REST Web Services

Richard Perrins — 24th May 2023

Contents

Introducing Sage X3 Web Services

Inbound REST Web Services

Outbound REST Web Services

Simple Examples

Things to think about

Recap and conclusions

Appendix

Introducing Sage X3 Web Services

What are Sage X3 Web Services?

A web service is a piece of 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

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

As discussed in previous BP Day sessions, there are two types of Web Service:

1) SOAP Web Services
2) REST Web Services

This particular session concentrates on REST Web Services.

What types of REST Web Services are there?

REST stands for Representational State Transfer and is a style of interface which operates on representations of application data, be it for requests to list, read, update, create or delete the underlying instances of the requested Representation. This contrasts with SOAP Web Services which are based on a Protocol associated with applications.

REST Web Services are tailored for the Internet and they are invoked via a 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.

There are two types of REST Web Services:

1. Inbound

2. Outbound

In the rest of this Presentation, we'll be talking from the X3 perspective.

What types of REST Web Services are there?

Inbound

As mentioned in the Spring 2022 Tips and Tricks session, X3 provides the ability to invoke RESTful calls to access X3 Data from External sources.

This enables other Sites to read and, potentially, even update the X3 Database via X3 Representations.

Note that Inbound Web Services calls do not use SOAP Web Service Pools – they use the SData2.0 protocol associated with X3, but with “api1” rather than “sdata” in the URL.

Outbound

X3 provides a mechanism to programmatically access, and potentially, update External Site Data through calls to Web APIs published by those External Sites. These calls use Sage X3 4GL built-in functions.

What types of Authentication are there?

Inbound

Inbound REST calls, accessing X3 Representations, require Sage X3 User credentials.

Outbound

Outbound REST calls must adhere to the Authentication methods in the specification of the External Site's APIs – this may be using:

1. None (no credentials)

2. Bearer (Credentials) – be they simple user/password combinations, JWT or OAuth2 Tokens obtained from a Token Site.

The Sage X3 code must provide the required Authentication as part of the Call.

Pros & Cons of Inbound and Outbound calls

When integrating with External Sites, you must decide which type of REST Web Service to use, Inbound or Outbound. Each has its Pros and Cons.

The Inbound approach is more appropriate when the External Site is driving the data exchange – this could be when data is updated in the External Site, and the changes need to be propagated to X3 in a timely fashion (for example, an Order Tracking Site where a Delivery has been made – this needs to be reported to X3 as soon as possible).

By their very definition, Inbound calls have a simpler specification, but the External Site must be amended to be “aware” of the X3 REST Web Services – this has a potential administration overhead when the External Site accesses multiple X3 Environments.

Setting up Inbound calls may involve amending the X3 Dictionary – this could be for a number of reasons:

1. New Tables, Objects, Classes and Representations to interface with the External Site

2. Amend existing Tables, Objects, Classes and Representations to interface with the External Site (e.g. extra fields)

3. Add additional 4GL Code for preparing X3 Data for the External Site or to process information from the External Site

Pros & Cons of Inbound and Outbound calls

The Outbound approach is more appropriate when X3 is driving the data exchange – i.e. some change in X3 Data needs to be propagated to the External Site in a timely fashion (e.g. Sales Order Tracking where an Order has been Created in X3, and needs to be recorded by the Tracking Site).

X3 is responsible for handling the calls to the External Site – how to prepare for a call, and what to do with the results of the calls.

Outbound calls have a pre-defined specification, and the Site does not need to be “aware” of X3 as such – it is up to X3 to invoke the External Site’s APIs correctly – so the External Site is more universally usable as long as the Specification is adhered to.

External Sites may have particular Authentication requirements, so these need to be taken into account by X3’s calls.

X3 must decide which calls, and when to invoke them – this may be on an ad-hoc basis or at scheduled times/intervals.

If the data held in the External Site is universally applicable, then it’s unlikely that authentication is required – if it specific to a Business, then authentication is more likely – it will identify which Business is making the request, and the External Site will only deal with data for that Business.

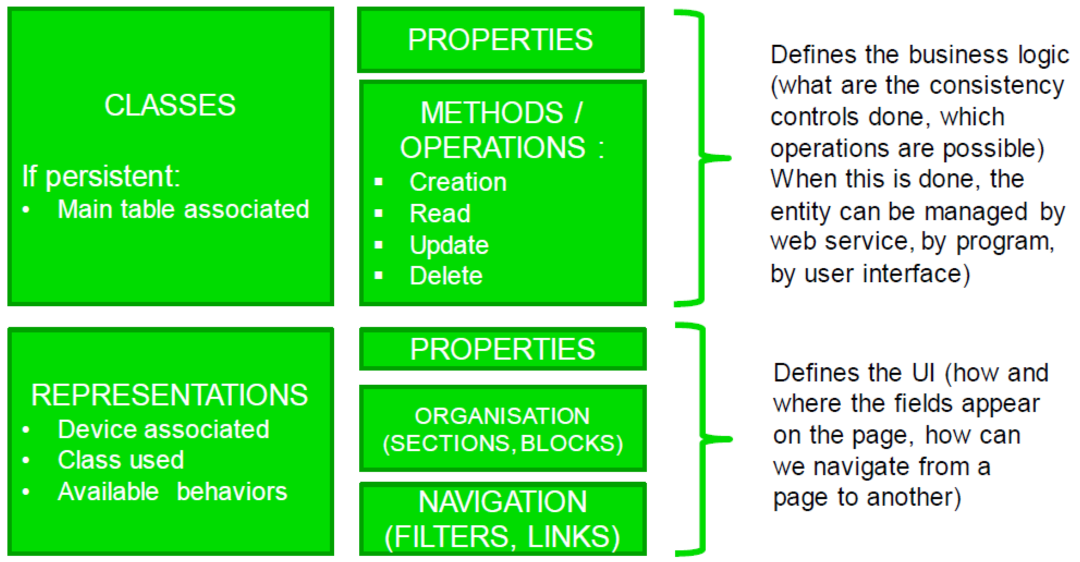
Inbound REST Web Services

Inbound REST Web Service calls

These are calls from an External Site requesting query/inquiry or creation/deletion/update operations on X3 Data.

Sage X3 includes standard Representations which "expose" the data defined by the associated Classes, and defines the available Operations on said Classes – note that most Representations are basically read-only (Query, Lookup and Detail), but they could be updateable (Edit).

Classes and representations



CLASSES
If persistent:
• Main table associated
PROPERTIES
METHODS / OPERATIONS :
▪ Creation
▪ Read
▪ Update
▪ Delete
Defines the business logic (what are the consistency controls done, which operations are possible) When this is done, the entity can be managed by web service, by program, by user interface)
REPRESENTATIONS
• Device associated
• Class used
• Available behaviors
PROPERTIES
ORGANISATION (SECTIONS, BLOCKS)
NAVIGATION (FILTERS, LINKS)
Defines the UI (how and where the fields appear on the page, how can we navigate from a page to another)

Inbound REST Web Service calls

There are three basic components of an Inbound REST Web Service definition:

1. Representation – this can be seen as the frontend of the REST call – it defines what CRUD Operations are available to External Sites invoking the REST Web Service.

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

2. Class – this represents the Data Model – i.e. what data is “behind” a Representations and the associated REST Calls.

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

3. Table(s) – these are the Data entities associated with the REST Calls.

   Tables are defined in Development > Data and Parameters > Tables > Tables (GESATB).

Inbound REST Web Service calls

Representation

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

Online Help for Representations:

https://online-help.sageerpx3.com/erp/12/staticpost/representation-management/?highlight=prototype

Note that a Class can be cited in more than one Representation – perhaps the Administrator only wishes certain columns of a Class to be visible to the "outside world" via a URL, or that a specific Representation must be used to perform Updates to the underlying Class's data.

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

The following Facets are available for each Representation:

1) Detail
2) Edit
3) Query
4) Lookup
5) Summary

Using these Facets, all the CRUD operations can be fulfilled:

CRUD Operation	RESTful Operation	Representation Facet	Description
Create	POST	Edit	Create a new instance of a Class
Read	GET	Query/Detail/Lookup/Summary	Retrieve data according to the criteria
Update	PUT	Edit	Update an existing instance of a Class
Delete	DELETE	Edit	Delete an instance of a Class

Inbound REST Web Service calls

Representations and Classes

As mentioned before, it is possible to have multiple Representations referencing a single Class.

You might envisage a scenario where an external Web Site needs access to certain data for a particular operation in a process, and needs other data for another operation – again, this is defined by the External Site, rather than X3.

This is covered in a bit more detail in the Examples section later.

Representatio Facets and CRUD Operations

CRUD Operations

As mentioned before, the following Facets are available for each Representation to fulfill the CRUD Operations:

1) Detail
2) Edit
3) Query
4) Lookup
5) Summary

Note : You should use Edit rather than Delete Facet – DELETE REST Operation/Method to say Delete rather than PUT which is used with Update.

https://www.sagecity.com/us/sage_x3/f/general-discussion/191739/rest-web-services-post-a-record-problem

Some more detail is available in the Spring 2022 Tips and Tricks session.

Inbound REST Web Service calls

$details Facet

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.

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

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

The $details URL is in the form

<X3-URL>/api1/x3/erp/solution_folder/CLASS(‘key-value’)?REPRESENTATION.$details

For example:

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

Inbound REST Web Service calls

$edit Facet

This allows an External Site to Create, Delete or Update an instance of the data behind the Representation depending on the RESTful Operation used (PUT, DELETE or POST).

For a Create or Update operation, the Payload provided with the Call will contain the data.

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

<X3-URL>/api1/x3/erp/solution_folder/$prototypes('representation.$edit')

The $edit URL is in the form

<X3-URL>/api1/x3/erp/solution_folder/CLASS('key-value')?REPRESENTATION.$edit

For example:

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

Inbound REST Web Service calls

$query Facet

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.

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

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

The $details URL is in the form

<X3-URL>/api1/x3/erp/solution_folder/CLASS?REPRESENTATION.$query&count-clause&where-clause

For example:

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

Inbound REST Web Service calls

$lookup Facet

The results from such a request is a list of instances of a Class – the fields/properties exposed are determined by the Yes/No settings in the Properties tab within Representations. The list can be refined using the “&key” parameter.

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

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

The $details URL is in the form

<X3-URL>/api1/x3/erp/solution_folder/CLASS?REPRESENTATION.$lookup&where-clause

For example:

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

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

Inbound REST Web Service calls

$summary Facet

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

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

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

The $details URL is in the form

<X3-URL>/api1/x3/erp/solution_folder/CLASS(key-value)?REPRESENTATION.$summary

For example:

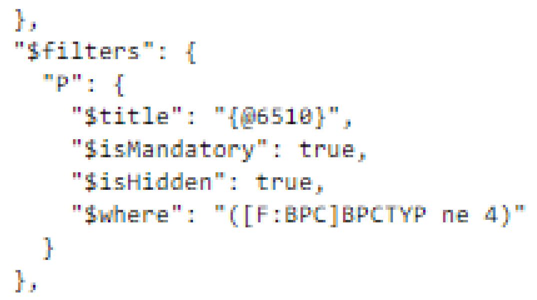
http://x3erpv12sqlvm:8124/api1/x3/erp/X3ERPV12_SEED/BPCUSTOMER(“GB011”)?representation=BPCUSTOMER.$summary

Note that BPCUSTOMER doesn’t have summary facet enabled by default, but this is the sort of URL required.

Inbound REST Web Service calls

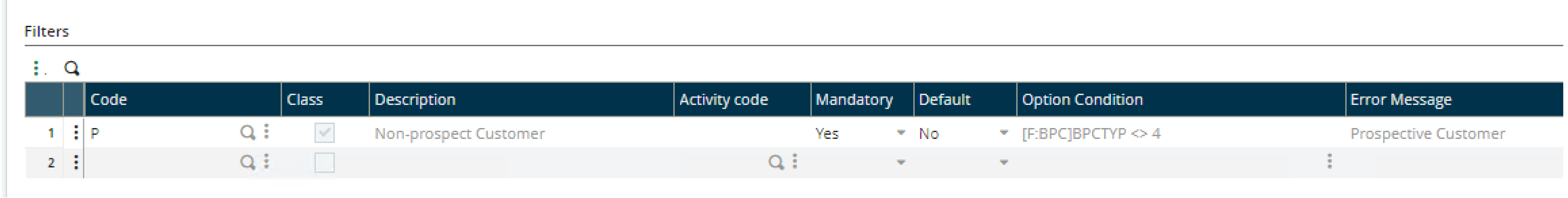
General observation

If you look at the prototypes for these facets, you may see a $filters section – for example, for BPCUSTOMER.$query:



```
},
"$filters": {
  "P": {
    "$title": "{@6510}",
    "$isMandatory": true,
    "$isHidden": true,
    "$where": "([F:BPC]BPCTYP ne 4)"
  }
},
```

This relates to the Filters section of the Representation. In this way, the data available to External Sites can be restricted on a Representation level.



Filters

	Code	Class	Description	Activity code	Mandatory	Default	Option Condition	Error Message
1	P	✓	Non-prospect Customer		Yes	No	[F:BPC]BPCTYP <> 4	Prospective Customer
2								

Outbound REST Web Services

Outbound REST Calls

Sage X3 provides several 4GL Functions to enable calls to External REST interface APIs.

When calling External Sites, you need to set up details of the Site in the

Administration > Administration > Web Services > Rest web services option.

See Online Help https://online-help.sageerpx3.com/erp/12/staticpost/outgoing-rest-web-services/

The Name in this option corresponds to the Name in the EXEC_ Functions described below

You may think Sage X3 doesn't come with any REST APIs built into it, but you could be surprised! For example, the Portuguese EFAT/SAFE functionality calls an External Site.

EFAT/SAFE provides a good example of building a REST interface with the use of Parameters to make it more configurable and flexible – the Parameters are covered in the Online Help.

EXEC_REST_WS/EXEC_REST_WSCLB

The most-used Function, perhaps, is EXEC_REST_WS.

Development added EXEC_REST_WSCLB to make it possible to use OAuth2 Tokens – EXEC_REST_WS has a limitation of 255 Characters for Parameters and Headers, and OAuth2 Tokens exceed this limit. In order to keep compatibility for any existing Code using EXEC_REST_WS, they created this new Function.

These Functions in turn call EXEC_JS and this calls EXEC_HTTP.

Online Help: https://online-help.sageerpx3.com/erp/12/staticpost/api-asyrrestcli/?highlight=EXEC_REST_WS

Unfortunately, the Online Help hasn't been updated to reflect the introduction of EXEC_REST_WSCLB.

The differences are highlighted below.

The format of clob files for these two new parameters must be defined like this : {"code1":val1,"code2":val2,"code3":val3...}.

EXEC_REST_WS	EXEC_REST_WSCLB
Value Char WEBSERVNAME() Value Clbfile HTTPMETHOD() Value Char SUBURL() Value Char PARAM_COD() Value Char PARAM_VAL() Value Char HEAD_COD() Value Char HEAD_VAL() Value Clbfile DATA() Value Integer FUTURE Value Char RETURNS Variable Clbfile RESHEAD() Variable Clbfile RESBODY()	Value Char WEBSERVNAME() Value Clbfile HTTPMETHOD() Value Char SUBURL() Value Clbfile PARAMS Value Clbfile HEADERS() Value Clbfile DATA() Value Integer FUTURE Value Char RETURNS Variable Clbfile RESHEAD() Variable Clbfile RESBODY()

```
RETVAL=func ASYRRESTCLI.EXEC_REST_WS(WEBSERNAME, HTTPMETHOD, SUBURL,
                 PARAM_COD,PARAM_VAL,HEAD_COD,HEAD_VAL,"{}", 0, "", RESHEAD, RESBODY)

RETVAL=func ASYRRESTCLI.EXEC_REST_WSCLB(WEBSERNAME, HTTPMETHOD, SUBURL,
                 PARAMS,HEADERS,"{}", 0, "", RESHEAD, RESBODY)
```

EXEC_JS

X3 provides a mechanism for adding new JavaScript bundles and then allowing 4GL Code to call the bundles.

Outlines of a couple of examples are provided in the Online Help:

Crypto Bundle

https://online-help.sageerpx3.com/erp/12/wp-static-content/static-pages/en_US/v7dev/integration-guide_4gl-crypto-bundle.html

Currency Converter Bundle

https://online-help.sageerpx3.com/erp/12/wp-static-content/static-pages/en_US/v7dev/integration-guide_4gl-crypto-bundle.html

Note the change for Node.js 18 in 2023R1 – node-fibres no-longer supported.

```
Funprog EXEC_JS(MODULE, FUNCTION_NAME, MODE, ARGUMENTS, ENCODINGS, CALLB, RETURNS,
RETURNS_ENC, RESHEAD, RESBODY)
Value Char MODULE()            : # The name of the published node.js javascript module
Value Char FUNCTION_NAME()     : # The name of the javascript function to be called
Value Char MODE()              : # A mode to determine if the call is synchronous or asynchronous
                                 # ('sync' or 'wait')
Value Clbfile ARGUMENTS()      : # Multiple arguments delimited by ',' ; It can be JSON format or
                                 # every types.
Value Char ENCODINGS()         : # Specify if arguments must be base64 encoded or not.
                                 #A list of '0' or '1' separated by ','.
Integer STATUSCODE             : # The status code returned by the javascript runner module.
Value Integer CALLB            : # Specify the location of callback function. This is needed only for
                                 # 'wait' mode.
Value Char RETURNS()           : # Used to filter JSON Objects if you don't want all object properties
Value Char RETURNS_ENC         : # Specify the needs to encode the response to base64 or not. ‘0’ or ’1’
Variable Clbfile RESHEAD       : # The http response header.
Variable Clbfile RESBODY       : # The Response body.

Integer STATUSCODE             : # The status code returned by the javascript runner module.
```

Note that the bundles are located under the Syracuse\bin\node_modules\bundles directory. It's best to put Custom bundles grouped in a directory under this folder to separate them from Standard ones.

As mentioned previously, the Portuguese functionality to interact with the EFAT Site uses RESTful calls which are written in PORLEGEFATLIB's SEND_TO_EFAT. The JS Bundle might be located under

D:\Sage\SafeX3\SYRA\Syracuse\bin\node_modules\bundles\bundle-sage-pt-efat

Custom bundles ext-1 and ext-2 could be stored under

D:\Sage\SafeX3\SYRA\Syracuse\bin\node_modules\bundles\Extras\ext-1

and

D:\Sage\SafeX3\SYRA\Syracuse\bin\node_modules\bundles\Extras\ext-2

Standard Portuguese REST EXEC_JS Calls

Regarding the actual call to update EFAT, the program uses EXEC_JS, the SEND_TO_EFAT function executes a “POST” Call

In order to retrieve information from the EFAT Site, PORLEGETEFATLIB uses REST_GET_EFAT_CALL function which does a “GET” call to EXEC_JS as well.

The calls look like

```
 EFAT_STATUSCODE   = func ASYRWEBSER.EXEC_JS(EFAT_MODULE, EFAT_FONCTION,
EFAT_MODE, EFAT_ARGUMENTS, '', 0, EFAT_RETURNS, '0', EFAT_RESHEAD, EFAT_RESBODY)
```

For different Calls, the variables have different values as in the next page.

When updating the EFAT Site, it is a POST Call:

```
EFAT_MODULE = 'bundle-sage-pt-efat/efatCommunication’
EFAT_FONCTION = 'send’
EFAT_MODE = 'wait’
EFAT_ARGUMENTS = '"Sage-EFAT", "POST", "Invoice/'+EFAT_URLTAXID+'/Invoice",{},{},' + EFAT_CTX + ',’
                  & + EFAT_DATA + ', "'+EFAT_TYPE+'",' + EFAT_OPT
```

Sage-EFAT is the REST web service

EFAT_CTX = EFAT Submission information
EFAT_DATA = this is the data in JSON format – comma-separated “name” : “value” pairs
EFAT_TYPE = “invoice”
EFAT_OPT = Options for Debug Mode, Callback Expected, and Callback URL

Similarly, there is another EXEC_JS Call to retrieve

```
EFAT_TYPE = "invoice_query“
EFAT_ARGUMENTS = '"Sage-EFAT", "GET", "Invoice/'+EFAT_URLTAXID+'",' + EFAT_PAR + ',{},' + EFAT_CTX
            & + ',{}, "'+EFAT_TYPE+'",' + EFAT_OPT
```

EXEC_HTTP

This call executes a URL from within X3 4GL Code

```
Funprog EXEC_HTTP(HEADERCOD, HEADERVAL, DATA, RESHEAD, RESBODY)

Value Char HEADERCOD()(1..)   : # An array that is contain header's keys.
Value Char HEADERVAL()(1..)   : # An array that is contain header's value.
Value Clbfile DATA()          : # Http body that you want to send (for methods 'POST' and 'PUT’
                                # only).
Variable Clbfile RESHEAD()    : # The Http response header.
Variable Clbfile RESBODY()    : # The Http Response body.



Integer STATUSCODE            : # The status code returned by the javascript runner module.
```

Online Help: https://online-help.sageerpx3.com/erp/12/wp-static-content/static-pages/en_US/v7dev/api-guide_api-asyrwebser.html

RESTful Authentication

JSON Web Tokens

In X3, the GET_TOKEN 4GL Function provides support for External Sites which require JWT (JSON Web Tokens) authentication rather than basic User/Password or OAUTH2 authentication.

If you need to use JWT, it might be worth looking at sites such as http://jwt.io if you need to test Tokens.

In order to get a JWT value, you need to set up a Connected application in

Administration > Administration > Settings Authentication > Connected application

The GET_TOKEN call uses the Connected Application's ClientID as the lookup.

RESTful Authentication

JSON Web Tokens

Some Web Sites use JWT Tokens for Authentication, and GET_TOKEN allows X3 to obtain and then use such Tokens.

```
GET_TOKEN (CONAPP, PARAMS, RESHEAD, RESBODY)

Value     Char  CONAPP()      # The Client ID from Connected Application option
Value     Clbfile  PARAMS()   # A YAML-format string contaning param-name:param-value pairs
Variable  Clbfile  RESHEAD    # Response Header
Variable  Clbfile  RESBODY    # Response Body – the JWT returned for the Connected Application
```

The Token retrieved in the RESBODY would then be used in the EXEC_REST_WS call with "Bearer" Authentication.

Just out of interest, you can see that JWT is used by Web Scheduling and GraphQL by the very fact that their installation process includes setting up a Connected Application.

RESTful Authentication

OAUTH2

Another method of Authentication is Oauth2 where a Token is obtained from an Oauth2 Site and then used as the Bearer token in a subsequent RESTful call.

As noted earlier, the outbound RESTful call cannot use EXEC_REST_WS due to the limitation on the size of the HCOD parameter array – it must use EXEC_REST_WSCLB instead.

A high-level view of a function called GET_EXTERN_DATA to extract a string value from an External Site with an Authentication Token for a specific User Account and Password might look like the code below:

RESTful Authentication

OAUTH2

```
Funprog GET_EXTERN_DATA (USR, PWD, DATAID, RESULTS)
Value Char USR()        # User to get authenticated
Value Char PWD()        # User's Password to authentication site
Value Char DATAID()     # The key to the data to be retrieved from the External Site
Variable Char RESULTS()  # Data retrieved from External Site

Local Char PCOD(100)(1..10),PVAL(250)(1..10),HCOD(100)(1..10),HVAL(255)(1..10)
Local Clbfile RESHEAD(0), RESBODY(0)
Local Clbfile PARAMS(0), HEADERS(0)

#First call to a web service for Token:
HCOD(1) = "Content-Type" : HVAL(1) = '"application/json"'
DATA = '{"username":"'+USR+'","password":"'+PWD+'"}'
RETVAL=func ASYRRESTCLI.EXEC_REST_WS("OAUTH2SITE","POST","",PCOD,PVAL,HCOD,HVAL,DATA,
                                    0,"",RESHEAD, RESBODY)
```

RESTful Authentication

```
#Process the RESBODY to obtain the TOKEN from the Authorisation call
Local Clbfile TOKEN(0)
If RETVAL=200
  TOKEN=left$(right$(RESBODY,11),len(RESBODY)-12)
  DATA = '{}'
  HEADERS = '{"Authorization" : Bearer "'+TOKEN+'"'}'
  PARAMS = '{"DATAID" : "' + DATAID + '" }'  # What's the specific data?

  RETVAL=func ASYRRESTCLI.EXEC_REST_WSCLB("INFOSITE","GET","",PARAMS, HEADERS,DATA, 0,"",RESHEAD,
                                                                                    RESBODY)

  # Extract Data returned from the External Site in RESBODY
  If RETVAL=200
    # return data in RESULTS parameters ...
  Endif
Endif

End RETVAL
```

Simple Examples

Example 1: One-way traffic

Exchange Rate System

A common requirement is that Currency Exchange Rates in X3 need to be updated with the latest values from an External Site on a regular (daily) basis.

How this is done would depend on the External Site – potentially:

1) The External Site may push out the values, so X3 would have in-bound REST Calls;

or

2) The External Site is purely passive, and X3 needs to invoke out-bound REST Calls to pull the information back

The latter of these is probably the most likely.

Of course, this could be enhanced – see considerations in the “Things to think about” section later.

Example 1: One-way traffic

Exchange Rate System Scenario 1

The first scenario is where an External Site pushes the Exchange Rates, so X3 in-bound REST Calls are made.

As mentioned before, in respect of in-bound RESTful Calls, we'd receive the Rates into a Custom Table using a Custom Representation and Class, and the data would then be propagated to the standard Exchange Rates Table, TABCHANGE, using an X3 Custom program.

The External Site always sends updates at 2am UTC, so it is safe to assume that all necessary updates to the Custom Table are done by 3am UTC. This means, we can write a program which is run by Batch Server to update TABCHANGE.

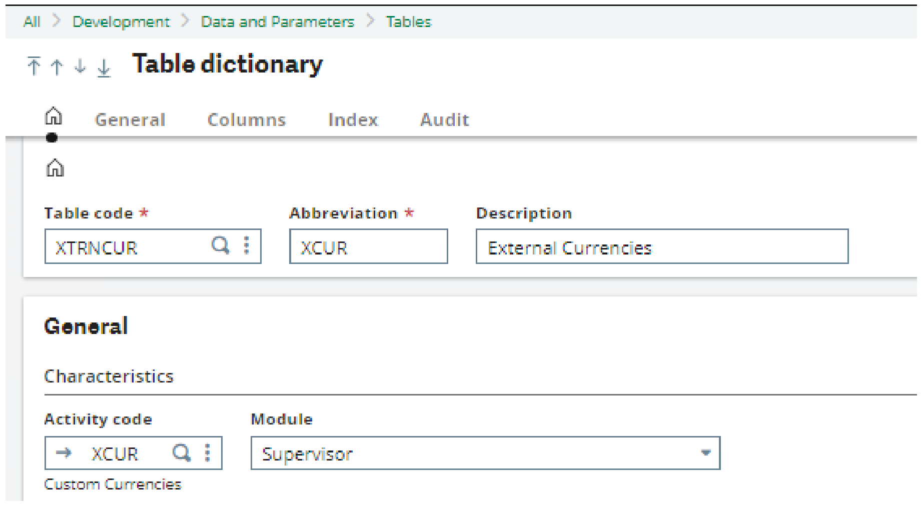
For us, the interesting bit is related to the in-bound REST Calls – we need to create a Custom Table, Class and Representation.

The details of this were demonstrated earlier, but just collected below as part of this Example scenario.

Example 1: One-way traffic

Exchange Rate System Scenario 1

Custom Table XTRNCUR:



All > Development > Data and Parameters > Tables

Table dictionary

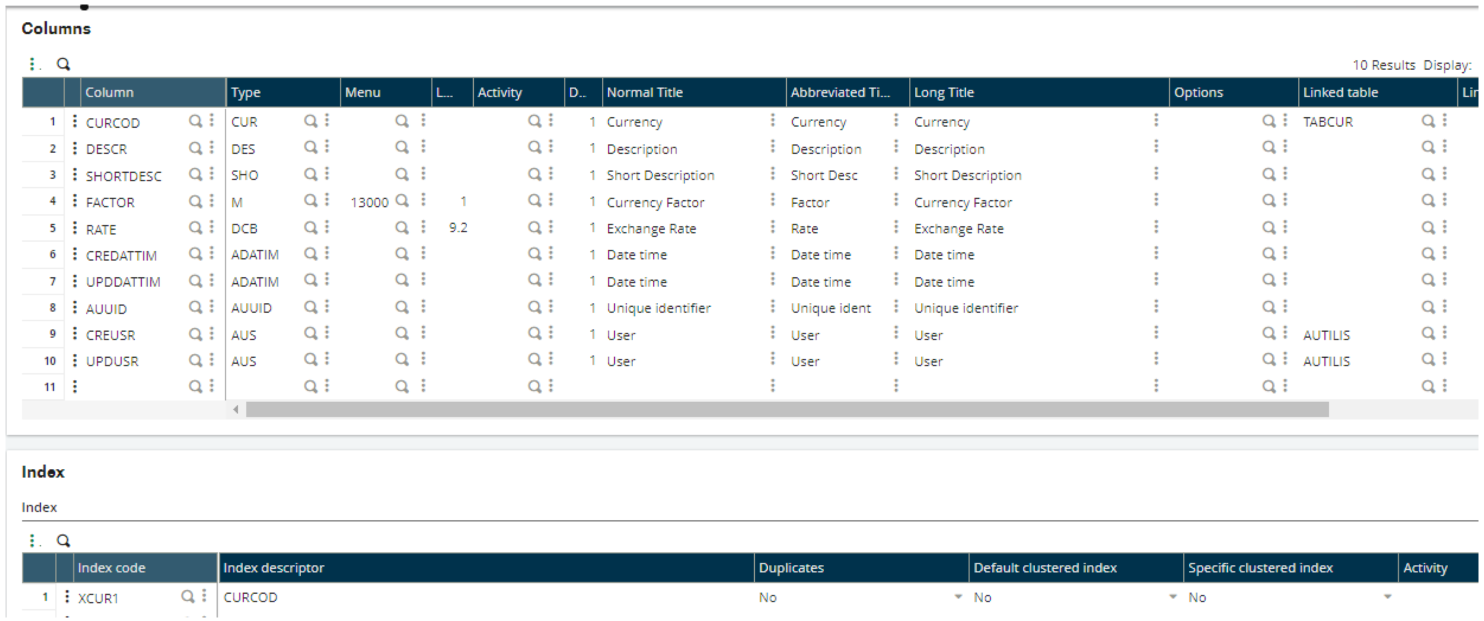
General Columns Index Audit

Table code *	Abbreviation *	Description
XTRNCUR	XCUR	External Currencies

General

Characteristics

Activity code	Module
XCUR (Custom Currencies)	Supervisor



Columns

10 Results Display:

	Column	Type	Menu	L...	Activity	D..	Normal Title	Abbreviated Ti...	Long Title	Options	Linked table
1	CURCOD	CUR				1	Currency	Currency	Currency		TABCUR
2	DESCR	DES				1	Description	Description	Description		
3	SHORTDESC	SHO				1	Short Description	Short Desc	Short Description		
4	FACTOR	M	13000	1		1	Currency Factor	Factor	Currency Factor		
5	RATE	DCB		9.2		1	Exchange Rate	Rate	Exchange Rate		
6	CREDATTIM	ADATIM				1	Date time	Date time	Date time		
7	UPDDATTIM	ADATIM				1	Date time	Date time	Date time		
8	AUUID	AUUID				1	Unique identifier	Unique ident	Unique identifier		
9	CREUSR	AUS				1	User	User	User		AUTILIS
10	UPDUSR	AUS				1	User	User	User		AUTILIS
11											

Index

Index

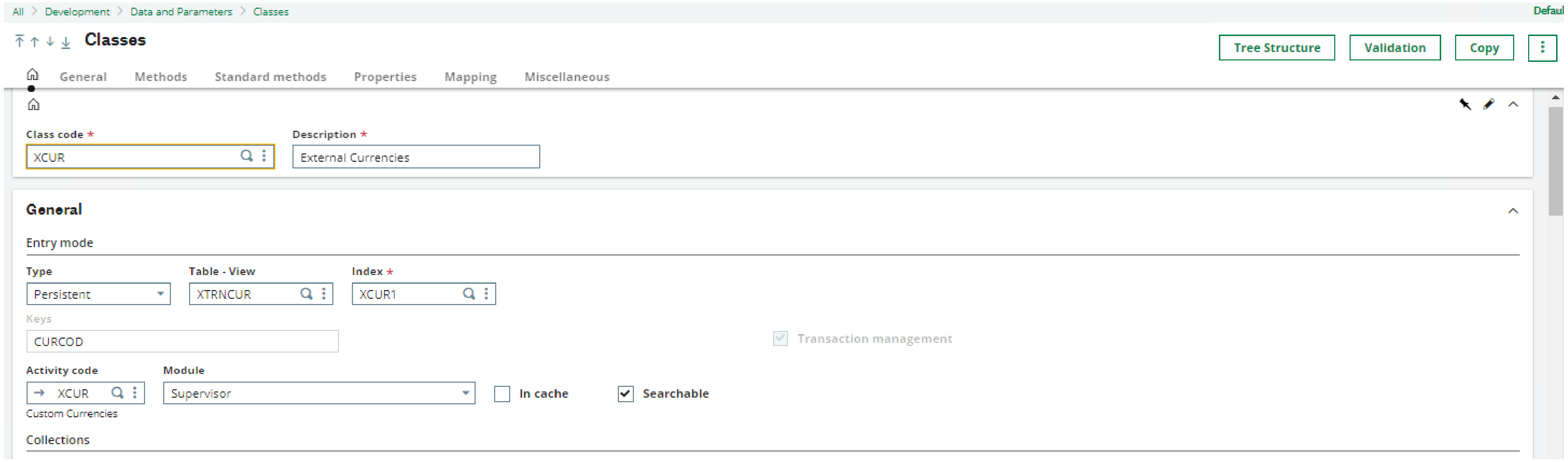
	Index code	Index descriptor	Duplicates	Default clustered index	Specific clustered index	Activity
1	XCUR1	CURCOD	No	No	No	

Note the use of Column Data Type CUR for the CURCOD – this ensures Data Integrity as the Data Type implies a lookup to TABCUR for Currency Codes.

Example 1: One-way traffic

Exchange Rate System Scenario 1

Custom Class XCUR:



All > Development > Data and Parameters > Classes

Classes

Tree Structure | Validation | Copy

General | Methods | Standard methods | Properties | Mapping | Miscellaneous

Class code * XCUR

Description * External Currencies

General

Entry mode

Type	Table - View	Index *
Persistent	XTRNCUR	XCUR1

Keys: CURCOD

☑ Transaction management

Activity code: XCUR (Custom Currencies)

Module: Supervisor

☐ In cache

☑ Searchable

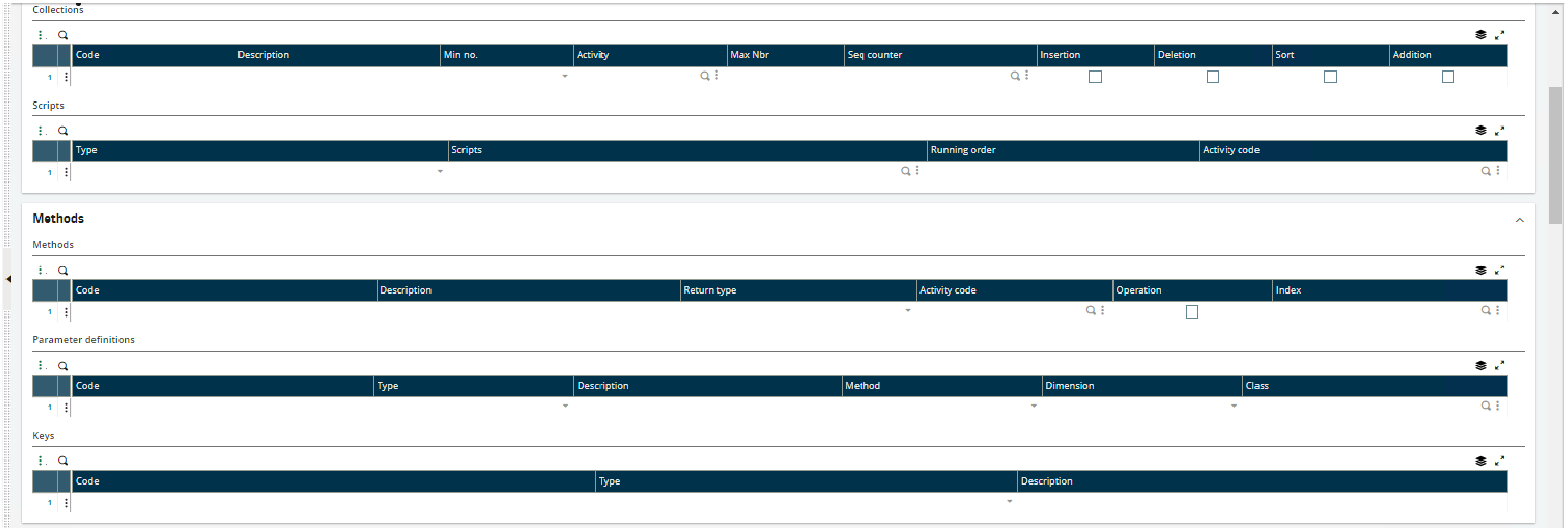
Collections

Note that the Class name doesn't have to be the same as the underlying Table name.

Example 1: One-way traffic

Exchange Rate System Scenario 1

Custom Class XCUR:



Collections

	Code	Description	Min no.	Activity	Max Nbr	Seq counter	Insertion	Deletion	Sort	Addition
1							☐	☐	☐	☐

Scripts

	Type	Scripts	Running order	Activity code
1				

Methods

Methods

	Code	Description	Return type	Activity code	Operation	Index
1					☐	

Parameter definitions

	Code	Type	Description	Method	Dimension	Class
1						

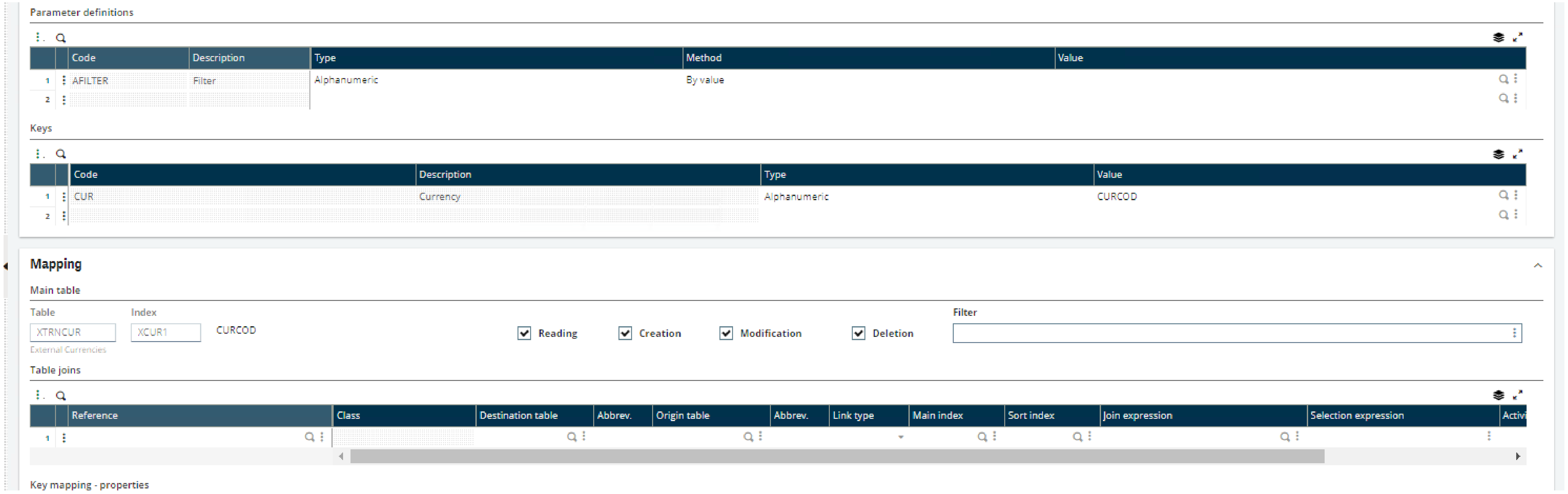
Keys

	Code	Type	Description
1			

Example 1: One-way traffic

Exchange Rate System Scenario 1

Custom Class XCUR:



Parameter definitions

	Code	Description	Type	Method	Value
1	AFILTER	Filter	Alphanumeric	By value	
2					

Keys

	Code	Description	Type	Value
1	CUR	Currency	Alphanumeric	CURCOD
2				

Mapping

Main table

Table	Index						Filter
XTRNCUR External Currencies	XCUR1	CURCOD	✓ Reading	✓ Creation	✓ Modification	✓ Deletion	

Table joins

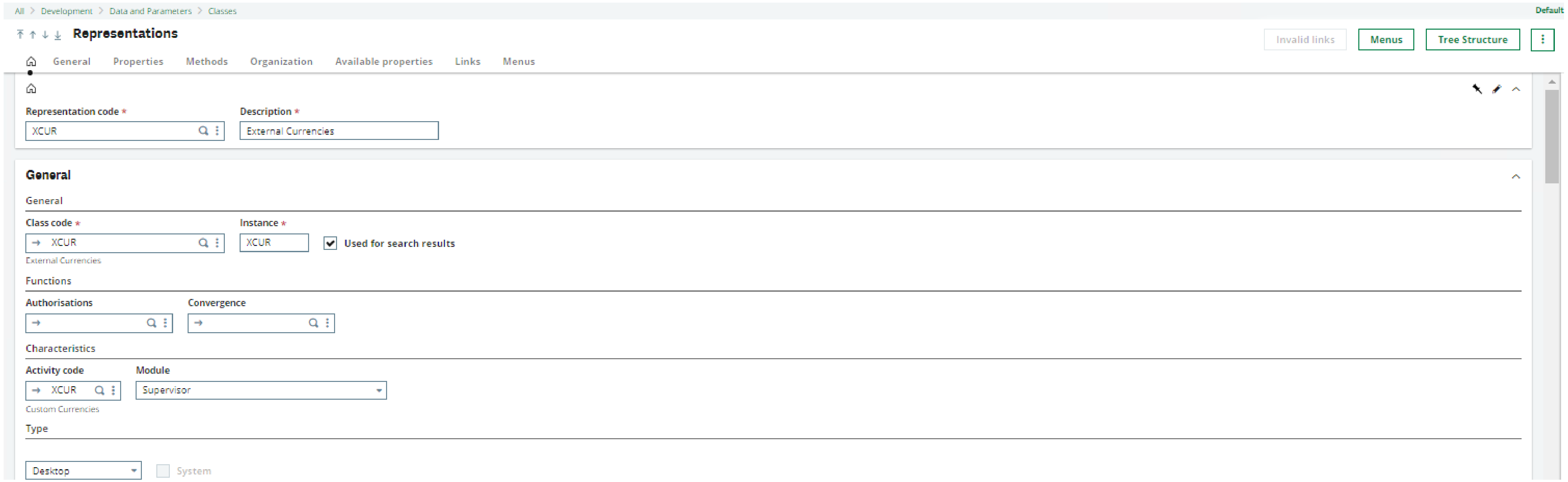
	Reference	Class	Destination table	Abbrev.	Origin table	Abbrev.	Link type	Main index	Sort index	Join expression	Selection expression	Activi
1												

Key mapping - properties

Example 1: One-way traffic

Exchange Rate System Scenario 1

Custom Representation XCUR:



Representation code *: XCUR

Description *: External Currencies

General

General

Class code *: XCUR

External Currencies

Instance *: XCUR

✓ Used for search results

Functions

Authorisations:

Convergence:

Characteristics

Activity code: XCUR

Custom Currencies

Module: Supervisor

Type

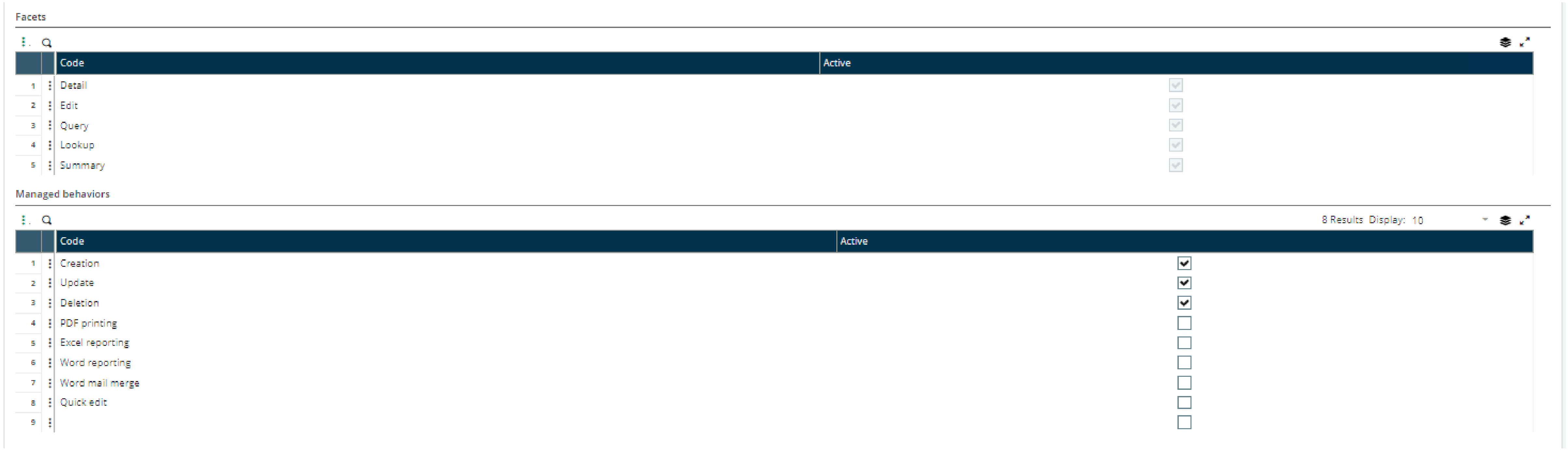
Desktop

System

Example 1: One-way traffic

Exchange Rate System Scenario 1

Custom Representation XCUR:



Facets

	Code	Active
1	Detail	✓
2	Edit	✓
3	Query	✓
4	Lookup	✓
5	Summary	✓

Managed behaviors

8 Results Display: 10

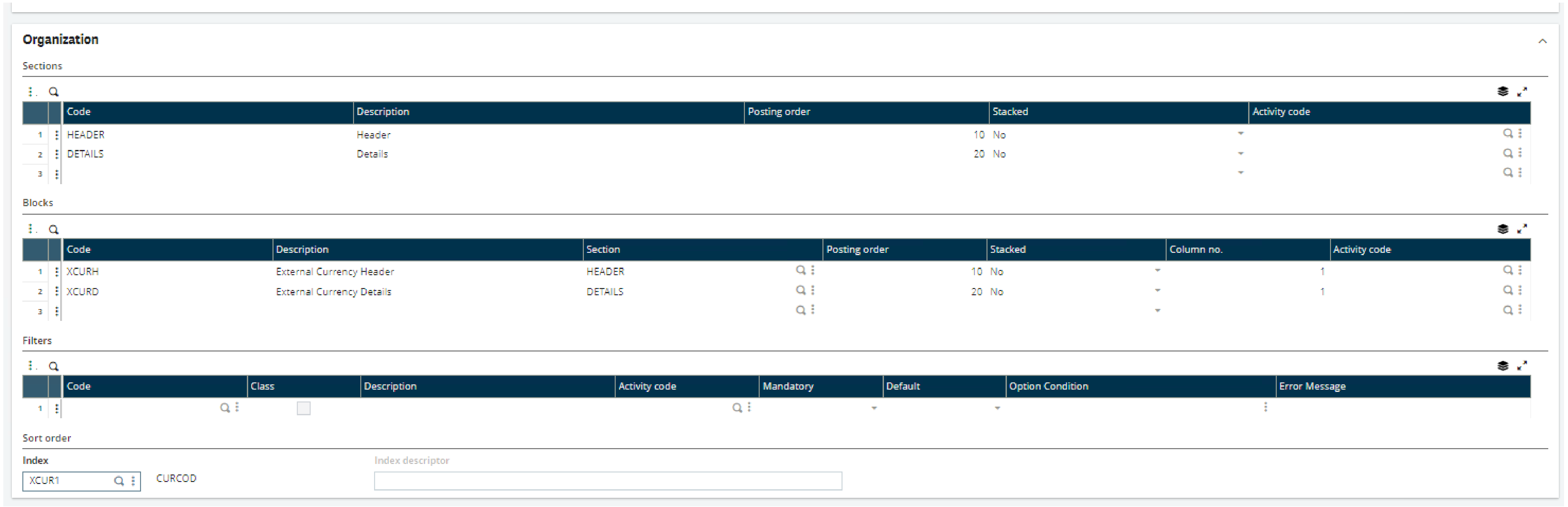
	Code	Active
1	Creation	✓
2	Update	✓
3	Deletion	✓
4	PDF printing	☐
5	Excel reporting	☐
6	Word reporting	☐
7	Word mail merge	☐
8	Quick edit	☐
9		☐

Note that I’ve allowed for Creations and Deletions as well – probably wouldn’t need the latter!

Example 1: One-way traffic

Exchange Rate System Scenario 1

Custom Representation XCUR:



Organization

Sections

	Code	Description	Posting order	Stacked	Activity code
1	HEADER	Header	10	No	
2	DETAILS	Details	20	No	
3					

Blocks

	Code	Description	Section	Posting order	Stacked	Column no.	Activity code
1	XCURH	External Currency Header	HEADER	10	No	1	
2	XCURD	External Currency Details	DETAILS	20	No	1	
3							

Filters

	Code	Class	Description	Activity code	Mandatory	Default	Option Condition	Error Message
1								

Sort order

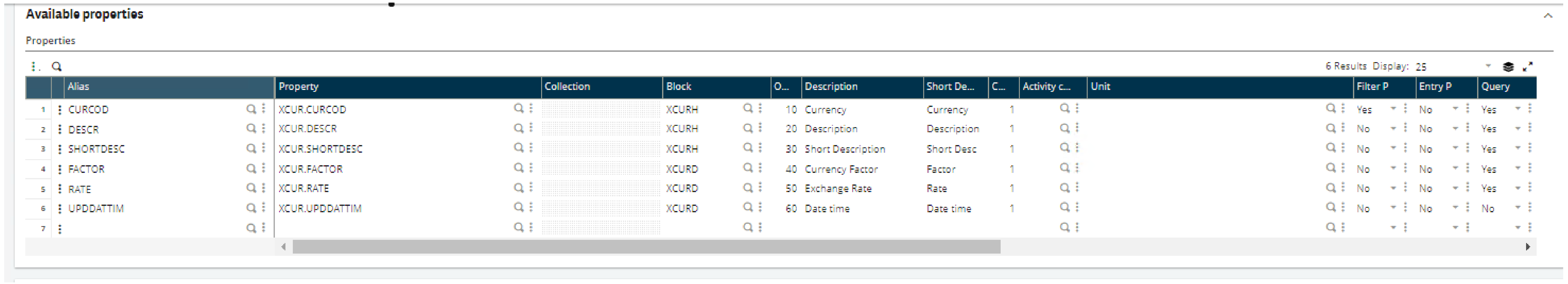
Index: XCUR1 CURCOD

Index descriptor

Example 1: One-way traffic

Exchange Rate System Scenario 1

Custom Representation XCUR:

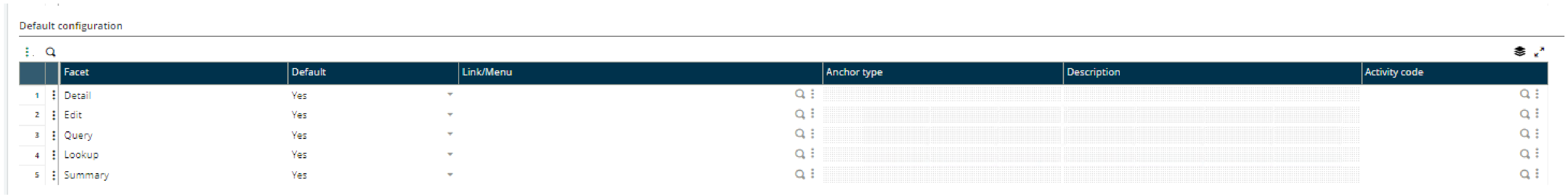


Available properties

Properties

6 Results Display: 25

	Alias	Property	Collection	Block	O...	Description	Short De...	C...	Activity c...	Unit	Filter P	Entry P	Query
1	CURCOD	XCUR.CURCOD		XCURH	10	Currency	Currency	1			Yes	No	Yes
2	DESCR	XCUR.DESCR		XCURH	20	Description	Description	1			No	No	Yes
3	SHORTDESC	XCUR.SHORTDESC		XCURH	30	Short Description	Short Desc	1			No	No	Yes
4	FACTOR	XCUR.FACTOR		XCURD	40	Currency Factor	Factor	1			No	No	Yes
5	RATE	XCUR.RATE		XCURD	50	Exchange Rate	Rate	1			No	No	Yes
6	UPDDATTIM	XCUR.UPDDATTIM		XCURD	60	Date time	Date time	1			No	No	No
7													



Default configuration

	Facet	Default	Link/Menu	Anchor type	Description	Activity code
1	Detail	Yes				
2	Edit	Yes				
3	Query	Yes				
4	Lookup	Yes				
5	Summary	Yes				

Example 1: One-way traffic

Exchange Rate System Scenario 2

The second scenario is where X3 pulls the Exchange Rates from the External Site, so X3 out-bound REST Calls are made.

In this scenario, we'd receive the Rates into a Custom Table via an outbound RESTful call, and the data would then be propagated to the standard Exchange Rates Table, TABCHANGE, using an X3 Custom program. Note that the Table is as described in the scenario above.

The External Site always updates its Rates at 2am UTC, so it is safe to assume that X3 can request all necessary updates to the Custom Table at 3am UTC. This means, we can write a program which is does a RESTful call to the External Site and it can run by Batch Server – the same program used by Scenario 1 can be scheduled to run at 3:30am to update TABCHANGE.

Note that it might be worth adding some error-checking to make sure the RESTful call was made properly…

For us, the interesting bit is related to the out-bound REST Calls – we need to write code to update a Custom Table based on the data from the External Sire.

Example 1: One-way traffic

Exchange Rate System 2

The External Site providing the APIs doesn’t require any Authentication. The site’s details are in the REST Web Services record “CURSITE”.

The data is returned as a JSON text list

```
Local Char PCOD(100)(1..10),PVAL(250)(1..10),HCOD(100)(1..10),HVAL(255)(1..10)
Local Clbfile RESHEAD(0), RESBODY(0)

#First call to a web service for Token:
HCOD(1) = "Content-Type" : HVAL(1) = '"application/json“’

RETVAL=func ASYRRESTCLI.EXEC_REST_WS(“CURSITE",“GET","",PCOD,PVAL,HCOD,HVAL,DATA,
0,"",RESHEAD, RESBODY)

If RETVAL = 200
  # Process  JSON results in RESBODY
Endif
```

Example 2 : two-way traffic

Sales Order Tracking System

This is a simplistic scenario, just to show how an interface to an External Site can require both in-bound and out-bound REST Calls:

1. When a Sales Order is “confirmed” in X3 (i.e. no way back!), a RESTful call is made to the Sales Order Tracking system (SOT) to create an entry for it to operate on.

2. Once the SOT has a record, it updates X3 with its SOT reference.

3. When the Sales Order is picked up by a Courier, X3 is updated with the Courier’s ID and pickup time.

4. Once the Courier has delivered the Sales Order, it updates the Status in X3, along with the delivery time.

Of course, this could be enhanced – see considerations in the “Things to think about” section later.

Example 2 : two-way traffic

Sales Order Tracking System : In or Out ?

The phases of the SOT process require both in-bound and out-bound REST Calls:

1. Outbound RESTful Call to notify SOT System that the Order is ready to ship
2. Inbound RESTful Call to update X3 with the SOT-Reference from the SOT System.
3. Inbound RESTful Call to X3 in order to update X3 with the Courier's ID and pickup time.
4. Inbound RESTful Call to X3 in order to update X3 with the Status in X3, along with the delivery time.

Because we've got a mixture of Inbound and Outbound Calls, we need to create a new Table, Class and Representation and a 4GL program to deal with step (1) of the process.

Again, extra "bells and whistles" could be added as required…

Example 2 : two-way traffic

For Sales Order Tracking, we create a table YSOT with a Key called SOHNUM which is associated with VCRNUM Data Type and SORDER Table – this provides automatic Data Integrity.

The YSOT table also needs to store the Courier ID when one is allocated in SOT and Status as it progresses through the SOT system.

In its simplest form, the Courier ID and Status fields can be Alphanumeric Fields and will rely on SOT to provide valid values – as discussed in the next section, this could change....

Once the YSOT Table has been created, the Class and Representation are set up to provide a RESTful interface for Inbound calls.

As mentioned, a 4GL program will be required to initiate an Outbound RESTful all to the SOT system to register individual Sales Orders.

Example 2 : two-way traffic

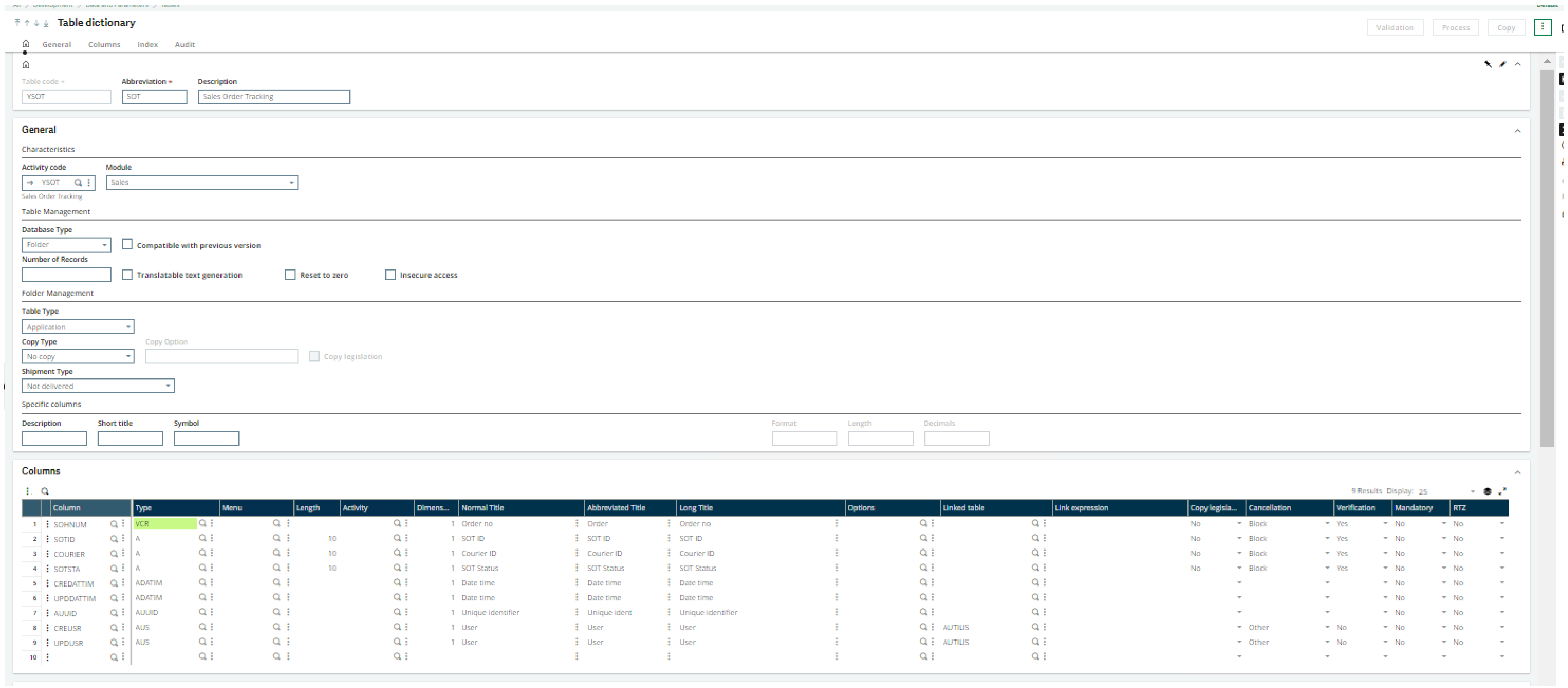


Table dictionary

General | Columns | Index | Audit

Validation | Process | Copy

Table code *: YSOT
Abbreviation *: SOT
Description: Sales Order Tracking

General

Characteristics

Activity code: YSOT (Sales Order Tracking)
Module: Sales

Table Management

Database Type: Folder
☐ Compatible with previous version
Number of Records:
☐ Translatable text generation ☐ Reset to zero ☐ Insecure access

Folder Management

Table Type: Application
Copy Type: No copy
Copy Option:
☐ Copy legislation
Shipment Type: Not delivered

Specific columns

Description: | Short title: | Symbol: | Format: | Length: | Decimals:

Columns

9 Results Display: 25

	Column	Type	Menu	Length	Activity	Dimens...	Normal Title	Abbreviated Title	Long Title	Options	Linked table	Link expression	Copy legisla...	Cancellation	Verification	Mandatory	RTZ
1	SOHNUM	VCR				1	Order no	Order	Order no				No	Block	Yes	No	No
2	SOTID	A		10		1	SOT ID	SOT ID	SOT ID				No	Block	Yes	No	No
3	COURIER	A		10		1	Courier ID	Courier ID	Courier ID				No	Block	Yes	No	No
4	SOTSTA	A		10		1	SOT Status	SOT Status	SOT Status				No	Block	Yes	No	No
5	CREDATTIM	ADATIM				1	Date time	Date time	Date time							No	No
6	UPDDATTIM	ADATIM				1	Date time	Date time	Date time							No	No
7	AUUID	AUUID				1	Unique identifier	Unique ident	Unique identifier							No	No
8	CREUSR	AUS				1	User	User	User		AUTILIS			Other	No	No	No
9	UPDUSR	AUS				1	User	User	User		AUTILIS			Other	No	No	No
10																	

Things to think about

Things to think about

Some guidance on how to approach developing Inbound and Outbound RESTful components

1) Can the requirement best be fulfilled using Inbound or Outbound calls?

   By their very nature, Inbound calls rely on the Class/Representation mechanism to control what happens – this makes it less easy to add “bells and whistles”.

   Inbound calls are initiated by the External Site – is this the appropriate source of calls?

   Inbound calls can only be made by External Sites which are “aware” of our target X3 system – if a Site doesn’t know who to call, or with what parameters, then it’s not going to work very well!

   Outbound calls are made by X3 4GL Code – this means more “bells and whistles” can be incorporated into their design.

   Outbound calls are initiated by X3 itself – be it as an Action, Launching a Menu item, or as a Batch Request.

   Outbound calls are easier in that we can make X3 “aware” of the External Site and its API definitions, and the calls can therefore be written appropriately.

Things to think about

2) Try to isolate the X3 parts of the Web Services:
- If you need to Create, Update or Delete X3 Data, can this be done in two phases via Staging Tables plus 4GL Code?
- If calling External sites, do the RESTful calls need to be done in real time, or can/should they be scheduled?

3) Provide an Audit Trail for Create, Update or Delete operations on X3 Data
   This can be in the shape of Staging Tables and/or Trace Files, or perhaps X3 Auditing.

4) Think about Data Integrity
   Endeavour to build-in a layer of checks that ensure the integrity of X3 Data.

5) Bells and whistles
   Are there extra features which need to be considered during the design phase?

Example : Sales Order Tracking

Can the requirements best be fulfilled using inbound or outbound calls?

Let’s take a look at the Sales Order Tracking example.

Well, this is dependant on what the task is:

When the Sales Order is “committed” in X3, the details required by SOT need to be sent in “real time” – this would be best served by an outbound call as X3 is the initiator of the task – otherwise, the SOT would have to be polling X3 all the time.

When the Courier picks up the Sales Order, then SOT should initiate an inbound X3 Call – again, this should be in “real time”.

When the Courier delivers the Sales Order, then SOT should initiate an inbound X3 Call.

Since a mixture of inbound and outbound calls is indicated, both Classes and Representations and 4GL Code will be involved in the overall design.

Example : Sales Order Tracking

Try to isolate the X3 parts of the Web Services

- Can the interaction with standard X3 Data be minimised – to reduce security risk and the amount of customisation of standard functionality.
- In the case of SOT, the data stored in X3 could be in one or more Custom Tables depending on what “bells and whistles” you may wish to implement. This will avoid impact on Standard Tables and Code.
- In the case of SOT, the minimum data could be held in a single table – say YSOT.

Example : Sales Order Tracking

Provide an Audit Trail

- RESTful Web Service Calls which affect X3 should be Audited. If Custom Tables are implemented to store data involved in the overall interface, then Auditing could be implemented *on* those Tables, or the Tables could be used to store the Audit details.
- You could add an extra table for Audit information by having one record per update – say YSOTAUD, or via AUDI.

Example : Sales Order Tracking

Thinking about Data Integrity

Applications should always ensure that Data is consistent – don't store data which doesn't make sense!

There are a couple of pieces of the data involved in the interface which refer to data which needs to be validated:

1) X3 Sales Order Number
2) SOT Courier ID

X3 should be responsible for validating these pieces of data.

Out-of-the-box, there is no "SOT Courier ID" in X3, so it may be wise to incorporate a mechanism for maintaining this in X3 itself.

Example : Sales Order Tracking

Thinking about Data Integrity

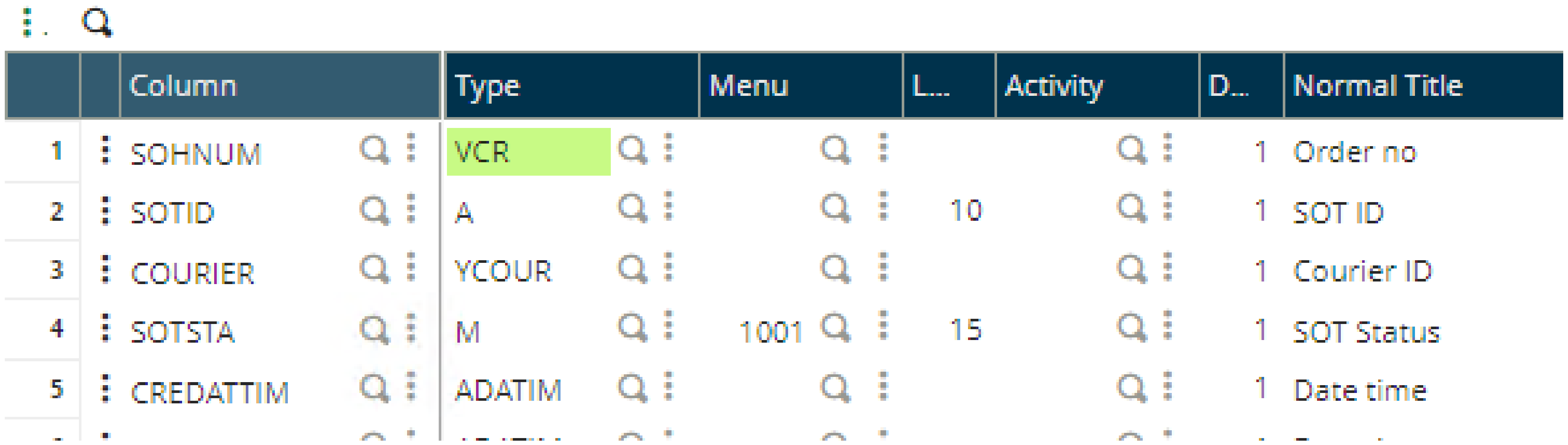
This leads to a possible extra Task: update the list of SOT Courier IDs in X3 – this, in turn, leads to the need for an extra Table, YSOTCOUR, in X3.

The update of the YSOTCOUR Table could be initiated by either X3 or SOT. Personally, I'd suggest that X3 makes scheduled outbound calls to SOT to get the most up-to-date list – perhaps every day at midnight?

As seen with Currencies, the YSOT Table could be defined with implicit data integrity through the use of Data Types.

Example : Sales Order Tracking

Addition to the built-in Data Integrity implied by the setting-up of SOHNUM as a VCRNUM associated with SORDER, we can add the YSOTCOUR Table and YCOUR Data Type. Also, set up a Local Menu for the SOTSTA Column – this is Local Menu 1001 (note the Local Menus Ranges https://online-help.sageerpx3.com/erp/12/staticpost/ranges-used-by-local-menus/?highlight=Local+Menu).



	Column	Type	Menu	L...	Activity	D...	Normal Title
1	SOHNUM	VCR				1	Order no
2	SOTID	A		10		1	SOT ID
3	COURIER	YCOUR				1	Courier ID
4	SOTSTA	M	1001	15		1	SOT Status
5	CREDATTIM	ADATIM				1	Date time

Example : Sales Order Tracking

Bells and Whistles

It might be useful to offer or consider extra features during the design phase.

For example, when designing the Sales Order Tracking add-on, you or the Customer may want to have emails sent when an Order is acknowledged as having been delivered. In order to do this, you could set up a Workflow which is triggered by the update of the Status in YSOT Table.

Setting up a Workflow would also open up the possibility of triggering Actions based on the change of YSOT…

Additional Representations drawing together the Sales Orders and SOT information in YSOT and YSOTCOUR might be nice?

Recap and Conclusions

What are REST Web Services good for?

Are you in or out?

Can you get more?

Don't forget

What are REST Web Services good for?

Web Services are good for adding functionality to X3 – either by allowing External Sites to interrogate or update X3's Data, or requesting data or functionality from External Sites.

Are you in or out?

REST Web Services can be initiated by X3 itself or by External Sites – it depends on which system is "driving" the interaction.

Can you get more?

As we've seen, inbound calls can provide a lot of features with little work. The Representations developed for the interface can also be used within X3 – for example, a Representation could be developed to show the progress of Sales Orders in the SOT System.

Outbound calls are written in X3's 4GL, so more functionality can be included by enhancing the Code. There's nothing stopping you from writing Representations, Queries or even Reports on the data within X3.

Don’t forget

Create an Activity Code to mark and safe-guard any Customisations (new columns, tables, classes, representations, sources/adx)

Think about Data Integrity

Think about Debugs and Auditing

Appendix

More information

EXEC_REST_WS / EXEC_REST_WSCLB

https://online-help.sageerpx3.com/erp/12/staticpost/api-asyrrestcli/?highlight=EXEC_REST_WS

https://support.na.sage.com/selfservice/viewdocument.do?noCount=true&externalId=113579

https://www.sagecity.com/us/sage_x3/b/sageerp_x3_product_support_blog/posts/how-to-call-an-external-rest-web-services-in-classic-functions

https://www.sagecity.com/us/sage_x3/b/sageerp_x3_product_support_blog/posts/how-to-call-an-external-rest-web-services-in-classic-function-in-future-asynchronous-mode

EXEC_JS

https://online-help.sageerpx3.com/erp/12/staticpost/javascript-extensibility-example-currency-converter-bundle/

EXEC_HTTP

https://www.sagecity.com/fr/sage-x3/f/technique/154214/http-get-request

More information

Sage University

Using REST Web Services for Query

X3 - Sage X3 - Web Services and Integration

X3 - Sage X3 - Using SOAP Web Services for Import and Export

X3 - Sage X3 - Expert Workshop: Web Services Use Case Review & Practice

X3 - Sage X3 - Expert Workshop: Why Classes and Representations are Still Interesting

Thank you!