Improving the business efficiency of OEBS workflow
Part 2: Validating OEBS E-mail Addresses

By
Gary Piper
www.PIPER-Rx.com

Introduction

Following on from ***"Part 1: Missing E-mail Addresses"*** we now move on to validating e-mail addresses.

The only fool proof method of validating an e-mail address is to send one and wait for a reply. However this is often not practical; this could really annoy your customers and could be considered SPAM which is illegal in some countries (it's about time!)….

The next best alternative is to comprehensively validate the e-mail address format which is what we will now cover.

There are a number of "simple" e- mail validation checks available on the web, from single validations like there must be at least one @ symbol, to more complex checks for a few items including invalid characters. Here is one example…

REGEXP

There are a number of checks available using the ***regexp*** feature that is available in 10g and above:

```
SELECT user_name,
       email_address,
       regexp_substr(email_address,'[a-zA-Z0-9._%-]+@[a-zA-Z0-9._%-]+\.[a-zA-Z]{2,4}')
status
 FROM  applsys.fnd_user;
```

Example Output

User Name	E-mail Address	Status
MFG	Jsmith	
MHARDING	Mharding	
RT1222426680	RT@localhost.oracleads.com	RT@localhost.oracleads.com
RT1222427321	RT@RT.com	RT@RT.com
SDAVISON	Sdavison	
STYSON	STYSON	

In this example a status value of null indicates the e-mail address failed the ***regexp*** validation.

IMHO you need a lot of experience to interpret this, and if you are still on Oracle version 9, which some OEBS sites are, it just won't work.

Typically the information found on the web relating to e-mail validation in an OEBS environment is quite piece meal and none of it readily usable for validating OEBS workflow e-mail addresses. As a result I wrote my own 12 point e-mail validation PLSQL function (as you do!). This paper provides both the function and the SQL to validate the e-mail addresses in workflow tables.

OEBS e-mail validation

This section covers checking the validity of e-mail addresses in both the ***wf_local_users*** and ***wf_local_roles*** objects.

To re-iterate what was covered in the ***"Part 1: Missing e-mail Addresses"***, the added complexity is that both the ***wf_local_roles*** and ***wf_local_roles*** tables store information including e-mail addresses originating from various other Oracle Applications modules such as ***applsys.fnd_users***, ***hr.per_all_people_f*** and ***ar.ra_contacts***…The information in these objects is synchronised from base application tables using the WF synchronisation concurrent programs. Oh and to make matters worse, the source information is very OEBS version dependant.

So first we need to find the invalid e-mail address used by workflow in the ***wf_local_roles*** and ***wf_local_roles*** tables and then, as we did in ***"Part 1: Missing e-mail Addresses"*** use the ***orig_system*** identifier to find the base table that holds the master e-mail address and correct the issue at the source ready for the next synchronisation.

Validating the e-mail address format

The PLSQL e-mail validation function performs the following twelve 12 basic e-mail validation checks. A valid e-mail address format must:

- Have a minimum length of 5 characters A@A.A
- Have a minimum of one dot
- Contain only one @ symbol
- Not have an @ and dot appear together
- Not have the @ symbol as the first character
- Not have a dot as the first character
- Not have the @ symbol as the last character
- Not have a dot as the last character
- Not have two dots together
- Have one dot after the @ symbol
- Not contain invalid characters
- Not exist in your banned domains list (OPTIONAL)

The optional check for banned domains includes generic e-mail domains such as facebookmail.com, gmail.com etc… You can add more at your own leisure. That being said, given the number of dodgy domains out there you should consider creating a table for banned domains…

As mentioned, this check is optional (default on). To turn this check off change the following line in the function code from:

```
v_check_banned_domains  boolean  :=  TRUE;
TO
v_check_banned_domains  boolean  :=  FALSE;
```

The e-mail validation function can be used to validate any e-mail address.

I am sure there are many more validation checks that can be included, but this is enough for now.

Step One – Download and install the function

The open source e-mail address validation function source code can down loaded from the PIPER-Rx website under the papers section.

In its simplest form the function can be executed as follows:

```
SELECT 'gary@piper-rx.com' email_address,
       piper_rx_email_validation ( 'gary@piper-rx.com') status
  FROM dual;
```

EMAIL_ADDRESS	STATUS
gary@piper-rx.com	PASS

Step Two – Validate the e-mail addresses in your *wf_local_roles* table

You can use the following SQL to validate the e-mail addresses in the ***wf_local_roles*** table:

```
SELECT wlr.name,
       wlr.orig_system_id,
       wlr.orig_system,
       wlr.email_address,
       initcap(piper_rx_email_validation ( wlr.email_address )) status
  FROM applsys.wf_local_roles wlr
 WHERE wlr.email_address is not null
   and (wlr.expiration_date is null
        or wlr.expiration_date > sysdate)
   and wlr.status = 'ACTIVE'
   and wlr.notification_preference != 'QUERY'
 ORDER by wlr.name;
```

Note: We excluded the ***wlr.notification_preference*** of QUERY as this notification option does not send e-mail notifications.

Example Output

Name	Orig id	Orig System	Email Address	Status
HZ_PARTY:37536	37536	HZ_PARTY	tester11	Fail
HZ_PARTY:3890	3890	HZ_PARTY	Operations Vice President	Fail
HZ_PARTY:401670	401670	HZ_PARTY	andrebeaulie@businessworld.com	Pass
HZ_PARTY:401672	401672	HZ_PARTY	RT@localhost.oracleads.com	Pass
HZ_PARTY:401677	401677	HZ_PARTY	RT@RT.com	Pass
HZ_PARTY:4030	4030	HZ_PARTY	harrison118Wbusinessweekmail.com	Fail
HZ_PARTY:4131	4131	HZ_PARTY	pino.ludovici	Fail

If you are using one of the latest releases of OEBS then the following SQL may also work:

```
SELECT wlr.name,
       wlr.orig_system_id,
       wlr.orig_system,
       wlr.email_address,
       initcap(piper_rx_email_validation ( wlr.email_address )) status
  FROM applsys.wf_local_roles wlr
 WHERE wlr.email_address is not null
   and (wlr.expiration_date is null
        or wlr.expiration_date > sysdate)
   and wlr.status = 'ACTIVE'
   and wlr.notification_preference != 'QUERY'
   and piper_rx_email_validation ( wlr.email_address ) = 'FAIL'
ORDER by wlr.name;
```

Step Three – Validate the originating application e-mail addresses

Next, assuming the originating system was USERS you simply run the check against ***applsys.fnd_users***. If the originating system was HZ_PARTY run the check against ***ar.hz_parties*** etc…

```
SELECT fu.user_name,
       fu.email_address,
       initcap(piper_rx_email_validation ( fu.email_address )) status
  FROM applsys.fnd_user fu
 ORDER by fu.user_name;
```

Example Output

User Name	E-mail Address	Status
CHORTON	chorton@vision.com	Pass
DGRAY	Dgray	Fail
DHOF	dhof@vision.com	Pass
DISTRIBUTION	pstock@vision.com	Pass
FEEDER SYSTEM	R11_SECURITY	Fail
HSPRAGUE	hsprague	Fail
I2	jbernste@vision.com	Pass
JFEINBERG	Jfein	Fail
JFROST	jfrost@vision.com	Pass
JSELLER	jhseller@vision.com	Pass
JSMITH	jsmith@visionops.com	Pass
KBROCK	kbrock@vision	Fail
KLANGHAM	klangham@vision.com	Pass
KWALKER	KWALKER	Fail
MFG	Jsmith	Fail
OPERATIONS	pstock@visionops.us.com	Pass
ORACLE	KELKINS	Fail
SERVICES	Amarlin	Fail

Step Four - Repeat the process for the *wf_local_users* table

Run the following SQL against your ***wf_local_users*** table:

```
SELECT wlu.name,
       wlu.orig_system_id,u
       wlu.orig_system,
       wlu.email_address,
       initcap(piper_rx_email_validation ( wlu.email_address )) status
  FROM applsys.wf_local_users wlu
 WHERE wlu.email_address is not null
   and (wlu.expiration_date is null
        or wlu.expiration_date > sysdate)
   and wlu.status = 'ACTIVE'
   and wlu.notification_preference != 'QUERY'
 ORDER by wlu.name;
```

If you are using one of the latest releases of OEBS then the following SQL may also work:

```
SELECT wlu.name,
       wlu.orig_system_id,
       wlu.orig_system,
       wlu.email_address,
       initcap(piper_rx_email_validation ( wlu.email_address )) status
  FROM applsys.wf_local_users wlu
 WHERE wlu.email_address is not null
   and (wlu.expiration_date is null
        or wlu.expiration_date > sysdate)
   and wlu.status = 'ACTIVE'
   and wlu.notification_preference != 'QUERY'
   and piper_rx_email_validation ( wlu.email_address ) = 'FAIL'
 ORDER by wlu.name;
```

Other source tables

The following covers **some** (not all) of the "other" OEBS e-mail address source tables, however be aware that the source tables are very version dependant. As such the following SQL may not work without modification on your version of OEBS:

AR.HZ_PARTIES

```
SELECT hp.party_id,
       substr (hp.party_name, 1, 30) ||
           decode(sign(length(hp.party_name) -30), 1, '...') contact_name,
       hp.email_address,
       initcap(piper_rx_email_validation ( hp.email_address )) status
  FROM ar.hz_parties hp;
 WHERE email_address is not null;
```

HR.PER_ALL_PEOPLE_F

```
SELECT papf.employee_number,
       substr (papf.first_name||' '|| papf.last_name, 1, 30) ||
           decode(sign(length(papf.first_name||' '|| papf.last_name) -30), 1,
'...') employee_name,
       papf.email_address email_address,
       initcap(piper_rx_email_validation ( papf.email_address )) status
  FROM hr.per_all_people_f papf
 WHERE email_address is not null
   and (papf.effective_end_date is not null
        or papf.effective_end_date > sysdate);
```

AR.RA_CONTACTS

```
SELECT rc.contact_id,
       substr (rc.first_name||' '|| rc.last_name, 1, 30) ||
           decode(sign(length(rc.first_name||' '|| rc.last_name) -30), 1, '...')
contact_name,
       rc.email_address,
       initcap(piper_rx_email_validation ( rc.email_address )) status
  FROM ar.ra_contacts rc
 WHERE rc.email_address is not null
   and rc.status = 'A';
```

AP.AP_SUPPLIER_CONTACTS

```
SELECT apsc.vendor_contact_id,
       substr (apsc.first_name ||' '||apsc.last_name, 1, 40) ||
           decode(sign(length(apsc.first_name ||'.'||apsc.last_name) - 40), 1,
'...') contact_name,
       apsc.email_address,
       initcap(piper_rx_email_validation ( apsc.email_address )) status
  FROM ap.ap_supplier_contacts apsc
WHERE apsc.email_address is not null
  and (apsc.inactive_date is null
       or apsc.inactive_date > sysdate);
```

You get the drift. Basically the function can be used for any e-mail address.

Conclusion

Given the propensity for invalid e-mail addresses to clog up workflow, if you find just one e-mail address that is fixed where subsequent workflow notifications do not get sent to an invalid address or generate a workflow error, it was worth the effort of installing the function and running the SQL.

Want to know more?

There is loads more ***FREE*** information on this topic and all aspects of OEBS Application Administration at the **PIPER-Rx** website. After over 20+ years working with Oracle (the

product, not the Company) and Oracle E-Business Suite (since Release 5) I have visited countless sites and pretty much seen it all when it comes to Applications Administration. Since the late 1990's I have spent more time sharing these learnings and the most popular papers and case studies I have presented are available at the **PIPER-Rx**.com website as well as a whole host of Tips and Reports I have used throughout my career.

All information at the **PIPER-Rx**.com website is ***FREE*** so why not check it out….I hope you find it useful! **– 30,000+ downloaders to date can't be wrong!**

Disclaimer

All material contained in this document is provided by the author "as is" and any express or implied warranties, including, but not limited to, any implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the author be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of any content or information, even if advised of the possibility of such damage. It is always recommended that you seek independent, professional advice before implementing any ideas or changes to ensure that they are appropriate.