Request for Quotation Process

Contents

[*2* *Overview* 3](#_Toc57295974)

[3 The RFQ 4](#_Toc57295975)

[4 Vendors Selected to bid 5](#_Toc57295976)

[5 ME41 Create a RFQ group 6](#_Toc57295977)

[6 ME41 - On Save 9](#_Toc57295978)

[7 ZERFQ Tender Cockpit 10](#_Toc57295979)

[8 Add Documents for vendors to reference 11](#_Toc57295980)

[9 Release the RFQ 11](#_Toc57295981)

[10 Monitor activities on Tender 11](#_Toc57295982)

[11 Vendor will receive SMS and Email with a Link 12](#_Toc57295983)

[12 E-sourcing screen 13](#_Toc57295984)

[13 Details Page 13](#_Toc57295985)

[14 Loading and Downloading Documents 14](#_Toc57295986)

[15 Optional Login or registration 14](#_Toc57295987)

[16 Reset Password with OTP 15](#_Toc57295988)

[17 Close and Evaluate 15](#_Toc57295989)

[18 On Sign on 15](#_Toc57295990)

[19 Password Reset 15](#_Toc57295991)

# Overview

Bidvest Facilities Management strives to get the best value when procuring outside services or buying products. To do this, they encourage Vendors to Provide quotes for the provision of goods or services. This way bidders are encouraged to provide the best prices and delivery times.

The basis of the process is simple:

* Create a RFQ – Request For Quotation
  + Clearly state what is required in terms of Quantity and Deliverables
  + Establish Cut off Dates
* Invite Vendors to offer prices and terms
* Provide a mechanism for Vendors to submit their offers
* Select the lowest price offer for products or the most suitable offer for a service.
* Create a Purchase Order to the selected Vendor and inform the others that they were not successful

The Solution we are developing covers the first 3 points completely with some support for the others. It will allow an RFQ to be created where only Vendors on the approved supplier list can be selected as potential bidders, while having a website where the Vendor can submit their bid.

# The RFQ

In order to comply with standards, the RFQ Process requires the following information to be maintained for each and every Quote:

1. What is being Quoted on:
   1. Standard Process:
      1. A Purchase Requisition can be created on SAP and turned into a RFQ.
      2. Additional Documents can be attached to the RFQ.
   2. A Non-Standard Process:
      1. A Reference Object is needed
         1. Work Order
         2. Notification
         3. Project / WBS
         4. Purchase Requisition – just a skeleton
      2. A Textual instruction can be created to explain what is required at a higher level.
      3. Bill of Quantities document can also be drafted and attached as a basis for the quote.
2. Who is being asked to Quote?
   1. Vendors should be on an approved list.
   2. Email Addresses are needed.
3. Release the RFQ to the vendors

# Vendors Selected to bid

All RFQ interactions require a Point of Contact at each selected Vendor or Service Provider.

For the Standard SAP RFQ Process, SAP uses Vendor Address numbers.

Because many Vendors will have multiple points of contact for various reasons, there needs to be a clear definition of who the specific point of contact can be and where this data is maintained.

For the Non-Standard RFQ process, another SAP method can be used.

The SAP Contact person is an ideal place to store the communication data of these persons.

They can be created or maintained via:

1. MAP1 , MAP2 or MAP3 or XK02 for Vendors and Service Providers
2. VAP1, VAP2 or VAP3 or XD02 for Customers.

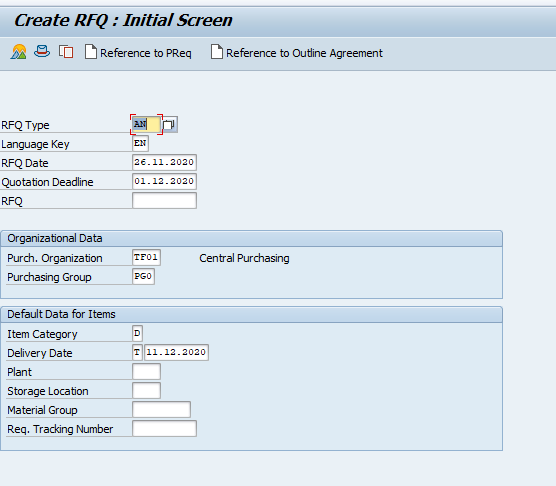
There is no limit to the number of persons that can be linked to a vendor as a Contact Person and the information stored at that level is more than what is required for the RFQ Process.

For the RFQ Process to work, we simply require the following for each contact person **per Vendor**:

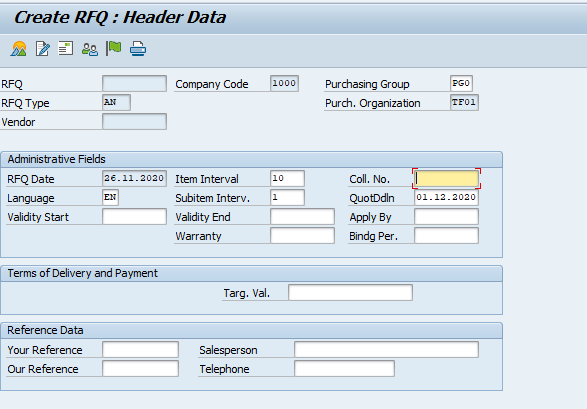
* Name of Person
* Contact Details
  + Email address
  + Mobile number for OTP processes if needed

SAP will allocate an unique identification number to each contact person with the only possible challenge being that where a person is the contact person for more than one vendor, that person will be allocated multiple Contact Person Identification numbers.

# ME41 Create a RFQ group

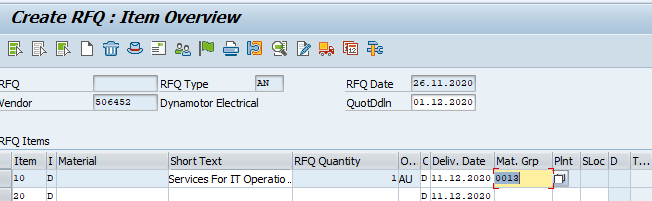


Most Important: Create a Collective Number on First RFQ.

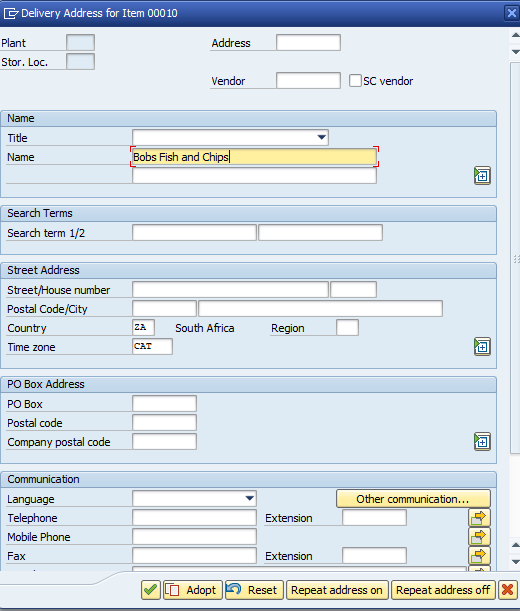


NB

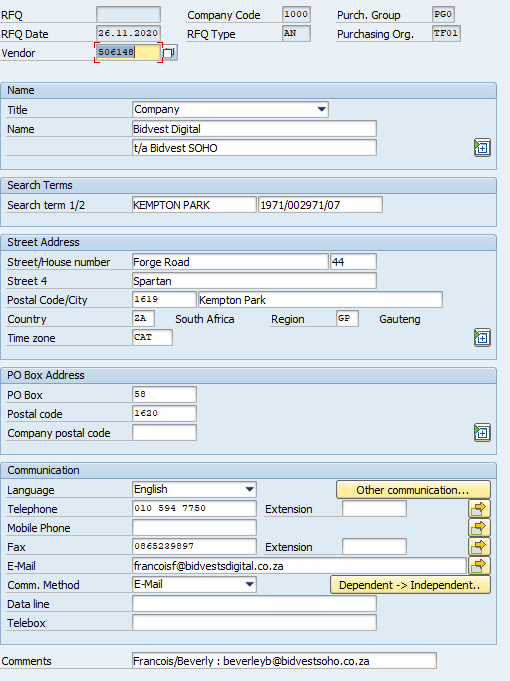
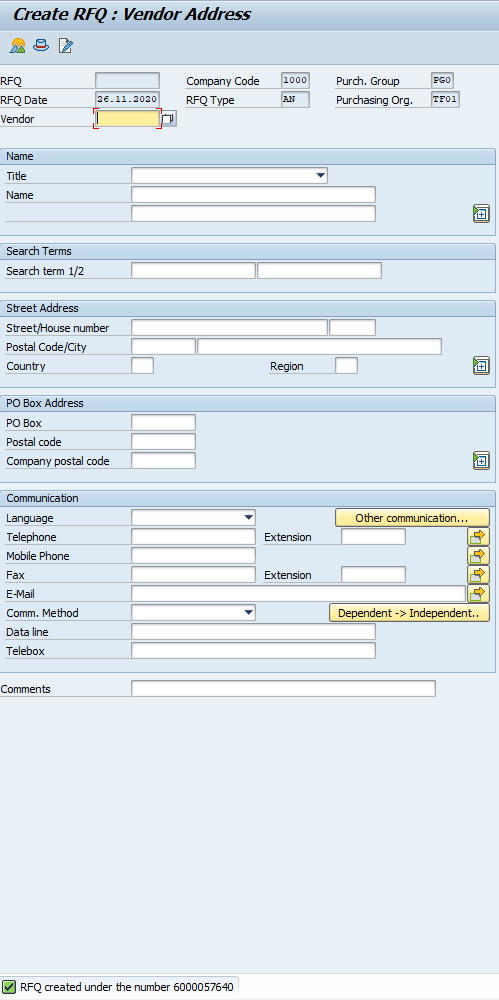
Then Create the Items:



Then the delivery Address must be entered



SAP will allow many vendors to be linked, while generating a new RFQ Document Number for each vendor.

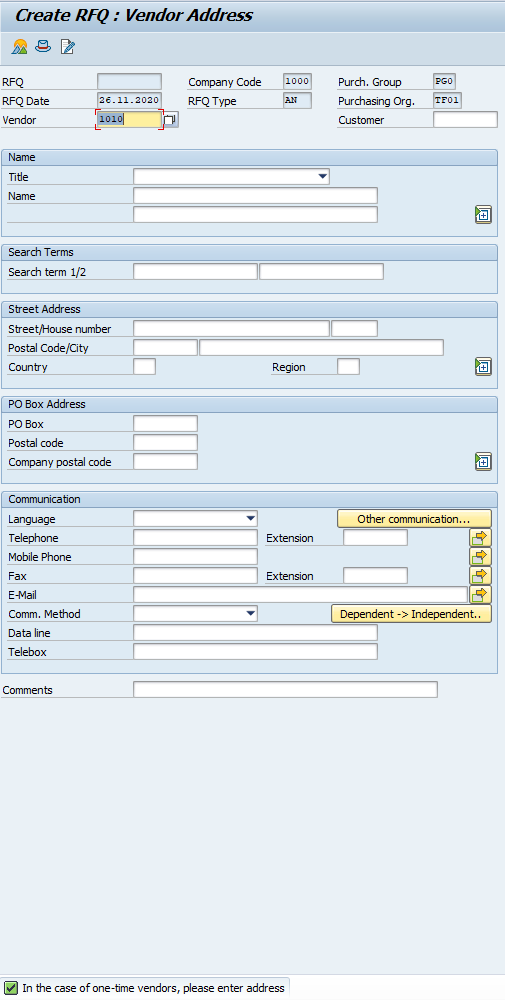
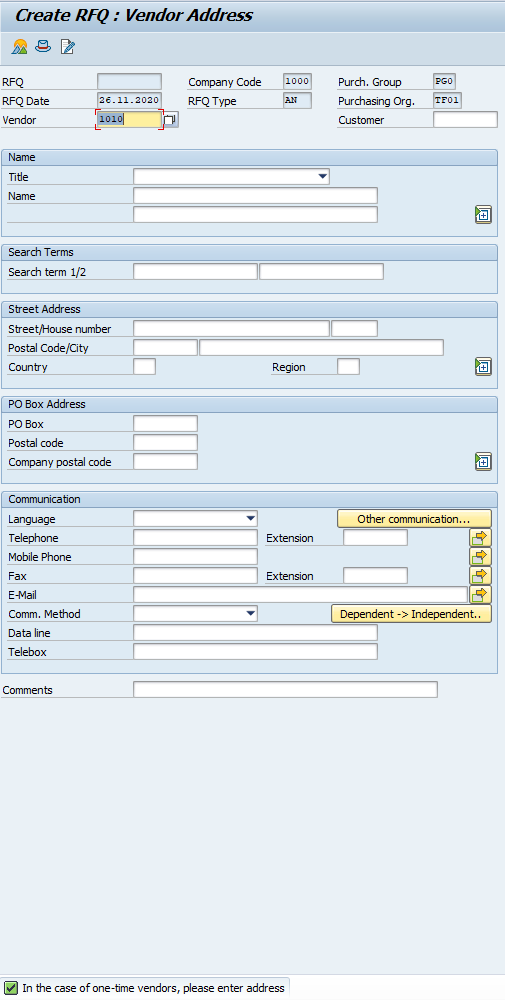
RFQ Number of Previous Vendor Document shown

**NOTE:** The Only Connecting factor is the Collective Number - if it is not in place, the process will have to start over.

For Each Vendor ensure:

* Vendor Email Address
* Vendor Cellphone

You May use ne-time Vendors as well:

# ME41 - On Save

On the Save-Event for each vendor added - Calls are made to:

**Class**: ZCL\_IM\_ME\_PURCHDOC\_POSTED **Method:** POSTED

Creates Header Record on first call in table - **ZRFC\_OBJ**

1. GUID – Unique number
2. OBJ -> Collective number
3. OBJ\_TYPE - RFQUOTE
4. DATA: dates record for Status Dates
5. CONTEXT: HEADER
6. KEYS: Status keys

Once the Header is created – each vendor (each RFQ Document) gets processed:

First: Check that Vendor Email address exists on table ZPARTNER – Create if needed.

Second: Create a record in **ZRFC\_OBJ** with:

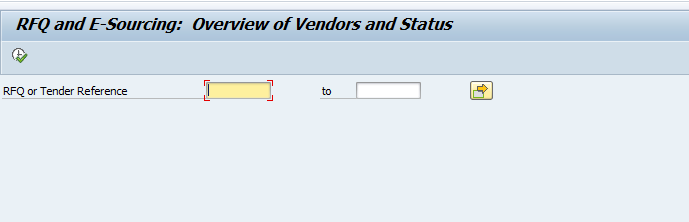
1. GUID – same as Header Record
2. OBJ – Email Address of Vendor
3. OBJ\_TYPE – Vendor Number
4. DATA: dates record for last visited, replied etc.
5. DATA\_TYPE: Collective number
6. CONTEXT: Unique GUID for Tokens

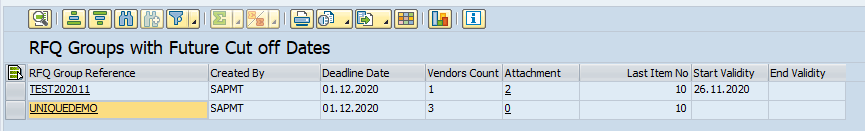
# ZERFQ Tender Cockpit

On the Tender Cockpit, the responses and activities of bidders can be seen.

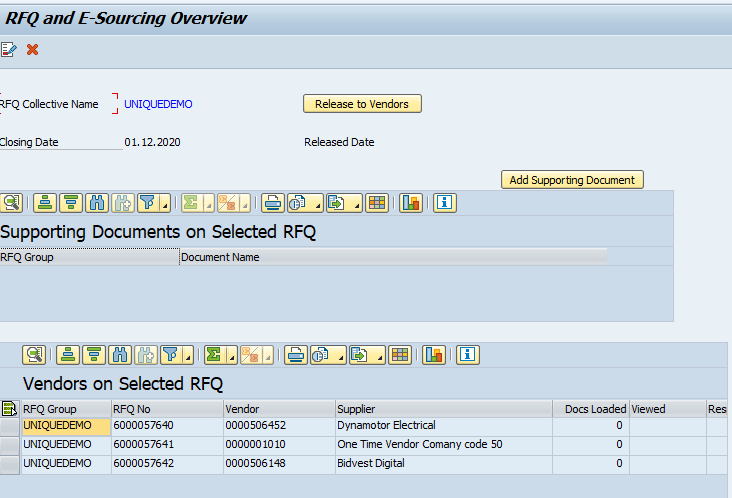
The importance of the Collective Number will be evident here.

Go to SAP transaction ZERFQ and you can filter by the Collective number.



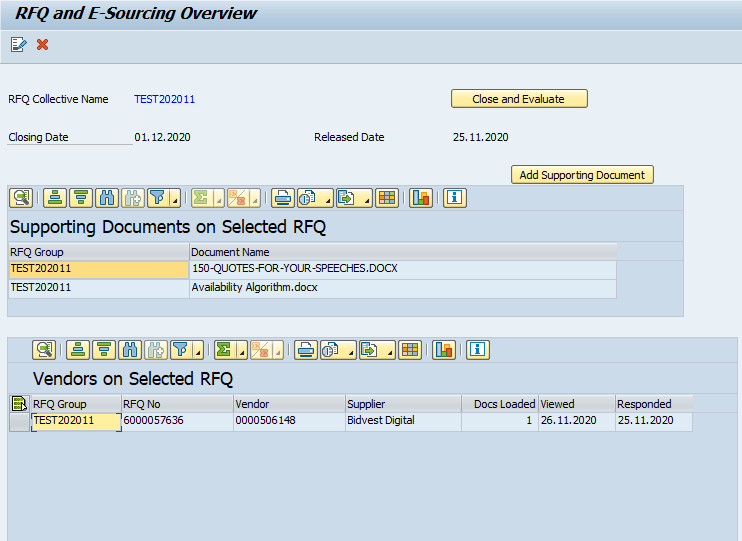


Click on the Collective number for more detail:



Notice the ***Release to Vendors*** and ***Add Supporting Document*** buttons.

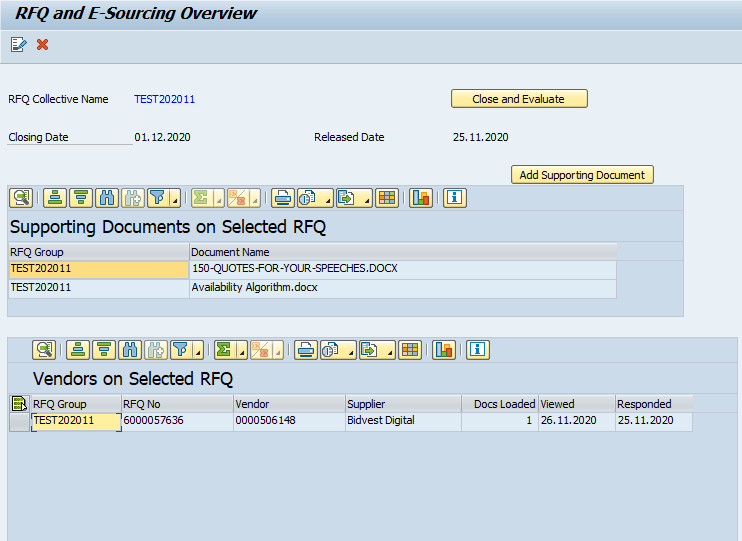
# Add Documents for vendors to reference



The Bidder will be able to see the documents on the website and download them for reference.

There are limits (5mb) for document sizes but try keep them small and compact.

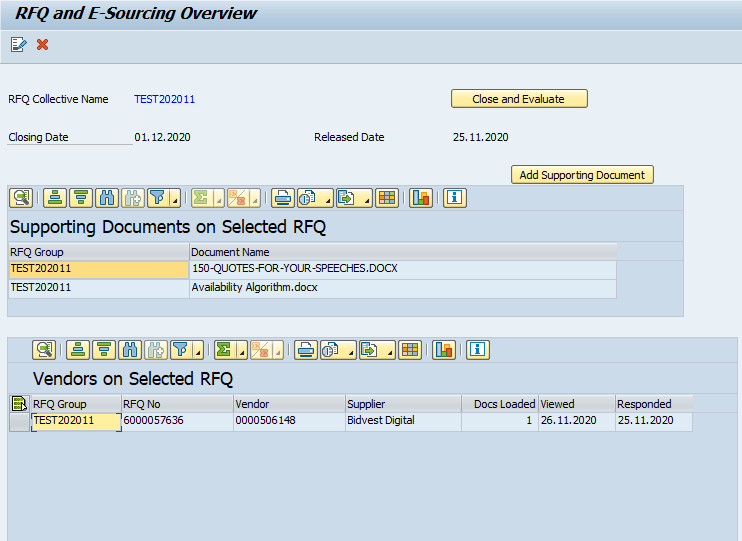
# Release the RFQ

When you are ready to send the RFQ to all the vendors then Press Release to Vendors. 

It calls class ZCL\_RFQ\_PROCESSES method set\_rfq\_release.

It will Release the Tender and on the Cockpit, the Button to ***Close and Evaluate*** is enabled.

# Monitor activities on Tender



On the Tender Cockpit, the responses and activities of bidders can be seen.

While the prices and documents are not accessible before the Tender has been closed, the Cockpit shows when a Vendor last viewed the RFQ and if they have uploaded any documents.

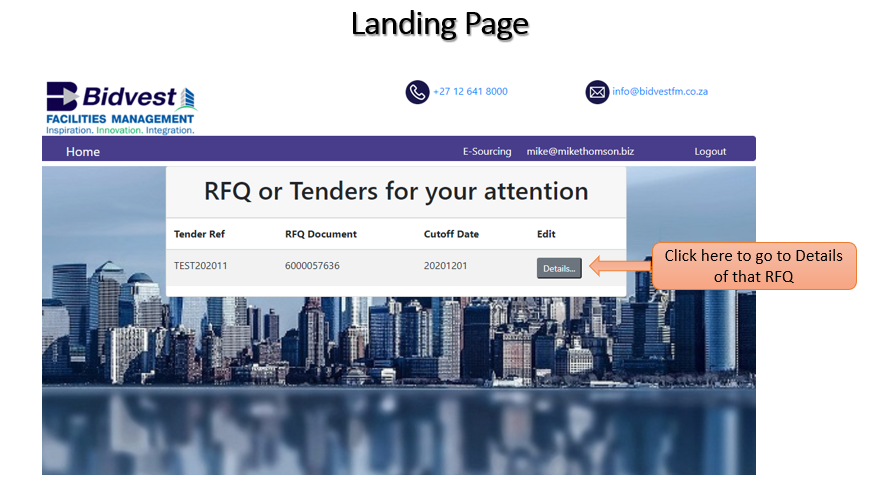
# Vendor will receive SMS and Email with a Link



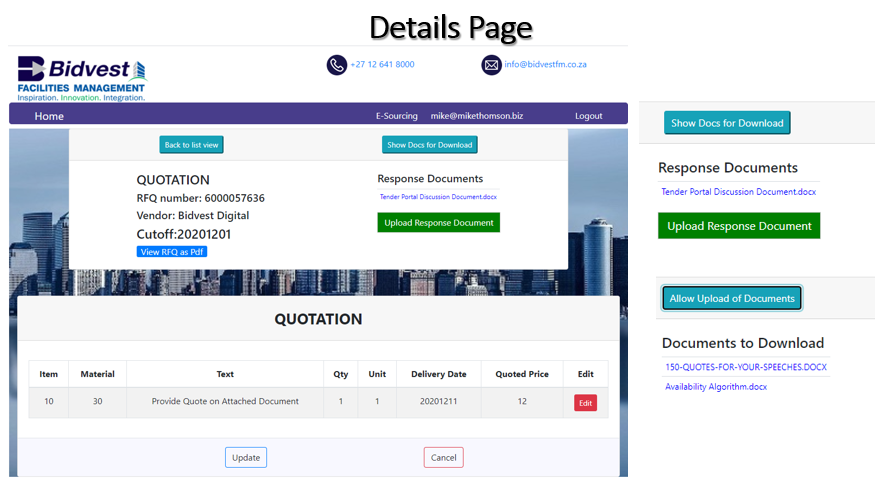


# E-sourcing screen

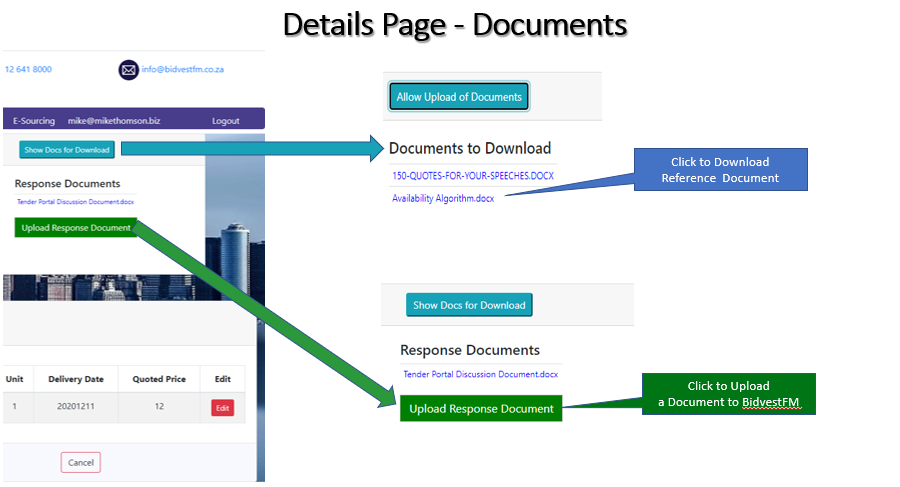
On logon, the bidder will land on the overview page that shows all tenders they are part of.



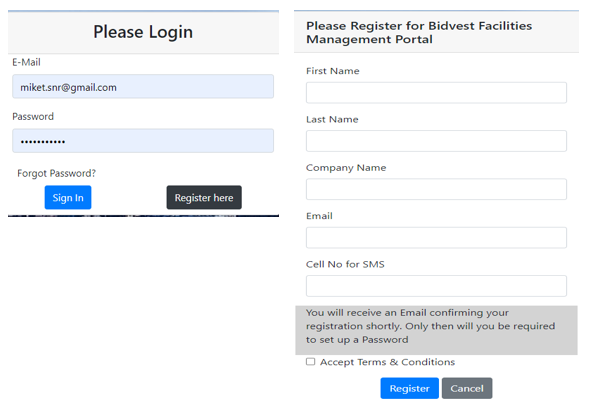
# Details Page



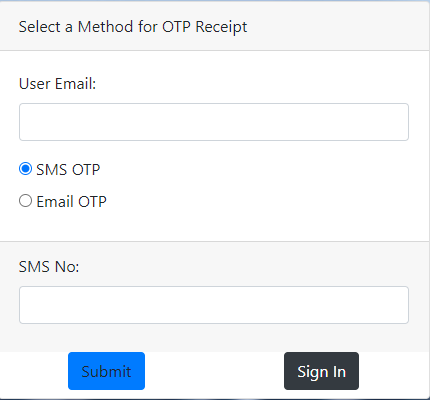
# Loading and Downloading Documents



# Optional Login or registration



# Reset Password with OTP



# Close and Evaluate

When the time is ready, the tender can be closed.

# On Sign on

The URL has 2 parameters, ‘REF’ and ‘SYSID’.

SYSID is dev or production and is handled in the pre-processing webpage to point to the correct website.

REF is made up of 2 parts, separated by a ‘-‘. The 2 parts are:

* The GUID of the Header record and
* The CONTEXT field of the RFQ Document entry in ZRFC\_OBJ. This is also a GUID.

A Call is made to SAP (System determined by parameter SYSID) to get the relevant user details.

SAP will return a token that is stored in the user’s browser, and this browser will allow access to data regarding all RFQ’s related to the email linked to that token. If the user has an elapsed token – they get a renewed one.

# Password Reset

A series of events take place.

**Password Reset requested**

https://io.bidvestfm.co.za/BIDVESTFM\_API\_AUTH/api/GETFLEX?Partner=ALL&Class=PWDR&CallContext=%7B%22USERNAME%22:%22mike@mikethomson.biz%22,%22TYPEOF%22:%22EMAIL%22,%22CELLNO%22:%22%22%7D

**OTP Received and captured for confirmation**

https://io.bidvestfm.co.za/BIDVESTFM\_API\_AUTH/api/GETFLEX?Partner=ALL&Class=OTPC&CallContext=%7B%22OTP%22:%22qufc%22,%22TOKEN%22:%22EB2F963E375415F1B80C0050568FBAD9%22%7D

**New Password Set**

This is a POST method

https://io.bidvestfm.co.za/BIDVESTFM\_API\_AUTH/api/Updatepwd

**Token Issued and used in the Process**

https://io.bidvestfm.co.za/BIDVESTFM\_API\_AUTH/api/GETFLEX?Partner=ALL&Class=USER&CallContext=%7BTOKEN:EB2F966E21B7EAF1B80C0050568FBAD9%7D

**Username etc returned.**

{"ServicesList":[{"JsonContext":null,"JsonsetName":"UNAM","JsonsetJstext":"{ \"UCEMAIL\":\"MIKE@MIKETHOMSON.BIZ\", \"EMAIL\":\"mike@mikethomson.biz\", \"FIRSTNAME\":\"Bidvest Digital\", \"LASTNAME\":\"t/a Bidvest SOHO\", \"CELLNO\":\"\", \"OBJ\":\"\", \"OBJTYP\":\"\", \"DATA\":\"\", \"DATA\_ROLE\":\"\"}"}],"refstatus":true}

# Technical Design – SAP Side

There are 2 different ways in which to generate a Request for Quotation:

1. Follow the Formal SAP RFQ Process
2. Reference a SAP Object ( Notification, Works Order or Project) and send out a non-standard RFQ.

## SAP RFQ Process

A RFQ document is created in the system for each bidder selected in Transaction ME41. These Documents are linked by the Collective Number in the Header field. This is painfully short (10 characters).

On the Save of the First document, a RFQUOTE - HEADER entry is made into the table ZRFC\_OBJ with a new unique GUID that is copied to all the individual entries later. The HEADER document will carry the status of the collective group (NEW , RELEASED, CLOSED) in the “data\_type” field.

The “DATA” field contains a JSON object that contains information on “DUE\_DATE” and “RELEASE\_DATE”.

The “TAGS field will contain any long text to and from the Bidder in JSON format.

For each Vendor (Bidder), an entry is made in ZRFC\_OBJ with the same GUI but more Bidder-Specific details:

* Key Fields:
  + “GUID” is the Guid of the header line.
  + “OBJ” contains the Email address (in Upper Case for search purposes)
  + “OBJ\_TYPE” contains the Bidder Account number and can be ‘1010’ where it is a “ONE-TIME VENDOR”
  + ”CONTEXT” contains a Token for unique identification of the line – does not link to any other line but is used for linking Documents that are uploaded from the Bidder
  + “SUB\_CONTEXT” is the RFQ Document Number in SAP for the standard Process
* Non Key Fields:
  + “DATA” contains a JSON Object for comments from the Bidder
  + “DATA\_TYPE” contains the Collective Text.
  + “TAGS” – To be Advised.
  + “KEYS” – To be advised.

## Non RFQ Document Process

The non SAP RFQ Process is simpler in that there is not a RFQ document to reference and only a single long (Plain) text that explains the requirement. Of course, there can be many documents attached for further detail.

For each Vendor (Bidder), an entry is made in ZRFC\_OBJ with the same GUI but more Bidder-Specific details:

* Key Fields:
  + “GUID” is the Guid of the header line.
  + “OBJ” contains the Email address (in Upper Case for search purposes)
  + “OBJ\_TYPE” contains the Bidder Account number and can be ‘1010’ where it is a “ONE-TIME VENDOR”
  + ”CONTEXT” contains a Token for unique identification of the line – does not link to any other line but is used for linking Documents that are uploaded from the Bidder
  + “SUB\_CONTEXT” is not used at present
* Non Key Fields:
  + “DATA” contains a JSON Object of the reply text from the bidder
  + “DATA\_TYPE” contains the Collective Text
  + “TAGS” – To be Advised
  + “KEYS” – To be advised

# SAP Objects in the code

The Standard Process uses the Standard ME41 Process and the objects are called in SAP extension methods (BADI) :

* ZCL\_IM\_ME\_PUCHDOC\_POSTED as the RFQ Doc is saved for creation and changes.
* ZCL\_RFQ\_PROCESSES method *RFQ\_UPDATED* is called from ZCL\_IM\_ME\_PURCHDOC\_POSTED in the POSTED method.
* ZCL\_WEB\_USERS\_AUTHS method GET\_EMAIL\_FORM\_PO is used to get the contact person of the selected vendor.
* ZCL\_RFC\_UTIL method GET\_GUID is used to generate Tokens and GUID.
* ZCL\_RFC\_OBJ method CREATE is used to write the HEADER entry in the ZRFC\_OBJ table.

The Non Standard Process is triggered by the Program ZACTION\_RFQ.

Most entries are done using SQL statements and the only objects used are:

* ZCL\_RFC\_UTIL method GET\_GUID is used to generate Tokens and GUID.
* ZCL\_RFC\_OBJ method CREATE is used to write the header entry in the ZRFC\_OBJ table . The vendor lines are added programmatically.

# Test Scenario

## Standard RFQ

Create a RFQ with at least 2 vendors using ME41

1. Remember to use an Unique Submi (Collective Number).
2. Fill address details to have at least an Email address

Go to ZERFQ and execute.

Add Reference Documents

Release when Ready

## Custom User Defined RFQ

Run Program ZACTION\_RFQ – or from Action in Notification.

# WEB Client Side

## EMAIL or SMS Link

Dadasd

## Website

### Pages

11

### RFC API Calls

#### getRfqList(Vendor: string)

getRfqList(vendor: string) {

const lrfqList: RFQHeader[] = [];

let rfqtokenstring = '';

const rfqtoken = localStorage.getItem('token') ;

if (rfqtoken) {

rfqtokenstring = ',RFQTOKEN:' + rfqtoken ;

} else {

rfqtokenstring = ' ';

}

const context = '{' + 'EMAIL:' +

this.auths.currentUserValue.username +

rfqtokenstring +

',VENDOR:' + vendor + ',HEADER:X }';

const params = new HttpParams()

.set('Partner', 'ALL')

.set('Class', 'RFQL')

.set('CallContext', context);

this.http

.get<any>(environment.BASE\_API + '/api/sap/rfq/getlist/email' , {

params

})

.subscribe(data => {

if (data.ServicesList instanceof Array) {

for (const rfqdoc of data.ServicesList) {

const tobj = JSON.parse(rfqdoc.JsonsetJstext);

const rfqline: RFQHeader = new RFQHeader();

rfqline.SUBMI = tobj.SUBMI;

rfqline.RFQNO = tobj.RFQNO;

rfqline.CUTOFF = tobj.CUTOFF;

rfqline.VENDORNO = tobj.VENDORNO;

rfqline.VENDOR = tobj.VENDOR;

lrfqList.push(rfqline);

}

this.currentRFQList.next(lrfqList);

}

});

}

#### getRfqItems(rfqno:string)

getRfqItems(rfqno: string) {

const lrfqItems: RFQItem[] = [];

const context = '{' + 'RFQNO:' + rfqno + ',DETAILS:X }';

const params = new HttpParams()

.set('Partner', 'ALL')

.set('Class', 'RFQL')

.set('CallContext', context);

this.http

.get<any>(environment.BASE\_API + '/api/sap/rfq/GETDETAIL/' + rfqno, {

params

})

.subscribe(data => {

if (data.ServicesList instanceof Array) {

for (const rfqdoc of data.ServicesList) {

const tobj = JSON.parse(rfqdoc.JsonsetJstext);

const rfqItem: RFQItem = new RFQItem();

rfqItem.SUBMI = tobj.SUBMI;

rfqItem.RFQNO = tobj.RFQNO;

rfqItem.ITEMNO = tobj.ITEMNO;

rfqItem.CUTOFF = tobj.CUTOFF;

rfqItem.DELIVERYDATE = tobj.DELIVERYDATE;

rfqItem.MTEXT = tobj.MTEXT;

rfqItem.QUANTITY = tobj.QUANTITY;

rfqItem.MATERIAL = tobj.MATERIAL;

rfqItem.UNIT = tobj.UNIT;

rfqItem.PRICE = tobj.PROMISEPRICE;

lrfqItems.push(rfqItem);

}

this.currentRFQItems.next(lrfqItems);

this.getRfqAttachments(rfqno);

}

});

}

#### getRfqAttachments(rfqno:string)

getRfqAttachments(rfqno: string) {

const lclsubmilist = [];

const lclchosenlist = [];

const context = '{APIKEY:RFQ, DOCNO:' + rfqno + ',COUNTER:0 }';

const params = new HttpParams()

.set('Partner', 'ALL')

.set('Class', 'RFDL')

.set('CallContext', context);

this.http

.get<any>(environment.BASE\_API + '/api/GETFLEX', { params })

.subscribe(data => {

if (data.ServicesList instanceof Array) {

for (const rfqdoc of data.ServicesList) {

const tobj = JSON.parse(rfqdoc.JsonsetJstext);

const docItem: DMSHeader = new DMSHeader();

docItem.id = tobj.COUNTER;

docItem.DOCNO = tobj.DOCNO;

docItem.COUNTER = tobj.COUNTER;

docItem.ORIGINALNAME = tobj.ORIGINALNAME;

docItem.FILESIZE = tobj.FILESIZE;

docItem.MIMETYPE = tobj.MIMETYPE;

docItem.APIKEY = tobj.APIKEY;

if (docItem.APIKEY === 'RFQQUOTE') {

lclsubmilist.push(docItem);

} else {

lclchosenlist.push(docItem);

}

}

this.chosendoclist.next(lclchosenlist);

this.chosensubmilist.next(lclsubmilist);

}

});

}

#### getvendordoc(docref: DMSHeader)

getvendordoc(docref: DMSHeader) {

let datain = '';

const context = docref.APIKEY + '-' + docref.DOCNO + '-' + docref.COUNTER;

const params = new HttpParams()

.set('Partner', 'ALL')

.set('Class', 'RFQD')

.set('CallContext', context);

this.http

.get<any>(environment.BASE\_API + '/api/GETFLEX', { params })

.subscribe(data => {

if (data.ServicesList instanceof Array) {

for (const dmsdoc of data.ServicesList) {

const tobj = JSON.parse(dmsdoc.JsonsetJstext);

datain = datain + tobj.ROLES;

}

const b64Data = datain;

if (b64Data !== undefined) {

this.currentblob.next(

this.b64toBlob(b64Data, docref.MIMETYPE, 512)

);

}

}

});

}

#### UpdateRfqItem(itemno: string)

UpdateRfqItem(itemno: string) {

const lrfqItems: RFQItem[] = [];

const context = itemno;

const params = new HttpParams()

.set('Partner', 'ALL')

.set('Class', 'RFQL')

.set('CallContext', context);

this.http

.get<any>(environment.BASE\_API + '/api/GETFLEX', { params })

.subscribe(data => {

if (data.ServicesList instanceof Array) {

for (const rfqdoc of data.ServicesList) {

const tobj = JSON.parse(rfqdoc.JsonsetJstext);

const rfqItem: RFQItem = new RFQItem();

rfqItem.SUBMI = tobj.SUBMI;

rfqItem.RFQNO = tobj.RFQNO;

rfqItem.ITEMNO = tobj.ITEMNO;

rfqItem.CUTOFF = tobj.CUTOFF;

rfqItem.MTEXT = tobj.MTEXT;

rfqItem.QUANTITY = tobj.PROMISEQTY;

rfqItem.PRICE = tobj.PROMISEPRICE;

lrfqItems.push(rfqItem);

}

this.currentRFQItems.next(lrfqItems);

}

});

}

#### uploadQuoteFile2SAP(file, resultobj, filerefer, vendor)

uploadQuoteFile2SAP(file, resultobj, filerefer, vendor) {

const data = resultobj.split(',').pop();

const httpOptions = {

headers: new HttpHeaders({

'Content-Type': 'application/json',

Authorization: 'Basic dXNlcm5hbWU6cGFzc3dvcmQ='

})

};

const uploadvar = {

callType: 'post',

chContext: {

CLASS: 'ATTACH',

METHOD: ''

},

chData: {

fileName: file[0].name,

fileSize: file[0].size,

fileType: file[0].type,

fileContent: data,

uname: this.currentUser.username,

targetObjId: filerefer,

targetObjType: 'RFQDOC',

extras: vendor,

apikey: 'RFQQUOTE'

}

};

this.http

.post<any>(environment.BASE\_POST, uploadvar, httpOptions)

.subscribe(

data => {

console.log(data);

},

e => {

console.log(e);

}

);

}

#### updateSAPItem(line: RFQItem)

updateSAPItem(line: RFQItem) {

let lcldatestr = '';

lcldatestr = line.DELIVERYDATE.split('-').join('');

const httpOptions = {

headers: new HttpHeaders({

'Content-Type': 'application/json',

Authorization: 'Basic dXNlcm5hbWU6cGFzc3dvcmQ='

})

};

const uploadvar = {

callType: 'post',

chContext: {

CLASS: 'UPDATEQUOTE',

METHOD: 'RFQP'

},

chData: {

EBELN: line.RFQNO,

EBELP: line.ITEMNO,

NETPR: line.PRICE,

EINDT: lcldatestr,

KTMNG: line.QUANTITY,

CREATEDBY: this.currentUser.username

}

};

this.http

.post<any>(environment.BASE\_POST, uploadvar, httpOptions)

.subscribe(

data => {

console.log(data);

},

e => {

console.log(e);

}

); }