	BPCINV	Bill-to	SOH1_1~BPCINV
1	BPINAM		SOH1_1~BPINAM

SOH0 Screen

	Field	B...	Posit...	Description	C...	C...	Type	Menu	L...	Entry
1	SALFCY	1	1	Sales Site	1	1	FCY			Enter
2	SOHTYP	1	1.1	Type	1	1	TSO			Enter
3	SOHNUM	1	1.2	Number	1	1	VCR			Enter
4	REVNUM	1	1.3	Revision	1	1	C		4	Display
5	CUSORDREF	1	2	Reference	2	1	A		20	Enter
6	SOHNUMEND	1	2.1	Final number	1	1	VCR			Display
7	ORDDAT	1	2.2	Date	2	1	D			Enter
8	BPCORD	1	3	Sold-to	4	1	BPC			Enter
9	BPCNAM	1	3.1			1	NAM			Display
10	CUR	1	3.2	Currency	1	1	CUR			Display
11	WSOHCAT	1	3.3			1	A		30	Display
12	LAN	1	4	Language	1	1	LAN			Hidden
13	CHGTYP	1	5	Rate type	1	1	M	202	15	Hidden
14	TSCCOD	1	7	Statistical Group	1	1	ADI		5	Hidden
15	OCNFLG	1	8	Print Acknowledgment	1	1	M	1	4	Hidden

Transactional Object Classic Web Services

Similarly, those fields in group SOH1_5 refer to the first child Screen (WK2ECM1 which equates to SOH1) Block 5:

Web Services Mapping tab

21	SOH1_5	1	HLDSTA
22	SOH1_5	1	HLDBTN
23	SOH1_5	1	HLDCOD

WK2ECM1 Screen

HLDSTA	5	7	Hold	Display
HLDBTN	5	7.1		Enter
HLDCOD	5	7.2		Display

Then, SOH2_2 is the second child Screen, WK2ECM2, Block 2.

29	SOH2_2	1	SHIDAT
30	SOH2_2	1	DEMDLVHOU
31	SOH2_3	1	MDL

4	DAYLT	2	2	Delivery LT	Hidden
5	SHIDAT	2	3	Shipment	Enter
6	DEMDLVHOU	2	4	Exp. delivery time	Enter
7	DRN	3	1	Route Number	Hidden
8	MDL	3	2	Shipping Mode	Enter

Note that DRN field is not listed in the Mapping as it is “Hidden”.

Transactional Object Classic Web Services

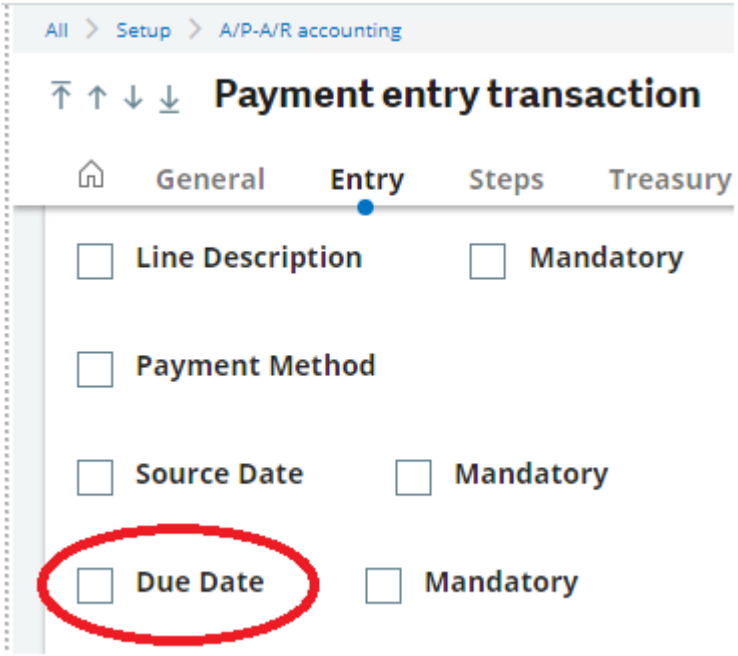
Tip: If the Object includes a header-detail data model with multiple detail-lines, then make sure the appropriate line-number column is included in the list of fields. For example, when defining the Web Service for SOH Object, include SOH_4->NUMLIG for the Line-number in SORDERP table.

Although only BPC, ITM and SOH Objects have been validated, other objects may work without much customisation – typically, some fields may need to be changed from “Hidden” to “Entered” in Development > Script Dictionary > Screens > Screens (GESAMK) in order to fulfil the requirements of Business Logic for the Object – if you continue to have problems, it could entail engaging Centre Of Excellence to give advice.

Transactional Object Classic Web Services

Note that some fields are controlled by the Transaction Entry – for example, the Due Date is not available by default in the A/P-A/R accounting > Payments > Payment / Receipt Entry (GESPAY)

It has to be enabled in Parameters > A/P-A/R accounting > Payment Entry Transactions (GESPTY)



Transactional Object Classic Web Services

I had a Support Case where the Business Partner needed to enable the DUDDAT Field in order to create a Payment, and we were unable to change the Entry-type from Hidden to Enter initially – the drop-down wasn't enabled:

126	BPSTYP	5	10		Hidden		1	A				5	
127	DUDDAT	5	10	Due Date	Hidden	1	1	D					Not transmitted
128	EPMTYP	5	11	Discount Type	Hidden	1	1	M		655		15	

The field's Entry-type can only be changed after enabling that field in the Entry Type and re-validating that:

25	CURAMTBAN	3	3.2		Display		1	CUR					
26	DUDDAT	3	4	Due Date	Enter	1	1	D					
27	CRDTYP	3	5	Card Type	Enter	1	1	ADI					

Note, the fields seem to get re-positioned after enabling the Due Date field – i.e. DUDDAT moves from Block 5 to Block 3 – this needs to be taken into account when reviewing the payload for the Web Service call.

Sub-program Classic Web Services

b) Sub-program

This class of Web Service is interacting with a Function which is not normally accessible to the User, but it does have parameters which need to be populated during the invocation – they are literally program parameters.

This sort of Web Service is not linked to any Object as such – the code is written specifically for use in the context of a Web Service.

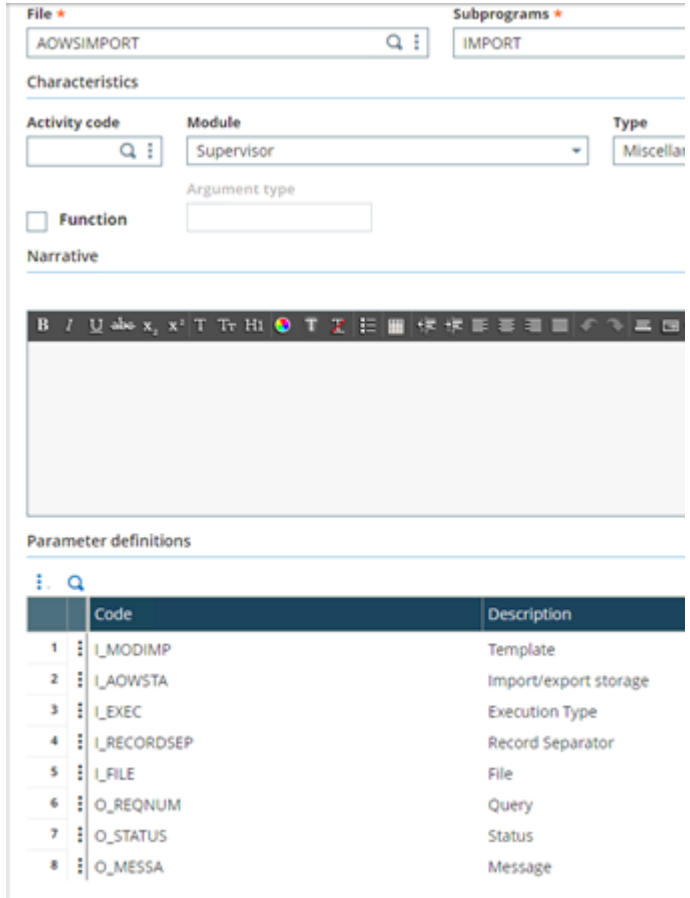
The most-used of these would be AOWSIMPORT which drives the Import function, Usage > Imports / Exports > Import (GIMPOBJ), in order to import data into an Object.

Again, Business Logic is built into the Function associated with the Import Template for an Object.

This class of Web Service is invoked using the run Operator rather than save, query or read Operators.

Sub-program Classic Web Services

For example, a call to run AOWSIMPORT requires the following Parameters to be passed to the underlying Sub-program, IMPORT:




```
#* Web service import
#*
#* @param I_MODIMP        : X3 template import
#* @param I_AOWSTA        : Temporary storage space = YES or NO
#* @param I_EXEC          :  Type of execution = BATCH or REALTIME
#* @param I_RECORDSEP : Separator of records for the import file
#* @param I_FILE          : It is the input file
#* @param O_REQNUM        : If the type is BATCH the number of batch request
#* @param O_STATUS        : Status of Web service : 0=OK else >0 ERROR
#* @param O_MESSA         : Error if status>0
#*!

Subprog IMPORT(I_MODIMP, I_AOWSTA,
I_EXEC,I_RECORDSEP,I_FILE,O_REQNUM,O_STATUS,O_MESSA)
```

Sub-program Classic Web Services

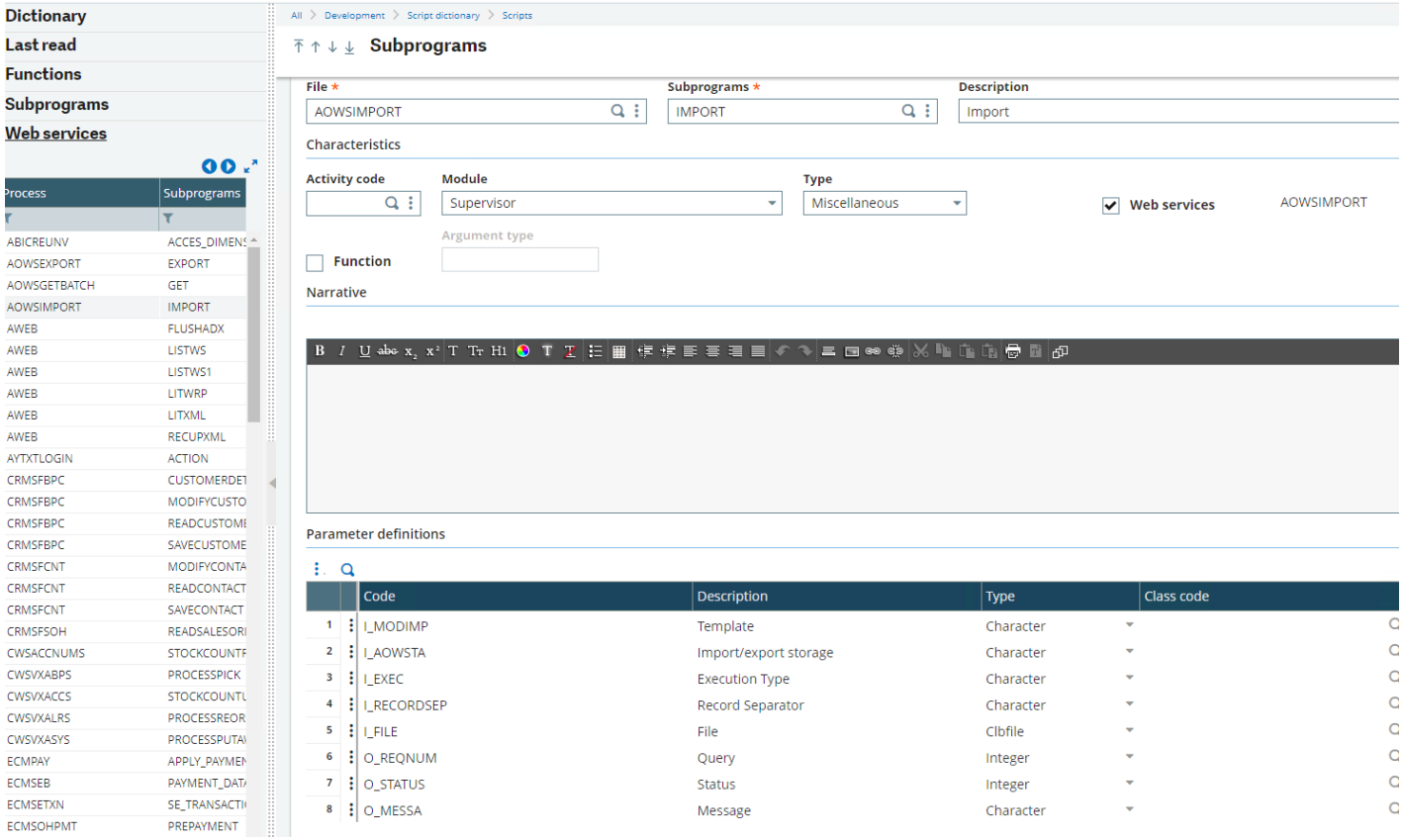
So, when the AOWSIMPORT Web Service is invoked, it will take the information in the payload and map it to the parameters expected by the underlying Subprogram (IMPORT) within the Script (AOWSIMPORT) and drive the Script's execution based on that data.

AOWSIMPORT actually drives Object-specific Scripts to carry-out the Import – for example: IMPSOH and PAYIMPORT.

Please see the Online Help URLs in the Appendix for more details.

Sub-program Classic Web Services

A full list of Sub-program SOAP Web Services can be seen in Development > Script Dictionary > Scripts > Sub-programs – you can also see the parameters expected by the Sub-program when calling it as a Web Service.



Sub-program Classic Web Services

The observant amongst you will notice that some new Sub-programs have crept-in in V12 – specifically, those with the CWS prefix – these are used by the “New ADC”.

Chapter 3
How to use Classic SOAP Web Services

There are three ways to invoke Web Service Calls:

Classic SOAP Web Services

During the Development of Web Services, it may be seen as the most convenient method of calling Web Services – although it is personal taste as to how often you use this option!

Personally, I find the interface is adequate for what I need to do and it displays the results in a reasonable format without needing to use extra directives (see SoapUI/Postman section below).

SoapUI/Postman

These are two third-party tools commonly used for Developing and Testing Web Service calls.

Programmatically

This mechanism is used to provide a way of calling Web Services from programs or scripts written to control Web Services and the scripts/programs may provide some Business Logic.

Common to all mechanisms is the list of operations which can be carried-out and the list of requestConfig directives:

Operations

* **run**
  This Operation allows the invoking code to run a Sub Program Web Service
* **save**
  This is used to create instances of an Object via an Object Web Service – this can be a Static or Transactional Object
* **delete**
  This is used to delete instances of an Object via an Object Web Service – this can be a Simple or Transactional Object
* **read**
  This is used to read specific instances of an Object via an Object Web Service – this can be a Simple or Transactional Object.
* **query**
  This returns the equivalent of the Left-list of a specified size for an Object via an Object Web Service – this can be a Simple or Transactional Object
* **getDescription**
  This is used to examine the Description of the schema associated with a Web Service (see Appendix B)
* **modify**
  This is used to modify an existing instance of an Object via an Object Web Service – this can be a Simple or Transactional Object

In addition to the above operators, the following are provided but they're a bit obscure!

* actionObject
  This triggers the specified Action for the Object
* actionObjectKeys
* getDataXmlSchema
* deleteLines

There is also a set of requestConfig directives:

Display directives:

Adxwss.optreturn=[JSON/XML]	How is the text in the result-section to be formatted – as JSON or XML?
Adxwss.beautify=[true/false]	Is the text of the result-section to be formatted or just “raw” JSON/XML?

Debugging directives:

Adxwss.trace.on=[on/off]	Activates the Web Service Trace option
Adxwss.trace.size=16384	Maximum size of trace-file – do not change.
Adonix.trace.on=[on/off]	Activates the X3 server-side trace options
Adonix.trace.level=[1/2/3]	Trace Level (Input, Output or Input&Output)
Adonix.trace.size=8	Maximum trace size (CLOB-size – do not change)
Adonix.debug.on=[on/off]	Activates X3 Debugger
Adonix.debug.host=localhost	Debugger client machine (where the debugger application is running)
Adonix.debug.port=1789	Debugger port

3.1
Invoking Web Services via Classic SOAP Web Service & SoapUI/Postman

Classic SOAP web services

Administration > Administration > Web Services > Classic SOAP web services (soapGenerics) is a built-in Function within X3 and, as such, it provides a mechanism to execute the above Operations from within X3.

SoapUI/Postman

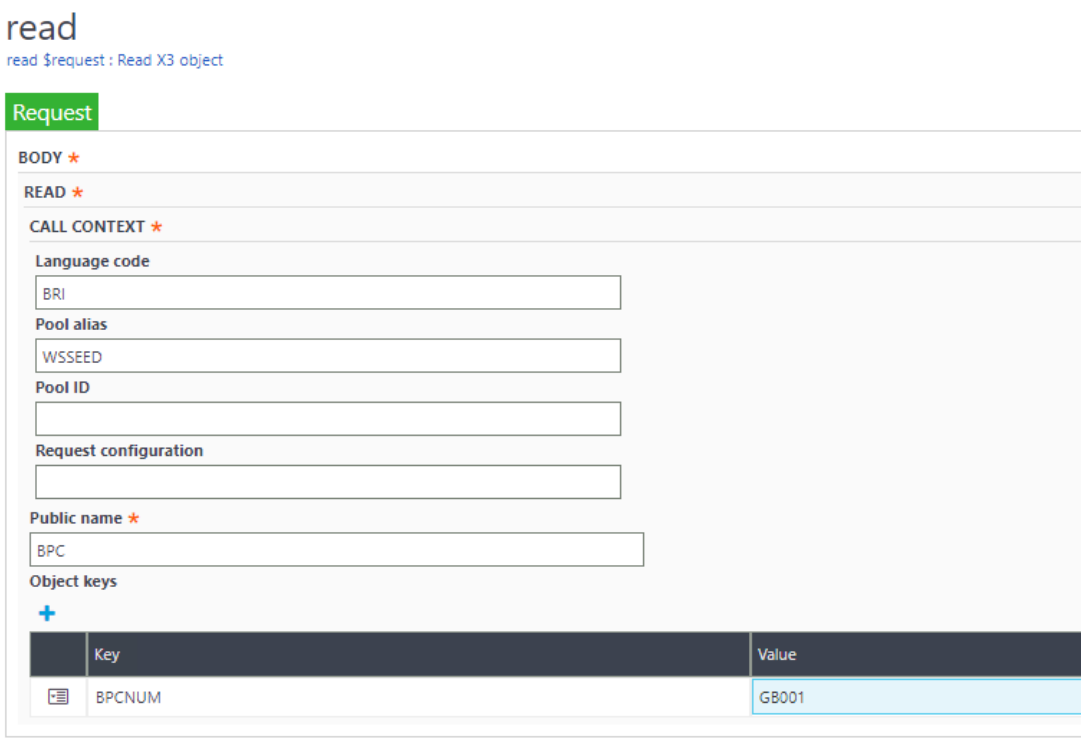
These two third-party utilities provide another way of invoking X3 Web Services – the same operations are available. They offer extra features such as:

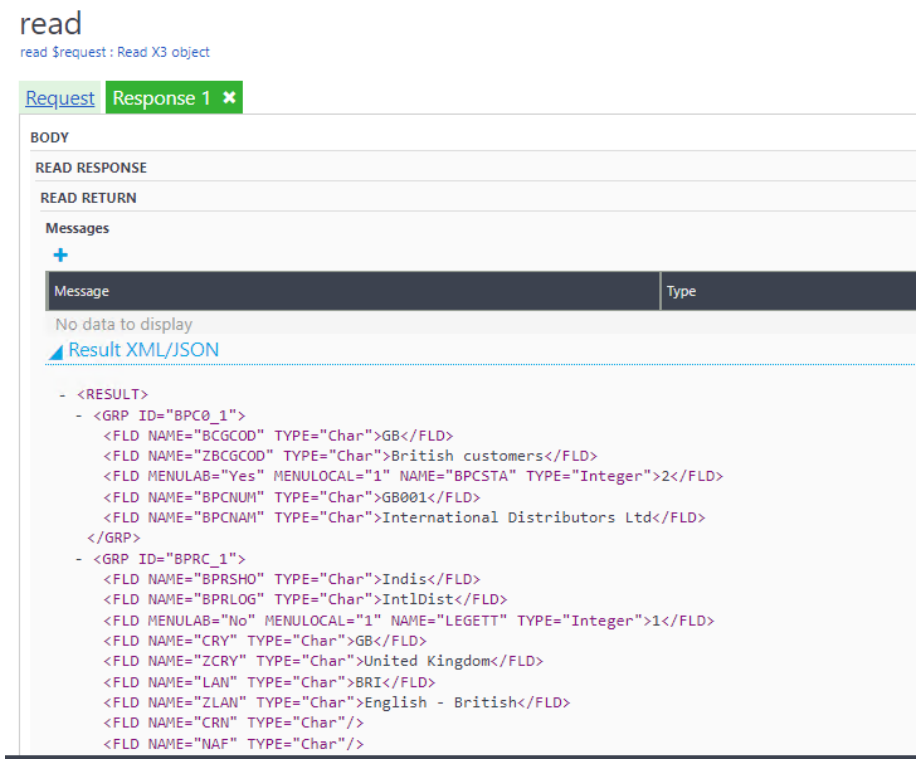
- Test Suites
- Load-testing
- The ability to invoke REST calls as well as SOAP calls in one place.

When an Operation is invoked, Classic SOAP web services prompts for the parameters which are required to carry-out the Operation – the parameter-sets will be different for each Directive. As we'll see, there's a little manual work in SoapUI/Postman but they will be the same across all three Mechanisms.

Read

Read a specific instance of an Object in Classic SOAP option:





The equivalent in SoapUI/Postman would be

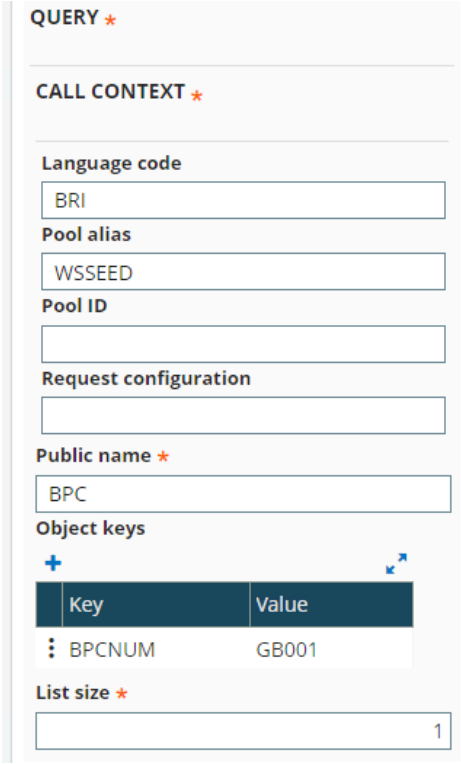
```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:wss="http://www.adonix.com/WSS"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
<soapenv:Header/>
<soapenv:Body>
<wss:read soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<callContext xsi:type="wss:CAdxCallContext">
<codeLang xsi:type="xsd:string">BRI</codeLang>
<poolAlias xsi:type="xsd:string">WSSEED</poolAlias>
<poolId xsi:type="xsd:string"></poolId>
<requestConfig xsi:type="xsd:string"> </reques tConfig>
</callContext>
<publicName xsi:type="xsd:string">BPC</publicName>
<objectKeys xsi:type="wss:ArrayOfCAdxParamKeyValue"
soapenc:arrayType="wss:CAdxParamKeyValue[]">
<key>BPCNUM</key><value>GB001</value>
</objectKeys>
</wss:read>
</soapenv:Body>
</soapenv:Envelope>
```

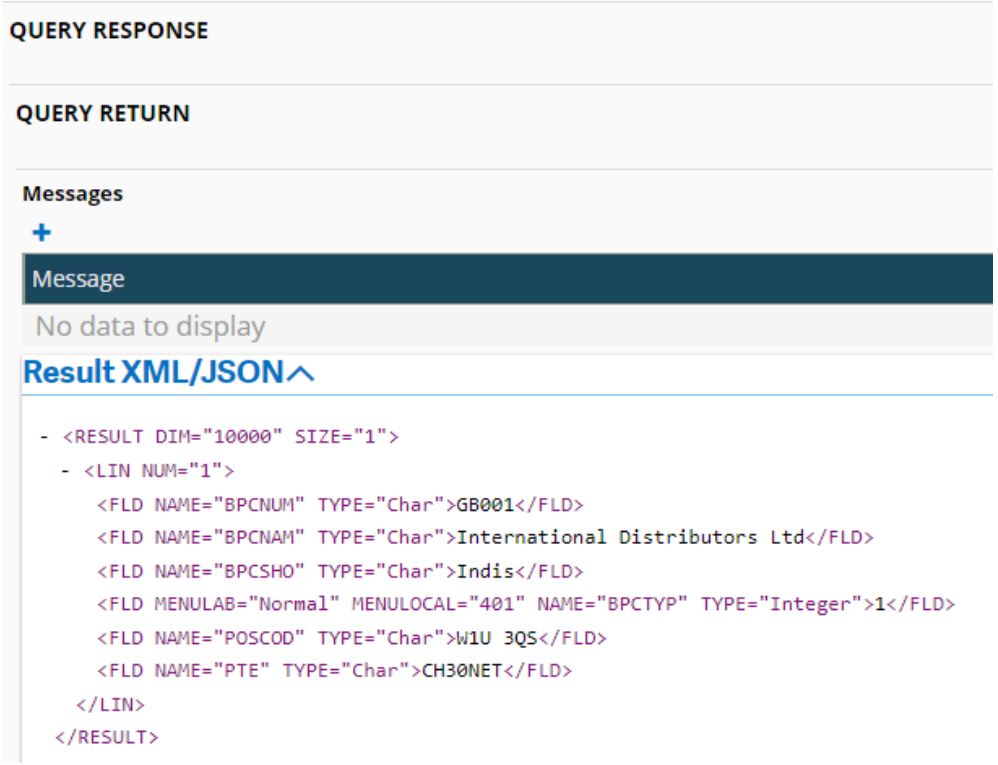
```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:wss="http://www.adonix.com/WSS">
  <soapenv:Body>
    <wss:readResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <readReturn xsi:type="wss:CAdxResultXml">
        <messages xsi:type="soapenc:Array" soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
soapenc:arrayType="wss:CAdxMessage[0]"/>
        <resultXml xsi:type="xsd:string"><![CDATA[<?xml version="1.0" encoding="UTF-8"?><RESULT><GRP
ID="BPC0_1"><FLD NAME="BCGCOD" TYPE="Char">GB</FLD><FLD NAME="ZBCGCOD" TYPE="Char">British
customers</FLD><FLD MENULAB="Yes" MENULOCAL="1" NAME="BPCSTA" TYPE="Integer">2</FLD><FLD
NAME="BPCNUM" TYPE="Char">GB001</FLD><FLD NAME="BPCNAM" TYPE="Char">International Distributors
Ltd</FLD></GRP><GRP ID="BPRC_1"><FLD NAME="BPRSHO" TYPE="Char">Indis</FLD><FLD NAME="BPRLOG"
TYPE="Char">IntlDist</FLD><FLD MENULAB="No" MENULOCAL="1" NAME="LEGETT" TYPE="Integer">1</FLD><FLD
MENULAB="No" MENULOCAL="1" NAME="BPRFBDMAG" TYPE="Integer">1</FLD><FLD NAME="CRY"
TYPE="Char">GB</FLD><FLD NAME="ZCRY" TYPE="Char">United Kingdom</FLD><FLD NAME="LAN"
TYPE="Char">BRI</FLD><FLD NAME="ZLAN" TYPE="Char">English - British</FLD><FLD NAME="CUR"
TYPE="Char">GBP</FLD><FLD NAME="ZCUR" TYPE="Char">British Pound</FLD><FLD NAME="CRN"
TYPE="Char"/><FLD NAME="NAF" TYPE="Char"/>
.....</RESULT>]]></resultXml>
        <status xsi:type="xsd:int">1</status>
        <technicalInfos xsi:type="wss:CAdxTechnicalInfos">
          <busy xsi:type="xsd:boolean">false</busy>
          <changeLanguage xsi:type="xsd:boolean">false</changeLanguage>
          <changeUserId xsi:type="xsd:boolean">false</changeUserId>
          <flushAdx xsi:type="xsd:boolean">false</flushAdx>
          <loadWebsDuration xsi:type="xsd:double">14</loadWebsDuration>
          <nbDistributionCycle xsi:type="xsd:int">-1</nbDistributionCycle>
          <poolDistribDuration xsi:type="xsd:double">3</poolDistribDuration>
          <poolEntryIdx xsi:type="xsd:int">3000</poolEntryIdx>
          <poolExecDuration xsi:type="xsd:double">331</poolExecDuration>
          <poolRequestDuration xsi:type="xsd:double">-1</poolRequestDuration>
          <poolWaitDuration xsi:type="xsd:double">4</poolWaitDuration>
          <processReport xsi:type="xsd:string" xsi:nil="true"/>
          <processReportSize xsi:type="xsd:int">-1</processReportSize>
          <reloadWebs xsi:type="xsd:boolean">false</reloadWebs>
          <resumitAfterDBOpen xsi:type="xsd:boolean">false</resumitAfterDBOpen>
          <rowInDistribStack xsi:type="xsd:int" xsi:nil="true"/>
          <totalDuration xsi:type="xsd:double">390</totalDuration>
          <traceRequest xsi:type="xsd:string"/>
          <traceRequestSize xsi:type="xsd:int">0</traceRequestSize>
        </technicalInfos>
      </readReturn>
    </wss:readResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Query

Produce a list of Records of a specified size.

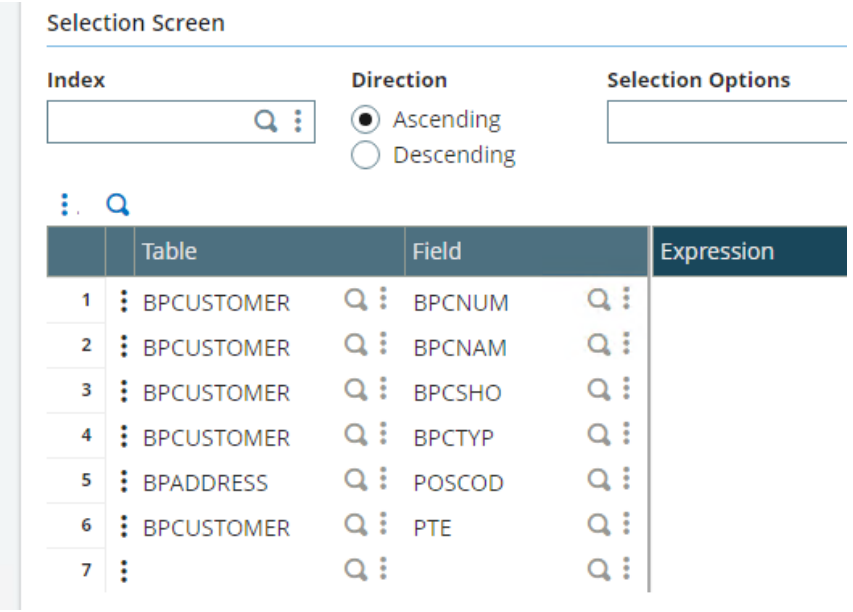
The data returned can be refined by including optional Key/Value pairs in addition to the number of instances to be returned For example:

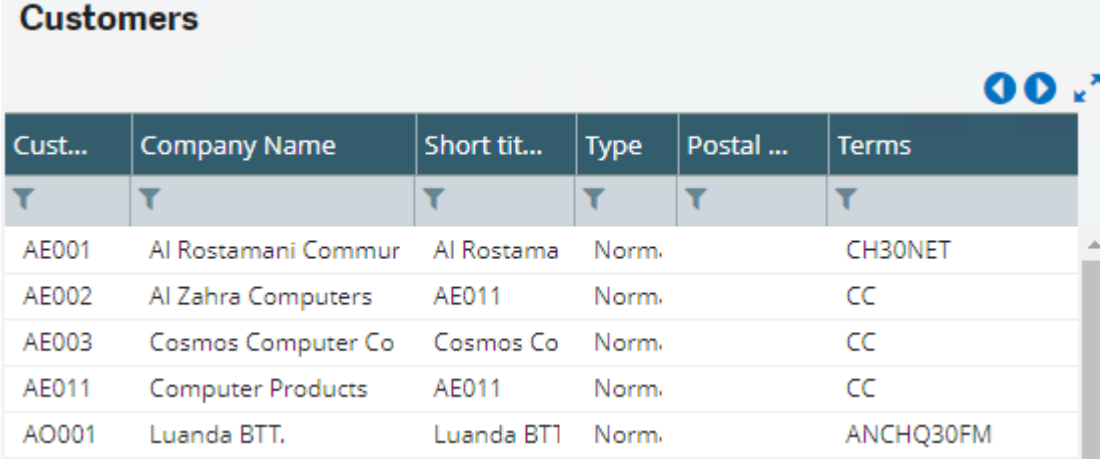




Note that the Keys/Fields used to refine a Query don't actually have to be part of the list of fields returned.

Note that the list of fields returned by Query equate to the left-list within the Object's function – for example, for the Selection-fields for BPC in Development > Script dictionary > Objects (GESAOB), and Common Data > BPs > Customers (GESBPC) :





This is in contrast with the fields returned by Read which are governed by the Web Service definition.

The equivalent in SoapUI/Postman would be

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wss="http://www.adonix.com/WSS" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
  <soapenv:Header/>
  <soapenv:Body>
    <wss:query soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <callContext xsi:type="wss:CAdxCallContext">
        <codeLang xsi:type="xsd:string">BRI</codeLang>
        <poolAlias xsi:type="xsd:string">WSSEED</poolAlias>
        <poolId xsi:type="xsd:string"></poolId>
        <requestConfig xsi:type="xsd:string">BPC</requestConfig>
      </callContext>
      <publicName xsi:type="xsd:string">?</publicName>
      <objectKeys xsi:type="wss:ArrayOfCAdxParamKeyValue"
soapenc:arrayType="wss:CAdxParamKeyValue[]">
      <key>BPCNUM</key><value>GB001</value>
      </objectKeys>
      <listSize xsi:type="xsd:int">1</listSize>
    </wss:query>
  </soapenv:Body>
</soapenv:Envelope>
```

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wss="http://www.adonix.com/WSS">
  <soapenv:Body>
    <wss:queryResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <queryReturn xsi:type="wss:CAdxResultXml">
        <messages xsi:type="soapenc:Array"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
soapenc:arrayType="wss:CAdxMessage[0]"/>
        <resultXml xsi:type="xsd:string"><![CDATA[<?xml version="1.0" encoding="UTF-8"?
><RESULT DIM="10000" SIZE="1"><LIN NUM="1"><FLD NAME="BPCNUM"
TYPE="Char">GB002</FLD><FLD NAME="BPCNAM" TYPE="Char">Amalgamate Supply
plc</FLD><FLD NAME="BPCSHO" TYPE="Char">Amalgamate</FLD><FLD
MENULAB="Normal" MENULOCAL="401" NAME="BPCTYP"
TYPE="Integer">1</FLD><FLD NAME="POSCOD" TYPE="Char">B16 8LA</FLD><FLD
NAME="PTE" TYPE="Char">CH30NET</FLD></LIN></RESULT>]]></resultXml>
        <status xsi:type="xsd:int">1</status>
        <technicalInfos xsi:type="wss:CAdxTechnicalInfos">
          <busy xsi:type="xsd:boolean">false</busy>
          <changeLanguage xsi:type="xsd:boolean">false</changeLanguage>
          <changeUserId xsi:type="xsd:boolean">false</changeUserId>
          <flushAdx xsi:type="xsd:boolean">false</flushAdx>
          <loadWebsDuration xsi:type="xsd:double">14</loadWebsDuration>
          <nbDistributionCycle xsi:type="xsd:int">-1</nbDistributionCycle>
          <poolDistribDuration xsi:type="xsd:double">1</poolDistribDuration>
          <poolEntryIdx xsi:type="xsd:int">3768</poolEntryIdx>
          <poolExecDuration xsi:type="xsd:double">113</poolExecDuration>
          <poolRequestDuration xsi:type="xsd:double">-1</poolRequestDuration>
          <poolWaitDuration xsi:type="xsd:double">0</poolWaitDuration>
          <processReport xsi:type="xsd:string" xsi:nil="true"/>
          <processReportSize xsi:type="xsd:int">-1</processReportSize>
          <reloadWebs xsi:type="xsd:boolean">false</reloadWebs>
          <resumitAfterDBOpen xsi:type="xsd:boolean">false</resumitAfterDBOpen>
          <rowInDistribStack xsi:type="xsd:int" xsi:nil="true"/>
          <totalDuration xsi:type="xsd:double">153</totalDuration>
          <traceRequest xsi:type="xsd:string"/>
          <traceRequestSize xsi:type="xsd:int">0</traceRequestSize>
        </technicalInfos>
      </queryReturn>
    </wss:queryResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Save

Create a new instance of an Object – this could involve creating data in multiple Tables. For example BPC (as defined in the Environments tab of the Object).

SAVE *

CALL CONTEXT *

Language code

BRI

Pool alias

WSSEED

Pool ID

Request configuration

Public name *

BPC

Object Xml *


```
<PARAM>
<GRP ID="BPC0_1">
<FLD NAME="BCGCOD" TYPE="Char">GB</FLD>
<FLD NAME="ZBCGCOD" TYPE="Char">British customers</FLD>
<FLD MENULAB="Yes" MENULOCAL="1" NAME="BPCSTA" TYPE="Integer">2</FLD>
<FLD NAME="BPCNUM" TYPE="Char">GB001a</FLD>
<FLD NAME="BPCNAM" TYPE="Char">International Distributors Ltd</FLD>
</GRP>
<GRP ID="BPRC_1">
<FLD NAME="BPRSHO" TYPE="Char">Indis</FLD>
<FLD NAME="BPRLOG" TYPE="Char">IntlDist</FLD>
<FLD MENULAB="No" MENULOCAL="1" NAME="LEGETT" TYPE="Integer">1</FLD>
<FLD MENULAB="No" MENULOCAL="1" NAME="BPRFBDMAG" TYPE="Integer">1</FLD>
```


SAVE RETURN

Messages

+

Message
No data to display

Result XML/JSON


```
- <RESULT>
  - <GRP ID="BPC0_1">
      <FLD NAME="BCGCOD" TYPE="Char">GB</FLD>
      <FLD NAME="ZBCGCOD" TYPE="Char">British customers</FLD>
      <FLD MENULAB="Yes" MENULOCAL="1" NAME="BPCSTA" TYPE="Integer">2</FLD>
      <FLD NAME="BPCNUM" TYPE="Char">GB001a</FLD>
    - <FLD NAME="BPCNAM" TYPE="Char">
      </FLD>
    </GRP>
  - <GRP ID="BPRC_1">
      <FLD NAME="BPRSHO" TYPE="Char">Indis</FLD>
      <FLD NAME="BPRLOG" TYPE="Char">IntlDist</FLD>
      <FLD MENULAB="No" MENULOCAL="1" NAME="LEGETT" TYPE="Integer">1</FLD>
      <FLD MENULAB="No" MENULOCAL="1" NAME="BPRFBDMAG" TYPE="Integer">1</FLD>
      <FLD NAME="CRY" TYPE="Char">GB</FLD>
      <FLD NAME="ZCRY" TYPE="Char">United Kingdom</FLD>
      <FLD NAME="LAN" TYPE="Char">BRI</FLD>
      <FLD NAME="ZLAN" TYPE="Char">English - British</FLD>
      <FLD NAME="CUR" TYPE="Char">GBP</FLD>
      <FLD NAME="ZCUR" TYPE="Char">British Pound</FLD>
      <FLD NAME="CRN" TYPE="Char"/>
      <FLD NAME="NAF" TYPE="Char"/>
      <FLD NAME="EECNUM" TYPE="Char">GB1000011122</FLD>
      <FLD NAME="EVCVAL" TYPE="Char">102</FLD>
```

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wss="http://www.adonix.com/WSS">
  <soapenv:Header/>
  <soapenv:Body>
    <wss:save soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <callContext xsi:type="wss:CAdxCallContext">
        <codeLang xsi:type="xsd:string">BRI</codeLang>
        <poolAlias xsi:type="xsd:string">WSSEED</poolAlias>
        <poolId xsi:type="xsd:string">?</poolId>
        <requestConfig xsi:type="xsd:string">?</requestConfig>
      </callContext>
      <publicName xsi:type="xsd:string">BPC</publicNae>
      <objectXml xsi:type="xsd:string"><![CDATA[<?xml version="1.0" encoding="UTF-8"?><?xml
version="1.0" encoding="UTF-8"?>
<PARAM>
<GRP ID="BPC0_1">
<FLD NAME="BCGCOD" TYPE="Char">GB</FLD>
<FLD NAME="ZBCGCOD" TYPE="Char">British customers</FLD>
<FLD MENULAB="Yes" MENULOCAL="1" NAME="BPCSTA" TYPE="Integer">2</FLD>
<FLD NAME="BPCNUM" TYPE="Char">GB001b</FLD>
```

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wss="http://www.adonix.com/WSS">
  <soapenv:Body>
    <wss:saveResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <saveReturn xsi:type="wss:CAdxResultXml">
        <messages xsi:type="soapenc:Array"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
soapenc:arrayType="wss:CAdxMessage[0]"/>
        <resultXml xsi:type="xsd:string"><![CDATA[<?xml version="1.0" encoding="UTF-8"?
><RESULT><GRP ID="BPC0_1"><FLD NAME="BCGCOD" TYPE="Char">GB</FLD><FLD
NAME="ZBCGCOD" TYPE="Char">British customers</FLD><FLD MENULAB="Yes"
MENULOCAL="1" NAME="BPCSTA" TYPE="Integer">2</FLD><FLD NAME="BPCNUM"
TYPE="Char">GB001b</FLD><FLD NAME="BPCNAM" TYPE="Char"></FLD></GRP><GRP
ID="BPRC_1"><FLD NAME="BPRSHO" TYPE="Char">Indis</FLD><FLD NAME="BPRLOG"
TYPE="Char">IntlDist</FLD><FLD MENULAB="No" MENULOCAL="1" NAME="LEGETT"
TYPE="Integer">1</FLD><FLD MENULAB="No" MENULOCAL="1" NAME="BPRFBDMAG"
```

The Interface for SoapUI/Postman is more basic that that of Classic SOAP Web Services, so you may wish to add extra directives, see above, to help format the results in the Request Configuration section of the operation:

For example, the output of Read with no requestConfig would look something like


```
adReturn xsi:type="wss:CAdxResultXml">
<messages xsi:type="soapenc:Array" soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" soapenc:arrayTyp
<resultXml xsi:type="xsd:string"><![CDATA[<?xml version="1.0" encoding="UTF-8"?><RESULT><GRP ID="BPC0_1"><FLD NAME="B
<status xsi:type="xsd:int">1</status>
<technicalInfos xsi:type="wss:CAdxTechnicalInfos">
    <busy xsi:type="xsd:boolean">false</busy>
    <changeLanguage xsi:type="xsd:boolean">false</changeLanguage>
    <changeUserId xsi:type="xsd:boolean">true</changeUserId>
```


Which is not very user-friendly!

You can see the return-value between <RESULT> and </RESULT> inside the <resultXml>…</resultXml> tags.

For example, a “raw” Read and resulting output might be

```
<soapenv:Header/>
<soapenv:Body>
   <wss:read soapenv:encodingStyle="http://schemas.xmlsoap.org/soa
      <callContext xsi:type="wss:CAdxCallContext">
         <codeLang xsi:type="xsd:string">BRI</codeLang>
         <poolAlias xsi:type="xsd:string">WSSEED</poolAlias>
         <poolId xsi:type="xsd:string">?</poolId>
         <requestConfig xsi:type="xsd:string"></requestConfig>
      </callContext>
      <publicName xsi:type="xsd:string">BPC</publicName>
      <objectKeys xsi:type="wss:ArrayOfCAdxParamKeyValue" soapenc
<key>BPCNUM</key><value>GB001</value>
</objectKeys>
</wss:read>
```

```
<soapenv:Body>
   <wss:readResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <readReturn xsi:type="wss:CAdxResultXml">
         <messages xsi:type="soapenc:Array" soapenv:encodingStyle="http://schemas.xmlso
         <resultXml xsi:type="xsd:string"><![CDATA[<?xml version="1.0" encoding="UTF-8"
         <status xsi:type="xsd:int">1</status>
         <technicalInfos xsi:type="wss:CAdxTechnicalInfos">
            <busy xsi:type="xsd:boolean">false</busy>
            <changeLanguage xsi:type="xsd:boolean">false</changeLanguage>
            <changeUserId xsi:type="xsd:boolean">false</changeUserId>
            <flushAdx xsi:type="xsd:boolean">false</flushAdx>
            <loadWebsDuration xsi:type="xsd:double">14</loadWebsDuration>
            <nbDistributionCycle xsi:type="xsd:int">-1</nbDistributionCycle>
            <poolDistribDuration xsi:type="xsd:double">3</poolDistribDuration>
```

Might be transformed by using &adxwss.beautify=on, the results are formatted and more readable – the default adxwss.optreturn is XML

```
<wss:read soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/
   <callContext xsi:type="wss:CAdxCallContext">
      <codeLang xsi:type="xsd:string">BRI</codeLang>
      <poolAlias xsi:type="xsd:string">WSSEED</poolAlias>
      <poolId xsi:type="xsd:string">?</poolId>
      <requestConfig xsi:type="xsd:string">&adxwss.beautify=true<
   </callContext>
   <publicName xsi:type="xsd:string">BPC</publicName>
   <objectKeys xsi:type="wss:ArrayOfCAdxParamKeyValue" soapenc:ar
y>BPCNUM</key><value>GB001</value>
bjectKeys>
ss:read>
</soapenv:Body>
```

```
<readReturn xsi:type="wss:CAdxResultXml">
   <messages xsi:type="soapenc:Array" soapenv:encodingStyle="http://schemas.xmlso
   <resultXml xsi:type="xsd:string"><![CDATA[<?xml version="1.0" encoding="UTF-8"
<RESULT>
   <GRP ID="BPC0_1">
      <FLD NAME="BCGCOD" TYPE="Char">GB</FLD>
      <FLD NAME="ZBCGCOD" TYPE="Char">British customers</FLD>
      <FLD MENULAB="Yes" MENULOCAL="1" NAME="BPCSTA" TYPE="Integer">2</FLD>
      <FLD NAME="BPCNUM" TYPE="Char">GB001</FLD>
      <FLD NAME="BPCNAM" TYPE="Char">International Distributors Ltd</FLD>
   </GRP>
   <GRP ID="BPRC_1">
      <FLD NAME="BPRSHO" TYPE="Char">Indis</FLD>
```

For example, a “raw” Read and resulting output might be

```
<soapenv:Header/>
<soapenv:Body>
   <wss:read soapenv:encodingStyle="http://schemas.xmlsoap.org/soa
      <callContext xsi:type="wss:CAdxCallContext">
         <codeLang xsi:type="xsd:string">BRI</codeLang>
         <poolAlias xsi:type="xsd:string">WSSEED</poolAlias>
         <poolId xsi:type="xsd:string">?</poolId>
         <requestConfig xsi:type="xsd:string"></requestConfig>
      </callContext>
      <publicName xsi:type="xsd:string">BPC</publicName>
      <objectKeys xsi:type="wss:ArrayOfCAdxParamKeyValue" soapenc
<key>BPCNUM</key><value>GB001</value>
</objectKeys>
</wss:read>
```

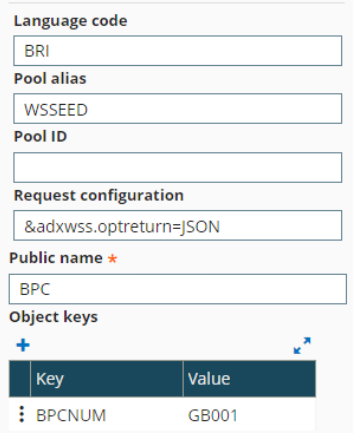
```
Raw
<soapenv:Body>
   <wss:readResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <readReturn xsi:type="wss:CAdxResultXml">
         <messages xsi:type="soapenc:Array" soapenv:encodingStyle="http://schemas.xmlsc
         <resultXml xsi:type="xsd:string"><![CDATA[<?xml version="1.0" encoding="UTF-8"
         <status xsi:type="xsd:int">1</status>
         <technicalInfos xsi:type="wss:CAdxTechnicalInfos">
            <busy xsi:type="xsd:boolean">false</busy>
            <changeLanguage xsi:type="xsd:boolean">false</changeLanguage>
            <changeUserId xsi:type="xsd:boolean">false</changeUserId>
            <flushAdx xsi:type="xsd:boolean">false</flushAdx>
            <loadWebsDuration xsi:type="xsd:double">14</loadWebsDuration>
            <nbDistributionCycle xsi:type="xsd:int">-1</nbDistributionCycle>
            <poolDistribDuration xsi:type="xsd:double">3</poolDistribDuration>
```

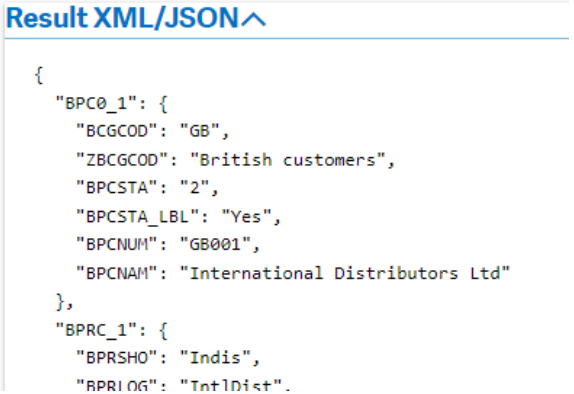
By using &adxwss.optreturn=JSON&adxwss.beautify=on, the results are formatted and more readable in JSON (if that’s your thing!):

```
<wss:read soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <callContext xsi:type="wss:CAdxCallContext">
      <codeLang xsi:type="xsd:string">BRI</codeLang>
      <poolAlias xsi:type="xsd:string">WSSEED</poolAlias>
      <poolId xsi:type="xsd:string">?</poolId>
      <requestConfig xsi:type="xsd:string">&adxwss.optreturn=JSON&adxwss.beautify=true<
   </callContext>
   <publicName xsi:type="xsd:string">BPC</publicName>
   <objectKeys xsi:type="wss:ArrayOfCAdxParamKeyValue" soapenc:arrayType="wss:CAdxParam
y>BPCNUM</key><value>GB001</value>
bjectKeys>
ss:read>
</soapenv:Body>
oapenv:Envelope>
```

```
<readReturn xsi:type="wss:CAdxResultXml">
   <messages xsi:type="soapenc:Array" soapenv:encoding
   <resultXml xsi:type="xsd:string">{
"BPC0_1": {
      "BCGCOD": "GB",
      "ZBCGCOD": "British customers",
      "BPCSTA": "2",
      "BPCSTA_LBL": "Yes",
      "BPCNUM": "GB001",
      "BPCNAM": "International Distributors Ltd"
},
"BPRC_1": {
      "BPRSHO": "Indis",
      "BPRLOG": "IntlDist",
      "BPRNAM": [
         "International Distributors Ltd",
```

As mentioned before, these directives can be used in Classic SOAP option:





Using SoapUI/Postman, you can build up test-harnesses which can be used to issue multiple Web Service Calls and even perform some load/performance testing – something you can’t do with the Classic SOAP Web Services option!

Using the debugging directives as well might give you something like

<requestConfig>&adxwss.trace.on=on&adxwss.trace.size=16384&adxwss.optreturn=JSON&adxwss.beautify=true&adonix.trace.on=on&adonix.trace.level=3&adonix.trace.size=8</requestConfig>

Sub-program Classic Web Services

As mentioned before, Sub-program Web Services are invoked using the run directive.

An example of invoking a Sub-program Web Service is doing a run with AOWSIMPORT – it can be invoked in Classic SOAP Web Services or SoapUI/Postman:

RUN *

CALL CONTEXT *

Language code

BRI

Pool alias

WSSEED

Pool ID

Request configuration

Public name *

AOWSIMPORT

Input XML/JSON *

```
<PARAM>
<FLD NAM="I_MODIMP">WSPAY1XML</FLD>
<FLD NAM="I_AOWSTA">NO</FLD>
<FLD NAM="I_EXEC">REALTIME</FLD>
<FLD NAM="I_RECORDSEP">|</FLD>
<FLD NAM="I_FILE">&lt;?xml version="1.0" encoding="ISO 8859" ?&gt;
&lt;p:WWIMPWSPAY1XML xmlns:p="http://com.sage/X3/WWIMPWSPAY1XML"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://com.sage/X3/WWIMPWSPAY1XML WWIMPWSPAY1XML.xsd "&gt;
  &lt;P&gt;
    &lt;PYH_NUM&gt;&lt;/PYH_NUM&gt;
    &lt;PYH_PAYTYP&gt;BNRCN&lt;/PYH_PAYTYP&gt;
    &lt;PYH_BPR&gt;GB003&lt;/PYH_BPR&gt;
```

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wss="http://www.adonix.com/WSS" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header/>
  <soapenv:Body>
    <wss:run soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <callContext xsi:type="wss:CAdxCallContext">
        <codeLang xsi:type="xsd:string">BRI</codeLang>
        <poolAlias xsi:type="xsd:string">WSSEED</poolAlias>
        <poolId xsi:type="xsd:string"/>
        <requestConfig xsi:type="xsd:string">
          <![CDATA[adxwss.optreturn=JSON&adxwss.beautify=true&adxwss.trace.on=off]]>
        </requestConfig>
      </callContext>
      <publicName xsi:type="xsd:string">AOWSIMPORT</publicName>
      <inputXml xsi:type="xsd:string">
        <![CDATA[
        {"GRP1": {
            "I_MODIMP": "WSPAY1",
            "I_AOWSTA": "NO",
            "I_EXEC": "REALTIME",
            "I_RECORDSEP": "|",
            "I_FILE":
"P;;BNRCN;;GB003;BRI;220000;;CHQ;GB011;GB1GB;2;GBP;1;1;0;20210211;20210211|D;RECGB;BRI;22000
0;GB003;SAINV;SIN1303GB000004;;GB003;20200331;GB011;GBP;1;;0;;30157|END"
            }
        }]]>
      </inputXml>
    </wss:run>
  </soapenv:Body>
</soapenv:Envelope>
```

Sub-program Classic Web Services


```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wss="http://www.adonix.com/WSS">
  <soapenv:Body>
    <wss:runResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <runReturn xsi:type="wss:CAdxResultXml">
        <messages xsi:type="soapenc:Array"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
soapenc:arrayType="wss:CAdxMessage[0]"/>
        <resultXml xsi:type="xsd:string">{
  "GRP1": {
    "I_MODIMP": "WSPAY1",
    "I_AOWSTA": "NO",
    "I_EXEC": "REALTIME",
    "I_RECORDSEP": "|",
    "I_FILE": "",
    "O_REQNUM": "0",
    "O_STATUS": "0",
    "O_MESSA": ""
  }
}</resultXml>
```

3.2
Programmatic invocation

Programmatic Invocation of Web Service calls

It’s all very well using Classic SOAP Web Services and SoapUI/Postman to invoke Web Service calls during the Development and Testing phases of a Project, but we need to be able to invoke the calls on-demand once the Web Services start being used “for real”.

Again, the different Operations have different parameter-lists:

getDescription (context, web-service)

query (context, object, key, num-lines)
read (context, object, key)
save (context, object, key)
modify (context, object, key, data)
delete (context, object, key)

actionObject (context, object, action, data)

run (context, sub-program, key)

So, typically, the calls might look something like:

```
resultXML = webService.modify(callContext, WEB_SERVICE_NAME, paramKey,
                    xmlInput.ToString());
resultXML = webService.deleteLines(callContext, WEB_SERVICE_NAME, paramKey,
                    blockKey.ToString(), lineKeys);
resultXML = webService.query(callContext, WEB_SERVICE_NAME, paramKey, nolines);
resultXML=  webService.run(callContext, WEB_SERVICE_NAME, xmlInput.ToString());
resultXML = webService.read(callContext, WEB_SERVICE_NAME, paramKey);
```

The actual code should look similar depending on the Development Environment being used.

The following slides show the general layout of a Web Service call to create a new instance of Sales Order (SOH Object).

Pseudo Code for a typical “Save” Call

This would be similar for any of the other operations but extra parameters may be required (for example for the objectKeys in a Read.

```
<< Set up Web Services context >>

<<  Generate payload  >>

// Call web service
resultXML = x3WebService.save(callContext, WEB_SERVICE_NAME, xmlInput.ToString());

<< Deal with resultXML >>
```

Typical code to instantiate the Web Service and even throw-in some requestConfig directives for good measure!

```
// Set up Web Services context

CAdxCallContext callContext = new CAdxCallContext();
CAdxWebServiceXmlCCService x3WebService = new CAdxWebServiceXmlCCService();
CAdxResultXml resultXML = new CAdxResultXml();

callContext.codeLang = "BRI"; // Connection language
callContext.codeUser = "admin"; // X3 user
callContext.password = "admin"; // X3 password
callContext.poolAlias = "WSSEED"; // Connection pool name

callContext.requestConfig =
"adxwss.trace.on=on&adxwss.trace.size=16384&adonix.trace.on=on&adonix.trace.level=3&adonix.trace.size=8";
```

Create the payload string – this will entail populating more than one Screen Group

```
StringBuilder xmlInput = new StringBuilder("<?xml version=\"1.0\" encoding=\"UTF-8\"?>");
xmlInput.Append("<PARAM>");

// Create Order Header
populateGroup(xmlInput.,"SOH0_1");
populateGroup(xmlInput.,"SOH1_1");
populateGroup(xmlInput.,"SOH1_4");
// etc…

// Add order lines
addLines(xmlInput, orderDetails);

// Create Order Footer
populateGroup(xmlInput.,“ADB1_1");
populateGroup(xmlInput.,“ADB2_1");
// etc…

xmlInput.Append("</PARAM>");
```

The Web Service operation will return the results into the resultXML structure, and the program can deal with the information as appropriate.

```
// Call web service
resultXML = x3WebService.save(callContext, “WSSOH”, xmlInput.ToString());
```

```
displayMessages(resultXML);

if (resultXML.status == 0)
{
    Status.Text = "Status: NOT OK";
}
else
{
    Status.Text = "Status: OK";
    displayData(resultXML);
}
```

Chapter 4
A word about ADCs

ADC was available via the old V6 Web Services configured in the SafeX3 Console, but these have been deprecated for some time – we won't look at these now,

In Version 12, ADC has been re-implemented in an updated form and this now uses a mixture of both Object and Sub-program Classic SOAP Web Services.

The ADCs use calls to Web Services with names starting CWS, and it also uses calls to AOWSIMPORT to import transactions using Import Templates starting with CWS:

Import/export templates

Clear filter

Template	Description	Object	File type
CWS			
CWSLPNG	License plate number grouping	LPNOPE	Flat
CWSLPNO	License plate number operation	LPNOPE	Flat
CWSPTH	CWS Purchase receipt	PTH	Flat
CWSSCSL	Stock change by LPN	SCSL	Flat
CWSSMO	CWS Misc. Issue	SMO	Flat
CWSSMR	CWS Misc. Receipt	SMR	Flat

Web services

Clear filter

Publication	Description	Object	Transaction	Processing
CWS				
CWSACCNUM	Renumber			CWSACCNUMS
CWSSCS	API Stock Change	SCS	ALL	
CWSSIS	API InterSite Transfer	SCS	STD	
CWSSST	SubContract transfer	SCS	TRF	
CWSVXABP	Pick ticket			CWSVXABPS
CWSVXACC	Stock count transaction			CWSVXACCS
CWSVXALR	Location reordering			CWSVXALRS
CWSVXASY	Putaway			CWSVXASYS

- Note : Currently, 2022R1, ADC always uses the first available Web Services Pool – this means ADCs can only connect to that single Folder. This is with Development to change.

- The New ADCs also use GraphQL to retrieve data for operations such as Browsing Products.

Chapter 5
How to use RESTful Web Services

RESTful Web Services

There are two types of RESTful Web Service calls:

1. X3 Facet REST Web Service

   X3 allows operations to be invoked on its data for any Object which has a Class – the Operations equate to Facets. This is based on SData 2.0 protocol but has “api1” rather than “sdata” in the URL.

2. External REST Web Service

   An X3 4GL Script can invoke a call to an external RESTful API using ASYRRESTCLI .EXEC_REST_WS.

   This mechanism allows Developers to wrap their RESTful Web Service calls up in a Sage X3 4GL program which might be run as a Function within Sage X3 – for example, a 4GL Program could be written to extract information from a third-party Web Site using RESTful web-call to retrieve Currency Exchange Rates on a daily basis

RESTful Web Services

1. X3 Facet REST Web Service

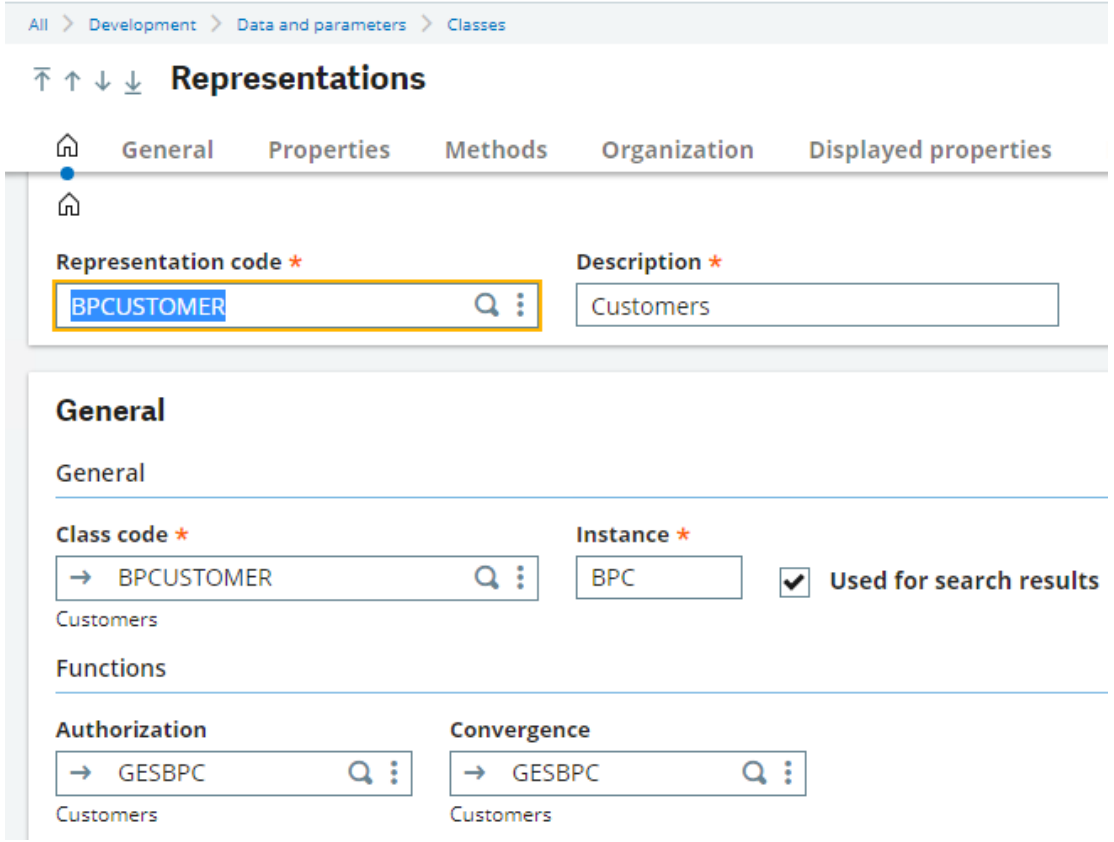
Every resource of the application (a supplier, a customer, a sales order, a product, a user, etc.) is identified by a simple URL. Through this URL, an external component can read the data of a resource, update it or even (if allowed by the business logic) delete it. An external component can also query a list of resources (a list of customers for example), with an optional criteria, and can as well invoke service operations on resources or list thereof. This Web API is available on every *representation* which has been appropriately configured.

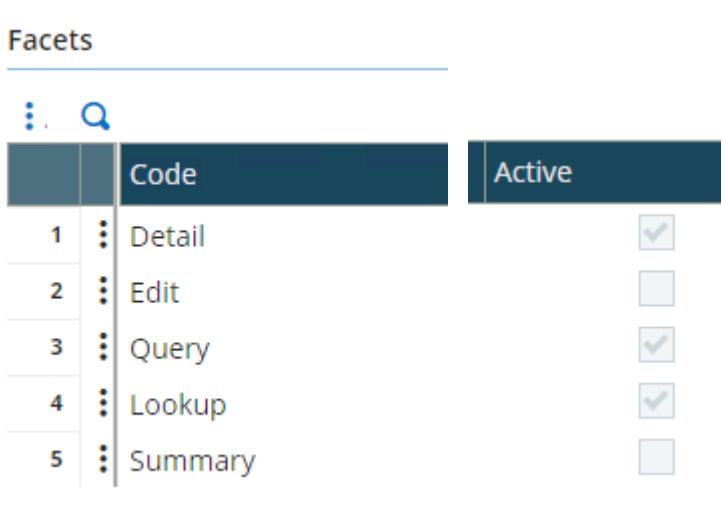
The actions are restricted to those ***Facets*** which are active in the specified Representation, and the fields which are available in a particular Facet are determined in the Available Properties tab of the Representation.

Representations are defined in Development > Data and Parameters > Classes > Representations (GESASW).

Facet RESTful Web Services

For example, the BPCUSTOMER Representation has $details, $query and $lookup activated:





Facet RESTful Web Services

The Facets where the fields BPCUSTOMER.BCGCOD, BPCSTA and BPCNUM are available is determined as follows:

Properties

217 Records

	Alias	Filter P	Entry P	Query	Initial status	Detail	Initial status	Edit
1	BCGCOD	No	No	No	Active	Yes	Active	No
2	BPCSTA	No	No	Yes	Active	Yes	Active	No
3	BPCNUM	No	No	Yes	Active	Yes	Active	No

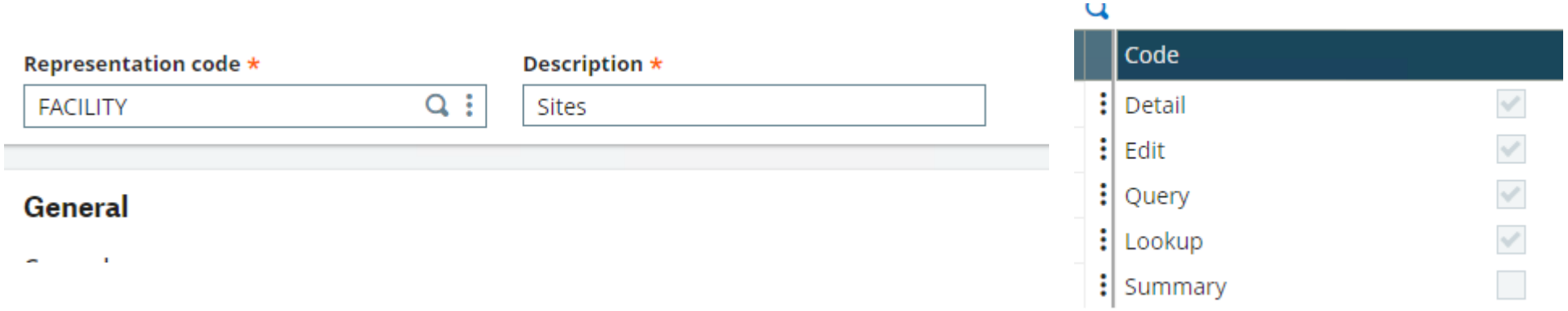
217 Records Page size: 25 1 2

Lookup	Initial status	Summary	Init
No	Active	No	Act
No	Active	No	Act
Yes	Active	No	Act

So, BCGCOD can be seen in the $details Facet only whereas BPCSTA and BPCNUM can also be seen in $query and BPCNUM can be seen in the $lookup.

Facet RESTful Web Services

Another example is FACILITY Representation where the $edit facet is also enabled by default, and fields can be



	Alias		Initial status	Edit	Initial status	Enterable	Hida...	Brea...	Lookup	Initial status
1	FCY		Active	Yes	Active	Yes	No	No	Yes	Active
2	FCYNAM		Active	Yes	Active	Yes	No	No	Yes	Active
3	FCYSHO		Active	Yes	Active	Yes	No	No	No	Active

Facet RESTful Web Services

You can query a list of resources by sending an HTTP GET request like the following URL:

http://myserver:myport/api1/x3/erp/MYENDPOINT/MYCLASS?representation=MYREPR.$query&count=?&key=condition&where=predicate&orderBy=criteria

http://myserver:myport/api1/x3/erp/MYENDPOINT/MYCLASS(key-value)?representation=MYREPR.$details

http://myserver:myport/api1/x3/erp/MYENDPOINT/MYCLASS?representation=MYREPR.$lookup&count=?&key=condition&where=predicate&orderBy=criteria

http://myserver:myport/api1/x3/erp/MYENDPOINT/MYCLASS(key-value)?representation=MYREPR.$summary

where:

myserver:myport is your server name and port number. For example localhost:8124 if the SAFE X3 web server is installed locally.

MYENDPOINT is the name of your Sage X3 endpoint.

Facet RESTful Web Services

MYCLASS is the name of the class that you want to query. It may be a standard class like BPCUSTOMER or a custom class that you have implemented

MYREPR is the name of the representation that you want to query. It may be a standard representation like BPCUSTOMER or a custom one (see the representation dictionaries documentation).

Count=N overrides the default (20) page-size – the query then returns the first N results on execution

Key=condition only returns the keys that satisfy the condition. For example, key=gt.0143 will return the instances where the key is greater than “0143”

Where=predicate filters the results – it is the equivalent of a WHERE-clause. Only those instances satisfying the “predicate” are returned. For example where=left(BPCNUM,2) eq ‘GB’ will only return Customers starting with “GB”.

orderBy=criteria controls how the results are sorted. The criteria is a comma-separated list of field/direction pairs – for example orderBy=BPCNAM desc, BPCNUM asc. Only those properties present on the query-facet can be chosen for sorting data.

Facet RESTful Web Services

Using this api1 adaptation of Sdata, you can obtain information as a JSON string containing a “page” of results with navigation $links to other “pages” for $query and $lookup Facets.

For example, a $query of BPCUSTOMER with 10 record per page and filtered by Customers starting with “GB” could have $links as follows:


```
"$links": {
  "$next": {
    "$url": "http://x3erpv12sqlvm:8124/api1/x3/erp/X3ERPV12_SEED/BPCUSTOMER?representation=BPCUSTOMER.$query&key=gt.GB008&orderBy=BPCNUM&where=left(BPCNUM%2C2)%20eq%20%27GB%27",
    "$type": "application/json;vnd.sage=syracuse",
    "$method": "GET"
  },
  "$last": {
    "$url": "http://x3erpv12sqlvm:8124/api1/x3/erp/X3ERPV12_SEED/BPCUSTOMER?representation=BPCUSTOMER.$query&key=lt&orderBy=BPCNUM&where=left(BPCNUM%2C2)%20eq%20%27GB%27",
    "$type": "application/json;vnd.sage=syracuse",
    "$method": "GET"
  }
```


If the $next link is invoked, then the links on the next page would be


```
"$links": {
  "$previous": {
    "$url": "http://x3erpv12sqlvm:8124/api1/x3/erp/X3ERPV12_SEED/BPCUSTOMER?representation=BPCUSTOMER.$query&key=lt.GB009&orderBy=BPCNUM&where=left(BPCNUM%2C2)%20eq%20%27GB%27",
    "$type": "application/json;vnd.sage=syracuse",
    "$method": "GET"
  },
  "$first": {
    "$url": "http://x3erpv12sqlvm:8124/api1/x3/erp/X3ERPV12_SEED/BPCUSTOMER?representation=BPCUSTOMER.$query&orderBy=BPCNUM&where=left(BPCNUM%2C2)%20eq%20%27GB%27",
    "$type": "application/json;vnd.sage=syracuse",
    "$method": "GET"
  },
  "$last": {
    "$url": "http://x3erpv12sqlvm:8124/api1/x3/erp/X3ERPV12_SEED/BPCUSTOMER?representation=BPCUSTOMER.$query&key=lt&orderBy=BPCNUM&where=left(BPCNUM%2C2)%20eq%20%27GB%27",
    "$type": "application/json;vnd.sage=syracuse",
    "$method": "GET"
  }
}
```

$query RESTful Web Services

This facet returns a limited set of fields for a list of instances of a Class.

You can obtain the description of the Web Service using the prototypes directive

<X3-URL>/api1/x3/erp/solution_folder/$prototypes(‘representation.$query’)

For example:

http://x3erpv12sqlvm:8124/api1/x3/erp/X3ERPV12_SEED/$prototypes('BPCUSTOMER.$query’)

To actually return the first 10 Customers with Customer Code starting ‘GB’:

http://x3erpv12sqlvm:8124/api1/x3/erp/X3ERPV12_SEED/BPCUSTOMER?representation=BPCUSTOMER.$query&count=10&where=left(BPCNUM,2) eq ‘GB’

$query RESTful Web Services

query prototype


```
{
  "$actxUser": "ADMIN",
  "$actxLogin": "admin",
  "$actxFolder": "SEED",
  "$actxSolution": "X3ERPV12",
  "$actxLan": "ENG",
  "$actxLanIso": "en-US",
  "$actxNbLeg": "21",
  "$actxLegCur": "",
  "$device": "desktop",
  "$baseUrl": "http://x3erpv12sqlvm:8124/api1/x3/erp/X3ERPV12_SEED",
  "$baseType": "application/json;vnd.sage=syracuse;vnd.sage.syracuse.representation=x3.erp.SEED",
  "$prototype": "{$baseUrl}/$prototype('{$representation}.$thumb')",
  "$representation": "BPCUSTOMER",
  "$repUrl": "{$baseUrl}/BPCUSTOMER",
  "$instanceUrl": "{$repUrl}/BPC",
  "$url": "{$baseUrl}/BPCUSTOMER?representation=BPCUSTOMER.$query",
  "$itemsPerPage": 300,
  "$title": "{@1503}",
  "$type": "{$baseType}.BPCUSTOMER.$query",
  "$properties": {
    "$resources": {
      "$type": "application/x-array",
      "$item": {
        "$url": "{$baseUrl}/BPCUSTOMER('{$key}')?representation=BPCUSTOMER.$queryItem",
        "$type": "application/json;vnd.sage=syracuse",
        "$description": "{BPCNAM}",
        "$title": "{BPCNAM}",
        "$key": "{BPCNUM}",
        "$value": "{BPCNUM}",
        "$properties": {
          "BPCSTA": {
            "$title": "{@188}",
            "$shortTitle": "{@188}",
            "$type": "application/x-boolean",
            "$columnNum": "1",
            "$isReadOnly": true,
            "$capabilities": "sort,filter"
          },
          "BPCNUM": {
            "$title": "{@3662}",
            "$shortTitle": "{@3662}",
            "$type": "application/x-string",
            "$columnNum": "1",
            "$isMandatory": true,
            "$isReadOnly": true,
            "$capabilities": "sort,filter,filter_upper",
            "$maxLength": 15,
            "$links": {
              "$details": {
                "$title": "{@49838}",
                "$target": "",
                "$type": "application/json;vnd.sage=syracuse",
                "$url": "{$baseUrl}/BPCUSTOMER('{BPCNUM}')?representation=BPCUSTOMER.$details"
              },
              "$lookup": {
                "$title": "{@6028}",
                "$type": "application/json;vnd.sage=syracuse",
                "$url": "{$baseUrl}/BPCUSTOMER?representation=BPCUSTOMER.$lookup"
              }
```


query results


```
{
  "$itemsPerPage": 10,
  "$resources": [
    {
      "$uuid": "fb562f84-0642-4ee4-a93a-d7ef380f77bb",
      "$etag": "2020-12-21T06:22:57Z",
      "BPCSTA": true,
      "BPCNUM": "GB001",
      "BPCSHO": "Indis",
      "BPCNAM": "International Distributors Ltd",
      "RBPAADDLIG1": "15 George Street",
      "RBPAADDLIG2": "",
      "RBPAADDLIG3": "",
      "RPOSCOD": "W1U 3QS",
      "RSAT": "FL",
      "RCTY": "LONDON",
      "RCRY": "GB",
      "RCRYDES": "United Kingdom",
      "BPCTYP": 1,
      "CNTNAM": "000000000000038",
      "REMAIL": "",
      "RCNTTTL": 1,
      "RCNTLNA": "SMITH",
      "RCNTFNA": "Edward",
      "RCNTFNC": 6,
      "RCNTEMA": "",
      "RCNTTEL": "02734355385",
      "RCNTFAX": "",
      "PTE": "CH30NET",
      "PTE_REF": {
        "$title": "30 Days",
        "$description": "Check 30 days date of Invoice"
      },
      "PTE_LEG": "",
      "PTE_LEG_REF": {
        "$title": ""
      }
    },
    {
      "$uuid": "f95b8b58-6e67-4f29-b308-199be463cacf",
      "$etag": "2020-12-21T06:22:57Z",
      "BPCSTA": true,
      "BPCNUM": "GB002",
      "BPCSHO": "Amalgamate",
```

$details RESTful Web Services

The results from such a request are more expansive than those from a simple $query request – this is determined by the Yes/No settings in the Properties tab within Representations.

Again, you can obtain the description of the Web Service using the prototypes directive

<X3-URL>/api1/x3/erp/solution_folder/$prototypes(‘representation.$details’)

For example:

http://x3erpv12sqlvm:8124/api1/x3/erp/X3ERPV12_SEED/$prototypes('BPCUSTOMER.$details')

http://x3erpv12sqlvm:8124/api1/x3/erp/X3ERPV12_SEED/BPCUSTOMER('GB001')?representation=BPCUSTOMER.$details

$details RESTful Web Services

$details prototype


```
{
  "$actxUser": "ADMIN",
  "$actxLogin": "admin",
  "$actxFolder": "SEED",
  "$actxSolution": "X3ERPV12",
  "$actxLan": "ENG",
  "$actxLanIso": "en-US",
  "$actxNbLeg": "21",
  "$actxLegCur": "",
  "$device": "desktop",
  "$baseUrl": "http://x3erpv12sqlvm:8124/api1/x3/erp/X3ERPV12_SEED",
  "$baseType": "application/json;vnd.sage=syracuse;vnd.sage.syracuse.representation=x3.erp.SEED",
  "$prototype": "{$baseUrl}/$prototype('{$representation}.$thumb')",
  "$representation": "BPCUSTOMER",
  "$repUrl": "{$baseUrl}/BPCUSTOMER",
  "$instanceUrl": "{$repUrl}/BPC",
  "$url": "{$baseUrl}/BPCUSTOMER('{$key}')?representation=BPCUSTOMER.$details",
  "$key": "{BPCNUM}",
  "$value": "{BPCNUM}",
  "$title": "{@1503}",
  "$type": "{$baseType}.BPCUSTOMER.$details",
  "$properties": {
    "BCGCOD": {
      "$type": "application/x-string",
      "$maxLength": 5,
      "$isExcluded": true
    },
    "BCGCOD_REF": {
      "$title": "{@707}",
      "$shortTitle": "{@707}",
      "$type": "application/x-reference",
      "$columnNum": "1",
      "$isReadOnly": true,
      "$capabilities": "sort,filter,filter_upper",
      "$item": {
        "$url": "{$baseUrl}/BPCCATEG('{$key}')?representation=BPCCATEG.$lookup",
        "$value": "{BCGCOD}",
        "$key": "{BCGCOD}",
        "$properties": {
          "BCGCOD": {
            "$type": "application/x-string"
          }
        },
        "BCGCOD": "{BCGCOD}",
        "$links": {
          "$lookup": {
            "$title": "{@6028}",
            "$type": "application/json;vnd.sage=syracuse",
            "$url": "{$baseUrl}/BPCCATEG?representation=BPCCATEG.$lookup",
            "$method": "GET"
          },
          "$query": {
            "$title": "{@49616}",
            "$target": "",
            "$type": "application/json;vnd.sage=syracuse",
            "$url": "{$baseUrl}/BPCCATEG?representation=BPCCATEG.$query",
            "$method": "GET"
```


$details results


```
{
  "$uuid": "fb562f84-0642-4ee4-a93a-d7ef380f77bb",
  "$etag": "2022-02-11T16:56:04Z",
  "$actxUser": "ADMIN",
  "$actxLogin": "admin",
  "$actxFolder": "SEED",
  "$actxSolution": "X3ERPV12",
  "$actxLan": "ENG",
  "$actxLanIso": "en-US",
  "$actxNbLeg": "21",
  "$actxLegCur": "",
  "BPAINVNAM": "Corporate",
  "BPAPYRNAM": "Corporate",
  "ABCCLS": 1,
  "ACCCOD": "LOCAL",
  "BCGCOD": "GB",
  "BCGCOD_REF": {
    "$title": "British",
    "$description": "British customers"
  },
  "BPAINV": "CORP",
  "BPAINV_REF": {
    "$description": "Corporate",
    "$title": ""
  },
  "BPAPYR": "CORP",
  "BPAPYR_REF": {
    "$description": "Corporate",
    "$title": ""
  },
  "BPCBPSNUM": "",
  "BPCCDTISR": "",
  "BPCCDTISR_REF": {
    "$title": ""
  },
  "BPCGRU": "GB001",
  "BPCINV": "GB001",
  "BPCNUM": "GB001",
  "BPCPYR": "GB001",
  "BPCPYR_REF": {
    "$title": "Indis",
    "$description": "International Distributors Ltd"
  },
  "BPCRSK": "GB001",
  "BPCSHO": "Indis",
  "BPCSNCDAT": "2022-02-03",
  "BPCSTA": true,
  "BPCTYP": 1,
  "BPCBPD": [
```


$query results


```
{
  "$itemsPerPage": 10,
  "$resources": [
    {
      "$uuid": "fb562f84-0642-4ee4-a93a-d7ef380f77bb",
      "$etag": "2020-12-21T06:22:57Z",
      "BPCSTA": true,
      "BPCNUM": "GB001",
      "BPCSHO": "Indis",
      "BPCNAM": "International Distributors Ltd",
      "RBPAADDLIG1": "15 George Street",
      "RBPAADDLIG2": "",
      "RBPAADDLIG3": "",
      "RPOSCOD": "W1U 3QS",
      "RSAT": "FL",
      "RCTY": "LONDON",
      "RCRY": "GB",
      "RCRYDES": "United Kingdom",
      "BPCTYP": 1,
      "CNTNAM": "000000000000038",
      "REMAIL": "",
      "RCNTTTL": 1,
      "RCNTLNA": "SMITH",
      "RCNTFNA": "Edward",
      "RCNTFNC": 6,
      "RCNTEMA": "",
      "RCNTTEL": "02734355385",
      "RCNTFAX": "",
      "PTE": "CH30NET",
      "PTE_REF": {
        "$title": "30 Days",
        "$description": "Check 30 days date of Invoice"
      },
      "PTE_LEG": "",
      "PTE_LEG_REF": {
        "$title": ""
      }
    },
    {
      "$uuid": "f95b8b58-6e67-4f29-b308-199be463cacf",
      "$etag": "2020-12-21T06:22:57Z",
      "BPCSTA": true,
      "BPCNUM": "GB002",
      "BPCSHO": "Amalgamate",
```

$lookup RESTful Web Services

This facet provides a short list of details for the appropriate instances of the entity.

You can obtain the description of the Web Service using the prototypes directive

<X3-URL>/api1/x3/erp/solution_folder/$prototypes(‘representation.$lookup’)

For example:

http://x3erpv12sqlvm:8124/api1/x3/erp/X3ERPV12_SEED/$prototypes('BPCUSTOMER.$lookup')

http://x3erpv12sqlvm:8124/api1/x3/erp/X3ERPV12_SEED/BPCUSTOMER?representation=BPCUSTOMER.$lookup

This effectively provides the left-list for the Entity.

$lookup RESTful Web Services

lookup prototype


```
{
  "$actxUser": "ADMIN",
  "$actxLogin": "admin",
  "$actxFolder": "SEED",
  "$actxSolution": "X3ERPV12",
  "$actxLan": "ENG",
  "$actxLanIso": "en-US",
  "$actxNbLeg": "21",
  "$actxLegCur": "",
  "$device": "desktop",
  "$baseUrl": "http://x3erpv12sqlvm:8124/api1/x3/erp/X3ERPV12_SEED",
  "$baseType": "application/json;vnd.sage=syracuse;vnd.sage.syracuse.representation=x3.erp.SEED",
  "$prototype": "{$baseUrl}/$prototype('{$representation}.$thumb')",
  "$representation": "BPCUSTOMER",
  "$repUrl": "{$baseUrl}/BPCUSTOMER",
  "$instanceUrl": "{$repUrl}/BPC",
  "$url": "{$baseUrl}/BPCUSTOMER?representation=BPCUSTOMER.$query",
  "$itemsPerPage": 300,
  "$title": "{@1503}",
  "$type": "{$baseType}.BPCUSTOMER.$query",
  "$properties": {
    "$resources": {
      "$type": "application/x-array",
      "$item": {
        "$url": "{$baseUrl}/BPCUSTOMER('{$key}')?representation=BPCUSTOMER.$queryItem",
        "$type": "application/json;vnd.sage=syracuse",
        "$description": "{BPCNAM}",
        "$title": "{BPCNAM}",
        "$key": "{BPCNUM}",
        "$value": "{BPCNUM}",
        "$properties": {
          "BPCSTA": {
            "$title": "{@188}",
            "$shortTitle": "{@188}",
            "$type": "application/x-boolean",
            "$columnNum": "1",
            "$isReadOnly": true,
            "$capabilities": "sort,filter"
          },
          "BPCNUM": {
            "$title": "{@3662}",
            "$shortTitle": "{@3662}",
            "$type": "application/x-string",
            "$columnNum": "1",
            "$isMandatory": true,
            "$isReadOnly": true,
            "$capabilities": "sort,filter,filter_upper",
            "$maxLength": 15,
            "$links": {
              "$details": {
                "$title": "{@49838}",
                "$target": "",
                "$type": "application/json;vnd.sage=syracuse",
                "$url": "{$baseUrl}/BPCUSTOMER('{BPCNUM}')?representation=BPCUSTOMER.$details"
              },
              "$lookup": {
                "$title": "{@6028}",
                "$type": "application/json;vnd.sage=syracuse",
                "$url": "{$baseUrl}/BPCUSTOMER?representation=BPCUSTOMER.$lookup"
              }
```


lookup results


```
{
  "$itemsPerPage": 20,
  "$resources": [
    {
      "$uuid": "b9a69a0c-4c3e-490a-8b36-6d5354e4a282",
      "$etag": "2019-12-15T20:34:18Z",
      "BPCNUM": "AE001",
      "BPCNAM": "Al Rostamani Communications ",
      "BPCTYP": 1,
      "PTE": "CH30NET",
      "PTE_REF": {
        "$title": "30 Days",
        "$description": "Check 30 days date of Invoice"
      },
      "PTE_LEG": "",
      "PTE_LEG_REF": {
        "$title": ""
      }
    },
    {
      "$uuid": "fb08f804-53bb-4afa-a5b7-bd5bc0b3f169",
      "$etag": "2019-12-15T20:34:18Z",
      "BPCNUM": "AE002",
      "BPCNAM": "Al Zahra Computers",
      "BPCTYP": 1,
      "PTE": "CC",
      "PTE_REF": {
        "$title": "30 Days",
        "$description": "CC 30 days date of Invoice"
      },
      "PTE_LEG": "",
      "PTE_LEG_REF": {
        "$title": ""
      }
    },
    {
```

$summary RESTful Web Services

This facet provides another short list of details for the appropriate instances of the entity.

You can obtain the description of the Web Service using the prototypes directive

<X3-URL>/api1/x3/erp/solution_folder/$prototypes(‘representation.$summary’)

For example:

http://x3erpv12sqlvm:8124/api1/x3/erp/X3ERPV12_SEED/$prototypes(COMPANY.$summary')

http://x3erpv12sqlvm:8124/api1/x3/erp/X3ERPV12_SEED/COMPANY(‘GB10’)?representation=COMPANY.$summary

This effectively provides the left-list for the Entity.

$summary RESTful Web Services

summary prototype


```
{
  "$actxUser": "ADMIN",
  "$actxLogin": "admin",
  "$actxFolder": "SEED",
  "$actxSolution": "X3ERPV12",
  "$actxLan": "ENG",
  "$actxLanIso": "en-US",
  "$actxNbLeg": "21",
  "$actxLegCur": "",
  "$device": "desktop",
  "$baseUrl": "http://x3erpv12sqlvm:8124/api1/x3/erp/X3ERPV12_SEED",
  "$baseType": "application/json;vnd.sage=syracuse;vnd.sage.syracuse.representation=x3.erp.SEED",
  "$prototype": "{$baseUrl}/$prototype('{$representation}.$thumb')",
  "$representation": "COMPANY",
  "$repUrl": "{$baseUrl}/COMPANY",
  "$instanceUrl": "{$repUrl}/CPY",
  "$url": "{$baseUrl}/COMPANY('{$key}')?representation=COMPANY.$summary",
  "$key": "{CPY}",
  "$value": "{CPY}",
  "$title": "{@1343}",
  "$type": "{$baseType}.COMPANY.$summary",
  "$properties": {
    "CPY": {
      "$title": "{@1343}",
      "$shortTitle": "{@1343}",
      "$type": "application/x-string",
      "$columnNum": "1",
      "$isMandatory": true,
      "$capabilities": "sort,filter,filter_upper",
      "$maxLength": 5
    },
    "CPYNAM": {
      "$title": "{@1644}",
      "$shortTitle": "{@1644}",
      "$type": "application/x-string",
      "$columnNum": "1",
```


summary results


```
{
  "$uuid": "4b25ae7b-c2a1-4fcc-95dc-fad0e41d71eb",
  "$etag": "2020-12-21T06:24:05Z",
  "$actxUser": "ADMIN",
  "$actxLogin": "admin",
  "$actxFolder": "SEED",
  "$actxSolution": "X3ERPV12",
  "$actxLan": "ENG",
  "$actxLanIso": "en-US",
  "$actxNbLeg": "21",
  "$actxLegCur": "",
  "ACCCUR": "GBP",
  "ACCCUR_REF": {
    "$title": "Bri pound",
    "$description": "British Pound",
    "$symbol": "#",
    "$scale": 2,
    "$precision": 13
  },
  "CPYBPA": [
    {
      "$uuid": "227c6588-44bc-4868-81e8-9c506bd617a4",
      "BPADES": "Corporate"
    }
  ],
  "CPY": "GB10",
  "CPYLEGFLG": true,
  "CPYLOG": "",
  "CPYNAM": "GB Discrete",
  "CPYSHO": "GBDiscrete",
  "CRN": "",
  "CRY": "GB",
  "CRY_REF": {
    "$title": "United Kingdom",
  "$description": "United Kingdom"
```

External RESTful Web Services

External RESTful Web Services

These are RESTful calls to external Sites which can provide information to be used by X3.

For example, a third-party Site provides Exchange Rate information which is updated on a daily basis.

X3 provides a function which can invoke a call to such Sites – it is called **ASYRRESTCLI .EXEC_REST_WS.**

This is described in the Online Help link

Api asyrrestcli | (sageerpx3.com)

An example is described in the Sage City Blog

How to call an external REST web services in classic functions - Sage X3 Support - Sage X3 - Sage City Community

External RESTful Web Services

The call prototype for SYRRESTCLI.EXEC_REST_WC is

ASYRRESTCLI.EXEC_REST_WC (NAME,HTTPMETHOD,SUBURL,PARAM_COD,PARAM_VAL, HEADER_COD,HEADER_VAL,DATA,FUTURE,RETURNS,RESHEAD,RESBODY)

NAME	The name of the Web Service described in Administration > Administration > Web Services > REST web services
HTTPMETHOD	The HTTP Method USED (CAN BE GET, PUT, POST, or Delete).
SUBURL	The Service Called in the REST Web Service - added to the Base URL from the REST web services.
PARAM_COD	Additional property-names accepted by the Service as part of the URL parameter-list
PARAM_VAL	The values being paired with the parameters in PARAMS_COD.
HEADER_COD	Additional property-name sent as part of the Request Header.
HEADER_VAL	The values being paired with the parameters in HEADER_COD.
DATA	The data sent with POST and PUT HTTP METHODs to update the external Site.
FUTURE	If 1, the Web Service is called in 'future' mode; If 0, the Web Service is called in 'wait' mode.
RETURNS	If empty, the whole JSON feed is returned in RESBODY; If not EMPTY, only THE Value OF the corresponding property is returned in the RESBODY.
RESHEAD	The Response Header returned by the Web Service
RESBODY	The Response Body returned by the Web Service

External RESTful Web Services

Example : Obtain Exchange Rates for multiple Currencies from the Web Site described below.

The external web-site, https://xecdapi.xe.com, provides a service where it returns a comma-separated list of exchange-rates relating to the parameters “from” and “to” – the “from” is the Base Currency and the “to” is a comma-separated list of Currency Codes to return.

https://xecdapi.xe.com/v1/convert_from.json/?from=xxx&to=yyy1,yyy2,yyy3...

Once the Exchange Rates have been obtained, X3 needs to be update with those values.

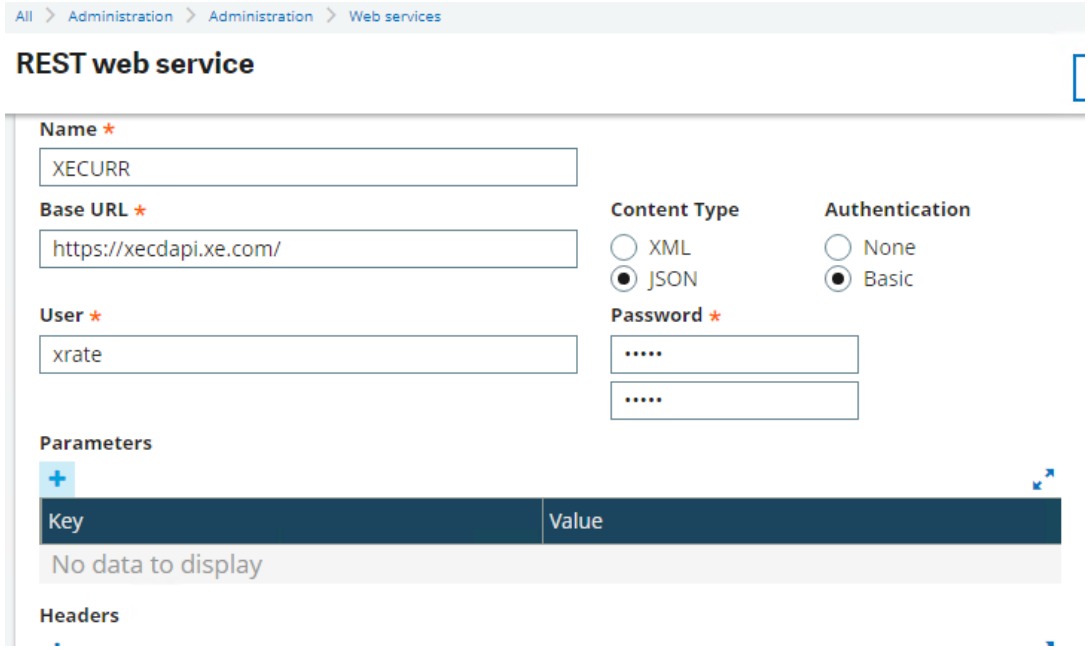
This is a daily task which could be manually invoked, but it should ideally be done as a Recurring Task controlled by the X3 Batch Server.

The list of Currencies is derived from the TABCUR Table, and the objective is to update the TABCHANGE table with the new Exchange Rates.

The Web Site restricts the number of RESTful calls that can be made in a 24-hour period, so specifying multiple Currencies in the “to” parameter gets around this.

External RESTful Web Services

The first step is to define a Web Service in Administration > Administration > Web Services > REST Web Services option



External RESTful Web Services

Create a 4GL Script to call GET_RATE for all the relevant Currencies – loop through the currencies and call GET_RATE with one as the Base Currency and all the others as a list and it will return the Exchange Rates for that Base Currency each time.

GET_RATE will set up the parameters for ASYRRESTCLI.EXEC_REST_WS to call the Rest Web Service “XECURR”.

ASYRRESTCLI.EXEC_REST_WS will return the Exchange Rates in RESBODY and GET_RATE will process that string to split out each Exchange Rate.

Once an Exchange Rate has been extracted, any updates to X3 Data can be executed before moving on to the next Base/Foreign-list pair.

External RESTful Web Services

```
# RCUR          List of Foreign Currencies
# RBASE         Base Currency
# RPDATE        Process Date
# CURRENCIES    Array of “To” Currencies
# CURR_COUNT    How many “To” Currencies are there?


Funprog GET_RATE(RCUR,RBASE,RPDATE,CURRENCIES,CURR_COUNT)

Value Char RBASE()
Value Char RCUR()
Value Date RPDATE()
value Char CURRENCIES()()
Value Integer CURR_COUNT

Local Char XHSUB(60)
Local Char XOK(30)
Local Integer RETVAL
Local Decimal XRATE
```

External RESTful Web Services

```
# API STRING URL

XHSUB = "v1/convert_from/"

# Additional parameters to URL BASE CURR + CURR TO CONVERT
Local Char PCOD(100)(1..10),PVAL(100)(1..10)
PCOD(1)= "from"   : PVAL(1)= chr$(34)+RBASE+chr$(34)
PCOD(2)="to"     : PVAL(2)=chr$(34)+RCUR+chr$(34)

# Additional header values
Local Char HCOD(100)(1..10),HVAL(100)(1..10)
HCOD(1)="" : HVAL(1)=""

# Store result header and body in clob
Local Clbfile RESHEAD(0),RESBODY(0)

# Calling the service  Details of Function Below

RETVAL = func ASYRRESTCLI.EXEC_REST_WS(
&    "XECURR","GET",XHSUB,PCOD,PVAL,HCOD,HVAL,"{}",0,"",RESHEAD,RESBODY)
```

External RESTful Web Services

```
If RETVAL=200
  XRS= instr(1,RESBODY,"from")
  If XRS=0
      XRATE=0
      Call ECR_TRACE("XRATE 0 ("-num$(I)+")",-1) From GESECRAN
  Else
    For I = 1 To CURR_COUNT
      XRS1=instr(XRS+1,RESBODY,CURRENCIES(I))
      XRSB=instr(XRS1+1,RESBODY,":")
      XRSE=instr(XRSB+1,RESBODY,",")

      If XRSE =0 : XRSE=instr(XRSB+1,RESBODY,"}") : Endif

      XRATE=val(mid$(RESBODY,XRSB+1,XRSE-XRSB-1))
      XOK= func PROCESS_RATES(RBASE,CURRENCIES(I),XRATE,RPROC_DATE)
   Next I
  Endif
Else
    Call RECORD_ERROR
    XRATE=-9999
Endif
```

External RESTful Web Services

Now that this RESTful call has been incorporated into an X3 4GL program, it could be run as a Recurring Task within X3's Batch Server so it is automatically called every morning to update the relevant Currency Exchange Rates within X3.

This is an example of how RESTful Web Services extends the reach of X3.

As well as enabling X3 to obtain information from external sites, it is possible to *update* the data in external sites with RESTful calls – these will be of HTTP-Type PUT, POST or DELETE as appropriate.

When things go wrong…

When things go wrong…

1. What is the Object in question – if it’s BPC, ITM or SOH, then it’s been “verified”, and any issues may be passed to Level3 if required. Otherwise, Support will endeavour to assist, but you may be asked to engage with Centre of Excellence for assistance.
2. Deploy Debug Directives adxwss.trace.on=on&adxwss.trace.size=16384&adonix.trace.on=on&adonix.trace.level=3&adonix.trace.size=8
3. “AWEB:RECUPHDAT” error – this is because the Web Service needs publishing
4. Check the Syracuse Logs (especially the W*.log files devoted to Web Services)
5. Use SPEWS to add OpenLog/CloseLog calls to be executed at run-time or enabling the Syracuse level logging via Administration> Administration> Settings> Global Settings. You can then change the logging levels of the X3 components to give more information, for example in the "x3Comm" section you can enable "Debug" level logging for the "soap" component which may give you the additional information you need – see Appendix
6. “Sales Reps : \Record does not exist” followed by “Field error [M:BPC1]REP(2)”

The REP(2) is down to the fact that there is a second <ITM></ITM> line which is empty – SUBBPC thinks it should have a value.

When things go wrong…

7. Some screens may have changed – for example

   “@SEED.TRT/WKBPC$adx 195 : Variable nonexistent BPADES”

   Looking at WKBPC.src, BPADES *seems* to be in [M:A_W6] which is BPC4 (Ship-to Customer), but in reality it is in [M:A_W2] which is BPABPC.

   If [L]BPADES_1(WW_I)<>"" : [M:A_W6]BPADES(WW_I) = [L]BPADES_1(WW_I) : [L]A_W6_1+=1 : Endif

   This is correct in V11 and the issue is corrected by re-publishing the Service.

   If [L]BPADES_1(WW_I)<>"" : [M:A_W2]BPADES(WW_I) = [L]BPADES_1(WW_I) : [L]A_W2_1+=1 : Endif

   I think it had been mixed-up with BPADESBPC4 which *is* in BPC4 screen:

   If [L]BPADESBPC4_5(WW_I)<>"" : [M:A_W6]BPADESBPC4(WW_I) = [L]BPADESBPC4_5(WW_I) : [L]A_W6_1+=1 : Endif

Appendices

Appendix A: Additional reading

Online Help

Overview of Web Services

- https://online-help.sageerpx3.com/erp/12/wp-static-content/static-pages/en_US/v7dev/api-guide_soap-web-services.html
- Web services integration | (sageerpx3.com)
- Integration guide (sageerpx3.com)

AOWSIMPORT

- **Using Import/Export Templates** https://online-help.sageerpx3.com/erp/12/wp-static-content/static-pages/en_US/v7dev/api-guide_api-soap-import-export.html
- **Examples of Import/Export Web Services** https://online-help.sageerpx3.com/erp/12/wp-static-content/static-pages/en_US/v7dev/api-guide_api-soap-import-export-example.html

Appendix A: Additional reading

REST Web Services

- **REST Web Services** https://online-help.sageerpx3.com/erp/12/staticpost/rest-web-services/?highlight=REST **– this also discusses SOAP.**

- https://online-help.sageerpx3.com/erp/12/wp-static-content/static-pages/en_US/v7dev/integration-guide_ws-overview.html#testing-interactively

- https://online-help.sageerpx3.com/erp/12/staticpost/rest-web-services/?highlight=actionObject

- Api asyrrestcli | (sageerpx3.com)

- https://online-help.sageerpx3.com/erp/12/wp-static-content/static-pages/en_US/v7dev/integration-guide_ws-query.html

Appendix A: Additional reading

Knowledgebase articles

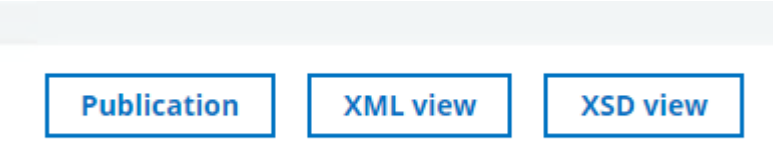
- How do I configure and test SOAP web services in Sage X3 Product Update 9 and later?
- Converting a Read-result into a template Save-Payload

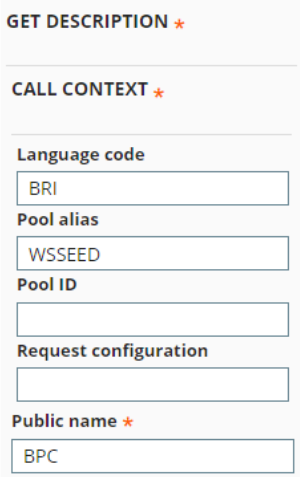
Sage City

- Illustrated guide to tracing Web Services - Sage X3 UK Support & Insights - Sage X3 UK - Sage City Community
- https://www.sagecity.com/us/sage_erp_x3/b/sageerp_x3_product_support_blog/posts/how-to-use-rest-web-services-to-get-and-post-a-record
- How to call an external REST web services in classic functions - Sage X3 Support - Sage X3 - Sage City Community

Appendix B: The Web Service Object XML

The getDescription Operation is equivalent to the “XML view” button within the Web Services option.





The XML file obtained from both mechanisms describes the screens and fields available to the Web Service, the Local Menus related to certain fields, the left-list (equivalent to “query” Operations), the Operations, and the built-in indexes.

Appendix B: The Object XML (getDescription)


```
▼<ADXDOC PNA="BPC" NAM="OBPC" TIM="20220221155536" OB3="BPC" FOL="SEED" SOL="X3ERPV12" WRP="WJBPC" USER="ADMIN" VER="6.30" HEAD="1">
 ▶<ADXDATA>
  ...
  </ADXDATA>
 ▼<ADXKEY>
  ▼<GRP NAM="LEFTLIST" DIM="10000">
     <FLD NAM="BPCNUM" MOD="Display" TYP="Char" LEN="15" C_FRA="Client" C_ENG="Customer" C_ITA="Cliente" C_POR="Cliente" C_SPA="Client
     شريك العمل" C_AUS="Customer"/>
     <FLD NAM="BPCNAM" MOD="Display" TYP="Char" LEN="35" C_FRA="Raison sociale" C_ENG="Company name" C_ITA="Ragione sociale" C_POR="Ra
     C_GER="Unternehmensname" C_POL="Nazwa firmy" C_CHI="公司名称" C_ARB="إسم الشركة" C_AUS="Company name"/>
     <FLD NAM="BPCSHO" MOD="Display" TYP="Char" LEN="10" C_FRA="Intitulé court" C_ENG="Short description" C_ITA="Descr. breve" C_POR="
     C_GER="Kurzbezeichnung" C_POL="Krótki tytuł" C_CHI="简称" C_ARB="وصف قصير" C_AUS="Short description"/>
     <FLD NAM="BPCTYP" MOD="Display" MEN="401" TYP="Char" LEN="15" C_FRA="Type" C_ENG="Type" C_ITA="Tipo" C_POR="Tipo" C_SPA="Tipo" C_
     <FLD NAM="POSCOD" MOD="Display" TYP="Char" LEN="10" C_FRA="C postal" C_ENG="Postal code" C_ITA="CAP" C_POR="C.postal" C_SPA="C.Po
     C_ARB="الرمز البريدي" C_AUS="Postal code"/>
     <FLD NAM="PTE" MOD="Display" TYP="Char" LEN="15" C_FRA="Paiement" C_ENG="Terms" C_ITA="Pagamento" C_POR="Pagamento" C_SPA="Pago"
     C_AUS="Terms"/>
    </GRP>
  </ADXKEY>
 ▶<ADXMEN>
  ...
  </ADXMEN>
 ▼<ADXSER>
    <MET ID="READ" C_FRA="Lire" C_ENG="Read" C_ITA="Leggi" C_POR="Ler" C_SPA="Leer" C_BRI="Read" C_GER="Lesen" C_POL="Odczyt" C_CHI="读
    <MET ID="CREATE" C_FRA="Créer" C_ENG="Create" C_ITA="Crea" C_POR="Criar" C_SPA="Crear" C_BRI="Create" C_GER="Anlegen" C_POL="Utwórz
    <MET ID="MODIFY" C_FRA="Modifier" C_ENG="Modify" C_ITA="Modifica" C_POR="Modificar" C_SPA="Modificar" C_BRI="Modify" C_GER="Ändern"
    <MET ID="DELETE" C_FRA="Supprimer" C_ENG="Delete" C_ITA="Elimina" C_POR="Suprimir" C_SPA="Borrar" C_BRI="Delete" C_GER="Löschen" C_
    <MET ID="LIST" C_FRA="Liste" C_ENG="List" C_ITA="Lista" C_POR="Lista" C_SPA="Lista" C_BRI="List" C_GER="Liste" C_POL="Lista" C_CHI=
  </ADXSER>
 ▼<ADXREAD TAB="BPCUSTOMER">
  ▼<GRP DIM="1" NAM="KEYS">
     <FLD NAM="BPCNUM" TYP="Char" MOD="Input" LEN="15" C_FRA="Client" C_ENG="Customer" C_ITA="Cliente" C_POR="Cliente" C_SPA="Cliente"
     شريك العمل" C_AUS="Customer"/>
    </GRP>
  </ADXREAD>
</ADXDOC>
```

```
▼<ADXDATA>
 ▼<GRP NAM="BPC0_1" TYB="List" DIM="1">
    <FLD NAM="BCGCOD" X3BLORIG="BPC0_1" X3FLORI
    C_GER="Kategorie" C_POL="Kategoria" C_CHI="
    <FLD NAM="ZBCGCOD" X3BLORIG="BPC0_1" X3FLOR
    <FLD NAM="BPCSTA" X3BLORIG="BPC0_1" X3FLORI
    C_GER="Aktiv" C_POL="Aktywny" C_CHI="激活" (
    <FLD NAM="BPCNUM" X3BLORIG="BPC0_1" X3FLORI
    C_GER="Kunde" C_POL="Klient" C_CHI="客户" C_
    <FLD NAM="BPCNAM" X3BLORIG="BPC0_1" X3FLORI
  </GRP>
 ▼<GRP NAM="BPRC_1" TYB="List" DIM="1">
    <FLD NAM="BPRSHO" X3BLORIG="BPRC_1" X3FLORI
    corta" C_BRI="Short title" C_GER="Kurzbezei
    <FLD NAM="BPRLOG" X3BLORIG="BPRC_1" X3FLORI
    C_POL="Akronim" C_CHI="首字母缩合词" C_ARB="
    <FLD NAM="BPRNAM" X3BLORIG="BPRC_1" X3FLORI
    <FLD NAM="LEGETT" X3BLORIG="BPRC_1" X3FLORI
    C_SPA="Persona física" C_BRI="Physical pers
    <FLD NAM="BPRFBDMAG" X3BLORIG="BPRC_1" X3FL
    C_POR="interdito mailing" C_SPA="Prohibido
    <FLD NAM="CRY" X3BLORIG="BPRC 1" X3FLORIG="
```

```
▼<ADXMEN>
 ▼<MNU NO="1">
    <VAL IND="1" C_FRA="Non" C_ENG="No" C_
    <VAL IND="2" C_FRA="Oui" C_ENG="Yes" C
  </MNU>
 ▼<MNU NO="202">
    <VAL IND="1" C_FRA="Cours du jour" C_E
    C_AUS="Daily rate"/>
    <VAL IND="2" C_FRA="Cours du mois" C_E
    C_AUS="Monthly rate"/>
    <VAL IND="3" C_FRA="Cours moyen" C_ENG
    C_AUS="Average rate"/>
    <VAL IND="4" C_FRA="Cours DEB" C_ENG="
    pliku dokumentu celnego" C_CHI="海关完
  </MNU>
 ▼<MNU NO="212">
    <VAL IND="1" C_FRA="Classe 'A'" C_ENG=
    <VAL IND="2" C_FRA="Classe 'B'" C_ENG=
    <VAL IND="3" C_FRA="Classe 'C'" C_ENG=
    <VAL IND="4" C_FRA="Classe 'D'" C_ENG=
  </MNU>
```

Thank you