Adding Activities to new tables

Adding activity tracking to tables in Service Manager that do not have activities out-of-box

HP® Software - Service Management

Introduction

Service Manager uses the activity tracking option in the Incident Management, Service Desk, Problem Management and Change Management modules. If you want to include activities in an area that does not yet use activities, you can tailor the system to do so by following the examples on how activities were implemented in these modules. This document gives step – by – step instructions for these implementations.

Requirements

Service Manager System Administrator access and experience with advanced tailoring are required.

Adding activity logging to a table

The following steps will show how to add activity logging to a record based on the example of Contact Information.

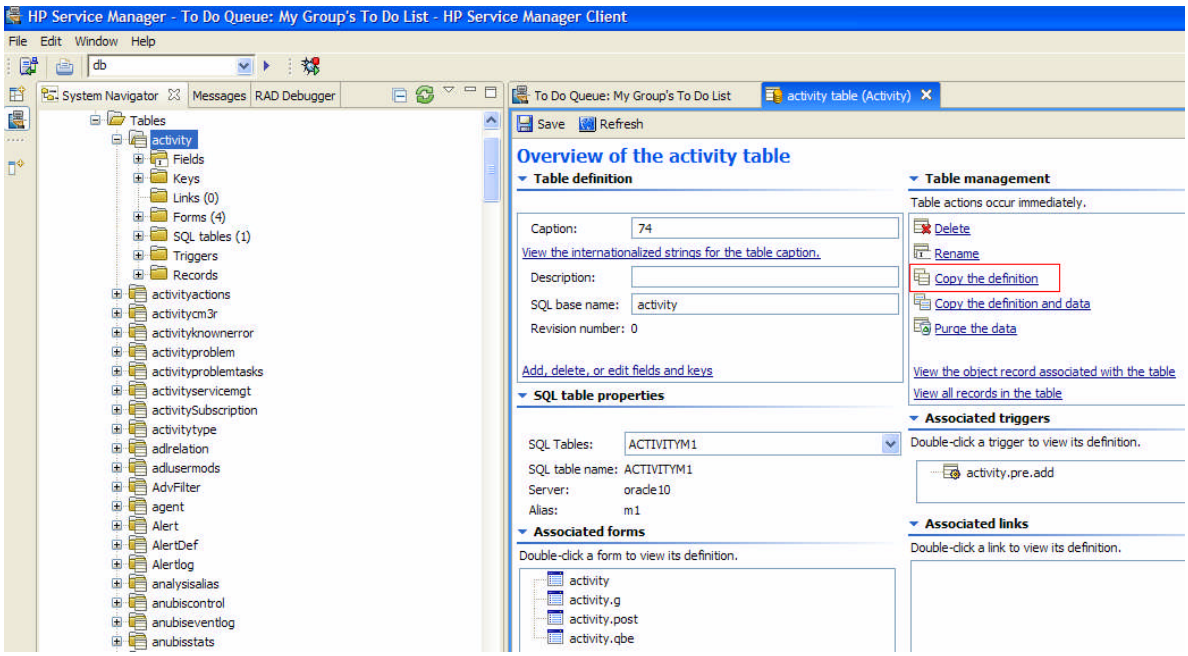
Note: The table associated to the form that uses activities has to have a valid dbdict record. Activities cannot be added to joined files such as joincomputer.

Step 1 – Creating the new activity file

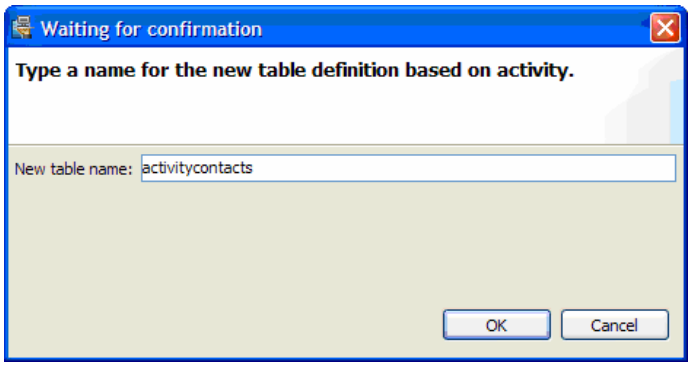
a. Go to the **System Definition Utility** in the **System Navigator**

b. Select the table named **activity** by double clicking

c. Click on **Copy the Definition**

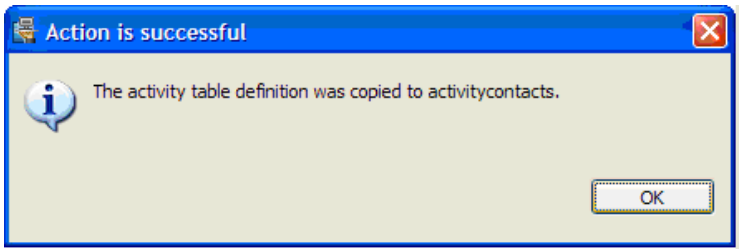


d. Name the new table **activity<*filename*>**



e. Click **OK**

f. Click **OK** when prompted with the message that the activity<*filename*> table has been created



g. Exit the System Definition Utility by closing the open tabs with the activity and activitycontacts definitions.

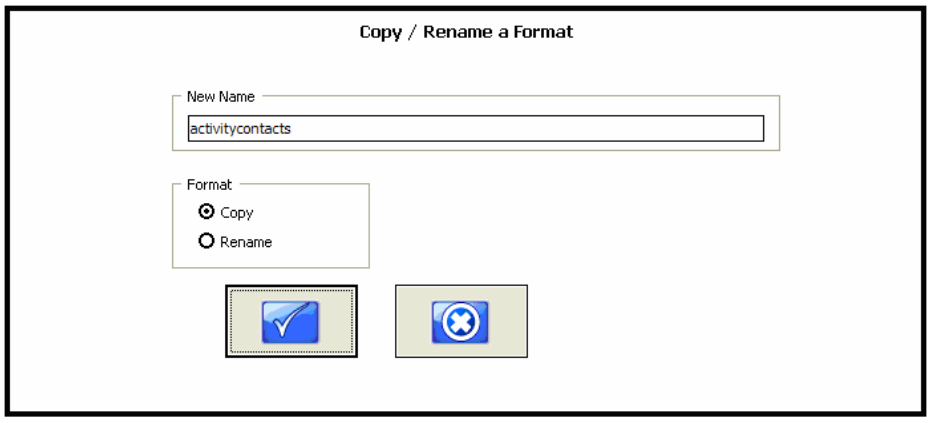
Step 2 – Creating the activity forms

a. Go to **Menu Navigation - Tailoring** – **Forms Designer**

b. Select the form named **activityTEMPLATE**

c. Select **Options** – **Copy / Rename**

d. Enter the new name of **activity**<***filename***>, select to **Copy** the form and click **OK**



e. Go into **Design** Mode

f. Change the title of the form to **Activity Log - <*Module*>** for example **Activity Log – Contacts**

g. Change the XXXXXX Number label to <Module Record> <unique key label>, for example **Contact Name**

h. If the Customer Visible feature should not be used, remove the flag from the form.

i. Click **OK** to save and exit out of Design Mode

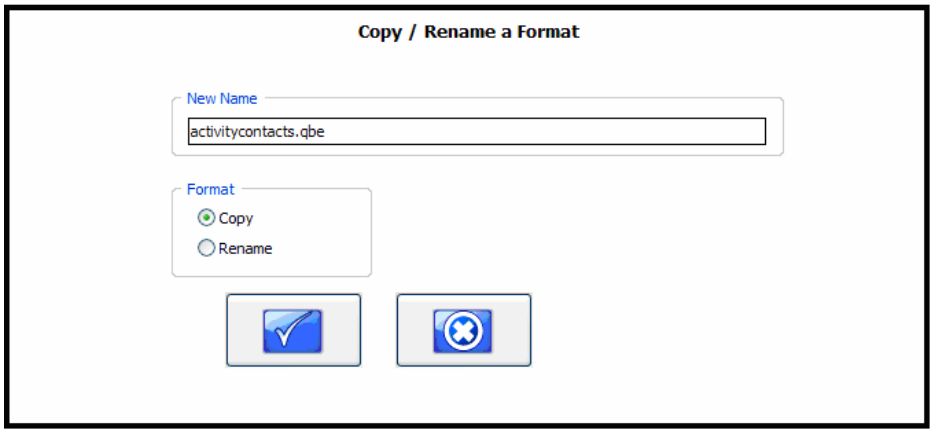
j. Under **Options** – **Filename**, enter the filename of the table created in step 1, for example **activitycontacts**.

k. Click **OK** to save and exit

l. In Forms Designer, select the **activityTEMPLATE.qbe** form

m. Select **Options** – **Copy / Rename**

n. Enter the new name of **activity<*filename*>.qbe**, select to **Copy** the form and click **OK**



o. Go into **Design** Mode

p. Change the XXXXXX Number column title to <Module Record> Number, for example **Contact Name**

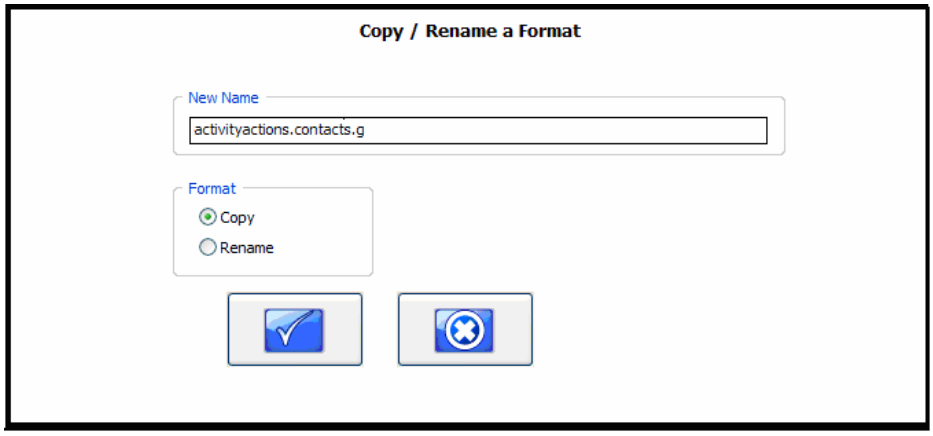
q. Click **OK** to save and exit Design mode

r. Click **OK** to save and exit

s. In Forms Designer, select the **activityactions.g** form

t. Select **Options** – **Copy / Rename**

u. Enter the new name of activityactions.<*filename*>.g, select to Copy the form and click OK



v. Click **OK** to save and exit

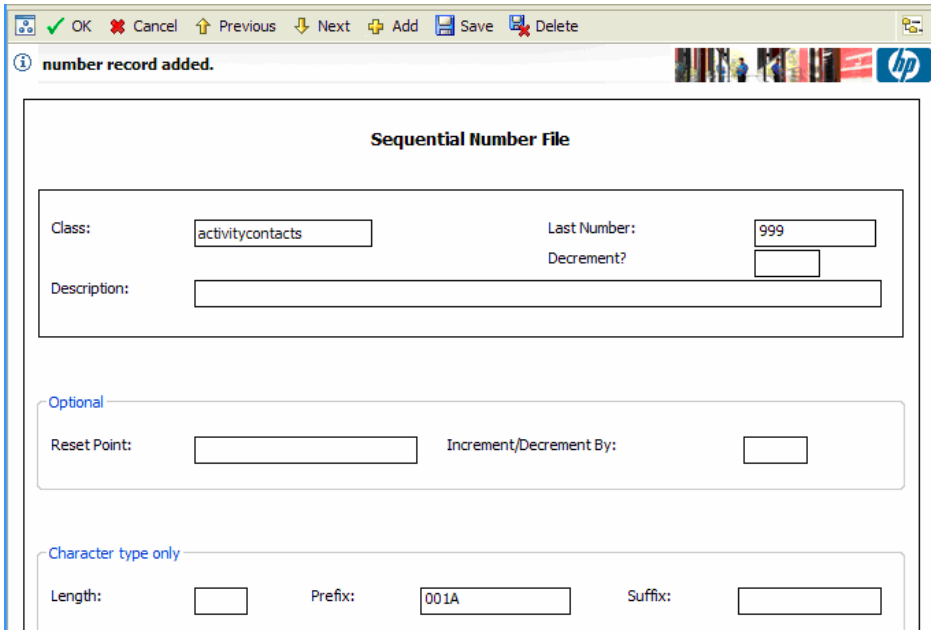
Step 3 – Creating the number record

a. Go to **Menu Navigation** – **Tailoring** – **Database Manager**

b. Select the **number** table

c. Select the **class="activity"** record

d. Enter the class of **activity<filename**. Set the Last Number to 999.



e. Click **Add** to add the number record

f. Click **OK** to save and exit

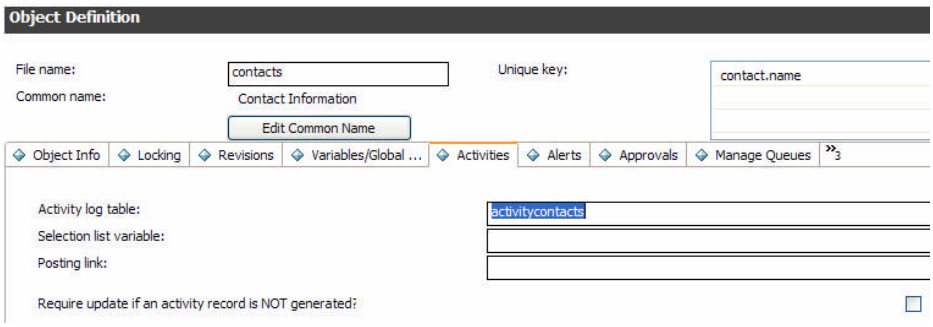
Step 4 – Enabling activities in the Object record

a. Go to **Tailoring** – **Document Engine**

b. Click on **Objects**

c. Select the Object record for the primary file to use activities

d. Add **activity<filename>** to the Activities Log Table field



e. Click **OK** to save and exit

Step 5 – Creating the pre-Add trigger

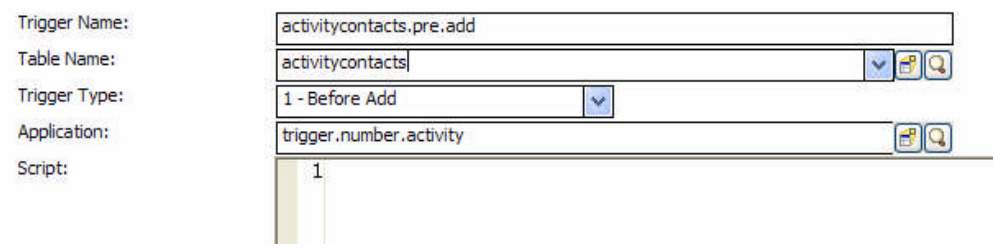
a. Go to **Tailoring** – **Database Manager**

b. Select the **triggers** table

c. Select the **name=" activity.pre.add"** record

d. Change the name to **activity<*filename*>.pre.add**

e. Change the table name to **activity<*filename*>**



f. Click **Add**

g. Click **OK** to save and exit

Step 6 – Preparing the primary table for activities

The table you are adding activities processing to may already have fields that hold the following information. If they do not exist, they have to be added:

- a read-only update history field
- an alias field for the unique key field of the primary table to use for linking to the activities

The viewing format needs to be modified to allow for the viewing of the activities. Add the following fields:

- a read-only update history field
- a temporary place-holder field for the current update being entered, defined with a variable input
- a description field (for the initial description that was entered for the ticket, that becomes read-only after initial creation)
- a tab that will virtual join into the activities table that was set-up previously
- a "filter" field that will allow you to filter out unwanted activity records from the virtual join list

Finally, the link record will have to be modified:

- Add a link line for the filter to work
- For bringing up an activity record by double-click, a second link line needs to be added (see the probsummary link record for an example).

On the example of the Contact information, these steps would be:

The read-only update history field will have to be added to the device dbdict.

a. Go to **Tailoring** – **Database Dictionary**

b. Select the **contacts** dbdict record

c. Click on the **descriptor** structure and click on **New Field / Key**

d. Name the field **journal.updates**

e. Select type **array**

f. Click **Add**

g. Select **character** for the array type

h. Click **Add**

Now the activity alias for the number field has to be created

a. In the descriptor structure, click on the **contact.name** field

b. Click on **Edit Field / Key**

c. Click on **Create Alias**

d. Change the name to **contact.name.activity**

e. Click on **OK** to create the alias

f. Click on **OK** to save and exit

Note: If you are prompted for an **ALTER TABLE** without having added fields other than the alias, choose USER ALTERS and don't do any additional action, since adding an alias does not modify the mapping and the ALTER TABLE is not necessary.

Next the viewing format will have to be changed. In our example, we will modify the contacts.g form. All other viewing formats that shall use activities will have to be modified the same way.

a. Go to **Tailoring** – **Forms Designer**

b. Select the form **contacts.g**

c. Go into **Design** Mode

d. Start a second Forms Designer Session by entering **fd** in the command line and pressing Enter

e. Select the form **PM.problem.planning** form

f. Go into **Design** Mode

g. Click on the **Activities** tab

h. Click **CTRL** + **C** to copy the activities tab

i. In the **contacts.g** form, click on the empty space next to the notebook tabs in the notebook to select the Notebook properties

j. Click **CTRL** + **V** to paste in the activities tab

k. Click **OK** twice to save the contacts.g form

l. Click **Cancel** twice to exit the PM.problem.planning form

m. Re-enter the **contacts.g** form to change it.

n. Go into **Design** mode

o. Select the new activities tab and move it behind the Contact Numbers tab (drag and drop)

p. Click on the **Activity Updates** sub-tab

q. Click on the **Activity Types** dropdown box

    i. Change the input to **$contacts.activity**

ii. Change the value list condition to **select("activity.name","activitytype","table","contacts","visible","YES")**

r. Click on the Current Update text box

i. Change the input to **$contacts.update**

s. Click on the Journal Updates sub-tab

i. Delete the visible condition and ensure the **visible** checkbox is checked

t. Click on the read-only text area and change the input to **journal.updates**

i. Delete the visible condition and ensure the **visible** checkbox is checked

u. Click on the Activity History sub-tab

i. Delete the visible condition and ensure the **visible** checkbox is checked

v. Click on the Filter by Activity Type dropdown

i. Change the input to **$G.contacts.activity.type**

ii. Change the value list condition (remove value list) to **select("activity.name","activitytype","table","contacts","visible","YES")**

w. Click on the virtual join area

i. Change the input to **contact.name.activity**

x. If you want to use the 'customer visible' flags for the activity records, you must add the flag (Customer Visible? – field name=cust.visible) to the Current Updates sub-tab and add the cust.visible Boolean field to the contacts dbdict as well.

y. Click **OK** twice to save and exit

With the new viewing format in place, we will now change the master link record contacts to allow for the activities to be displayed in the format.

b. Go to **Tailoring** – **Tailoring Tools** – **Links**

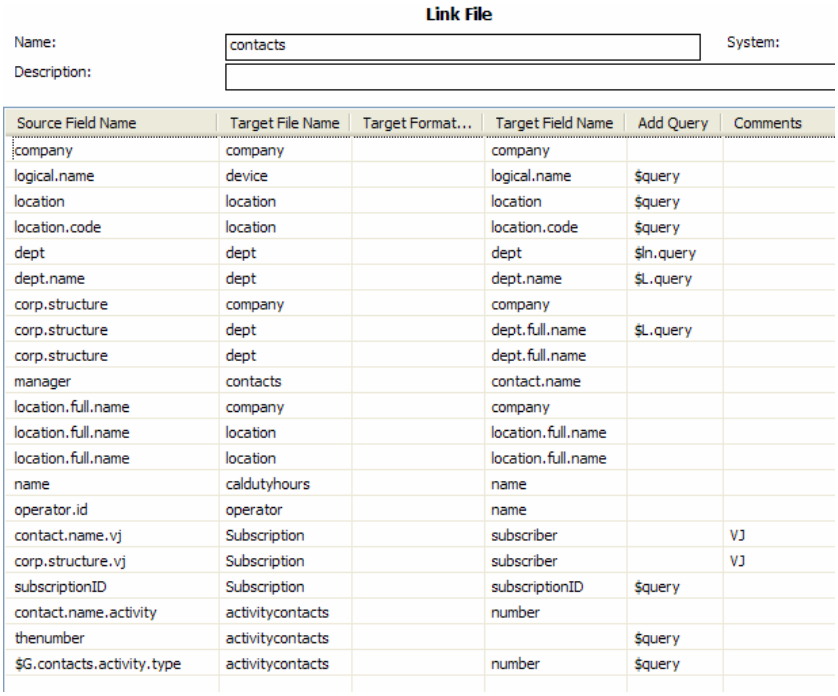
c. Select the **contacts** link

d. Add the following three lines to the end of the link

i. contact.name.activity – activitycontacts – – number

ii. thenumber – activitycontacts – – –$query

iii. $G.contacts.activity.type – activitycontacts – – number – $query



Link File

Name: contacts System:

Description:

Source Field Name	Target File Name	Target Format...	Target Field Name	Add Query	Comments
company	company		company		
logical.name	device		logical.name	$query	
location	location		location	$query	
location.code	location		location.code	$query	
dept	dept		dept	$ln.query	
dept.name	dept		dept.name	$L.query	
corp.structure	company		company		
corp.structure	dept		dept.full.name	$L.query	
corp.structure	dept		dept.full.name		
manager	contacts		contact.name		
location.full.name	company		company		
location.full.name	location		location.full.name		
location.full.name	location		location.full.name		
name	caldutyhours		name		
operator.id	operator		name		
contact.name.vj	Subscription		subscriber		VJ
corp.structure.vj	Subscription		subscriber		VJ
subscriptionID	Subscription		subscriptionID	$query	
contact.name.activity	activitycontacts		number		
thenumber	activitycontacts			$query	
$G.contacts.activity.type	activitycontacts		number	$query	

e. Right-click on the "thenumber" line and choose Select Line to edit the link line:

   i. In the Expressions set **$query="thenumber = \""+nullsub(cursor.field.contents(), "XXX")+"\""**

   ii. Set **$fill.skip=true**

   iii. Source Field: **thenumber**, Target Field: **thenumber**

f. Right-click on the "$G.contacts.activity.type" line and choose Select Line to edit the link line:

   i. $query="number = \""+contact.name.activity in $File+"\""

   ii. if (not null($G.contacts.activity.type)) then $query+=(" and type = \""+$G.contacts.activity.type+"\"")

   iii. cleanup($G.contacts.activity.type)

Step 7 – Changes to the activity setup files: activityactions, activitytype

First you will need to add data to the activitytype table that is appropriate for the table you are working with.

Note: The activity types that you enter here will have to be added to the value list for the globallist record: **Activity Types** if they do not yet exist.

a. Go to **Tailoring** – **Database Manager**

b. Select the file **activitytype**

c. Add the following records for contacts (Enter information, then click **Add** to add the record):

   i. Activity name: Contact Information Added, table: contacts, activity.number:1, visible: YES

ii. Activity name: Contact Information Updated, table: contacts, activity.number:2, visible: YES

iii. Activity name: Contact moved to new Department, table: contacts, activity.number:3, visible: YES

iv. Activity name: Contact moved to new Location, table: contacts, activity.number:4, visible: YES

v. Activity name: Contact Deactivated, table: contacts, activity.number:5, visible: YES

d. Go to **Tailoring** – **Database Manager**

e. Select the **globallists** table

f. Select the globallist with the name of **Activity Types**

g. Add

- Contact Information Added,
- Contact Information Updated
- Contact moved to new Department,
- Contact moved to new Location
- Contact Deactivated

to the list in alphabetical order as shown in the example below.

```
{"alert stage 1","alert stage 2","alert stage 3","Accepted",
"Analysis/Research","Assignment","Closed","Communication with
customer","Communication with vendor","DEADLINE ALERT","Contact
Information Added","Contact Information Updated","Contact moved to new
Department","Contact moved to new Location","Contact Deactivated","Dial-
in","Escalation","External Update","Incident reproduction","Not
Applicable","Open","Operator update","Partial Diagnosis","Pending
Change","Pending Customer","Pending other","Pending Vendor","Problem
Closure","Problem Identified","Problem Reproduction","Problem
Workaround","Referred","Rejected","Replaced
Problem","Reassignment","Referred","Reopen","research","Resolution","Reso
lved","RFA","RFS","RST","Site Visit","Status
Change","STU","Suspend","SVC","Unsubscribe","Unsuspended","Unsuspend
Alert","Update","Update from customer","Work In Progress"}
```

h. Ensure that the expiration time will occur in the near future or the immediate past

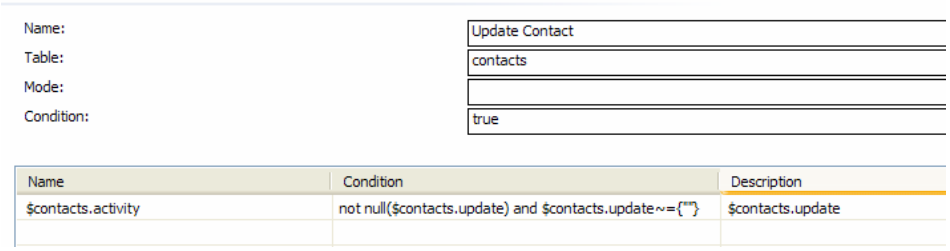
i. Click **OK** to save and exit

Now we will add appropriate data to the activityactions file. Activity actions are set in three areas: Create activities, update activities and close activities. The following example shows create and update activities that could apply to a contact record.

a. Go to **Tailoring** – **Database Manager**

b. Select file **activityactions**

c. Add the following records by entering the information pictured below and clicking **Add**



Step 8 – Changes to the display screens and -options

If you want to limit visibility of the activity tab and sub-tabs, you can set variables in the display screen to use on the forms. In our example, we have the activity tab visible under all circumstances. To limit this, do the following:

If you would like to control whether or not the Journal Updates tab is visible to the user, make sure that the visible condition on your Journal Updates tab is set to:

```
[$L.journal.contacts]=true
```

This variable can be set in the contacts.view.init Initial Process to the contacts.view State as follows (the capability word can be replaced by any custom-made capability word):

```
$L.journal.contacts=evaluate(index("SysAdmin", $lo.ucapex)>0)
```

If you would like to control whether or not the Historic Activities tab is visible to the user, you will need to make sure that the visible condition on your Journal Updates tab is set to:

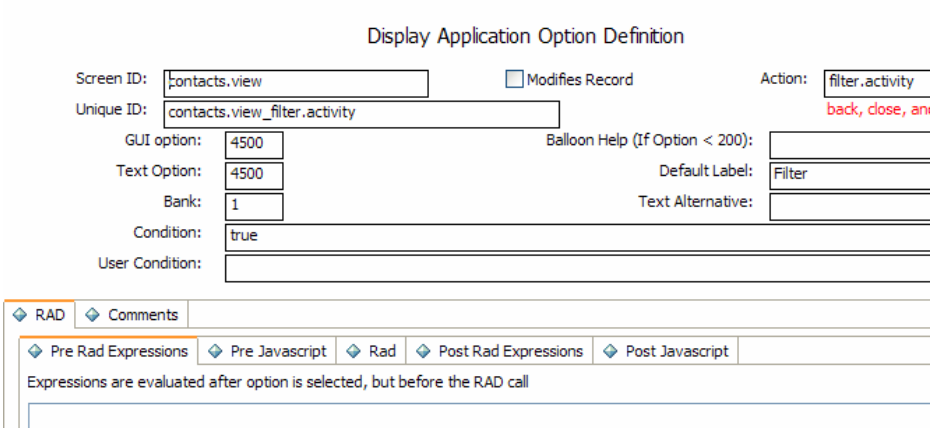
```
[$L.use.activity]=true
```

This variable can be set in the contacts.view.init Initial Process to the contacts.view State as follows:

```
if (not null(activitylog.file.name in $L.object)) then
($L.use.activity=true) else ($L.use.activity=false)
```

To enable the activity filter function from the activity history tab, the Process filter.activity will have to be called from the State record contacts.view with a corresponding display action from a display option that needs to be added for the contacts.view screen.

a. Go to **Tailoring** – **Document Engine** – **States**
b. Select the record **contacts.view**
c. Add a new line as follows:
    i. **filter.activity – filter.activity.contacts – true**
d. Click **OK** to save and exit
e. Go to **Tailoring** – **Document Engine** – **Processes**
f. Select the Process called **filter.activity.pm**
g. Change the name to **filter.activity.contacts**
h. Click **Add**
i. Replace all occurrences on all tabs of $G.pm.activity.type with **$G.contacts.activity.type**
j. Click **Save**
k. Click **OK** to save and exit
l. Go to **Tailoring** – **Tailoring Tools** – **Display Options**
m. Select the Display Options for **contacts.view**
n. Add a new Display Options with Button ID **4500** as shown below:



Step 9 – Ensuring the activities are written

There are two possibilities: Either the Process responsible for updating the record is calling se.base.method or it is calling its own application. If se.base.method is called, it will take care of the activity updates. If its own application is called, then the sc.activity RAD application will have to be called afterwards to add the activities.

To ensure that journaling works, the input field variable has to be initialized as an array. To do so, go to the initialization Process for your record and initialize the variable there.

a. Go to **Tailoring** – **Document Engine** – **States**

b. Select the update State record for update activities, in this case **contacts.view**

c. Go to the Initialization Process (Click Find if it does exist, if it does not exist, create a new one)

d. In the Initial Expressions, enter **$contacts.update = {}** (or the name of the input variable you used) to initialize the variable as an array.

e. Click **OK** to save and exit back to the State Definition record.

Enter the following initializations for each updating Process that is handled from the contacts.view State. For example:

a. Click on **Process Name** (i.e. contacts.do.save)

b. Click on Find to go to Process Definition record for this process

c. Enter the initializations on the Initial Expressions tab

Or, if the standard Processes are used, exchange $L.file with $L.filed and do these steps in the display option records that perform add, OK or Save.

```
if ($contacts.update={} or $contacts.update={""}) then
($contacts.update=NULL)
if (journal.updates in $L.file=NULL) then (journal.updates in
$L.file={""})
if ($G.bg and not null($G.bg.activity.type)) then
($contacts.update=nullsub($contacts.update,
$G.bg.activity.text);$contacts.update=nullsub($contacts.update, "External
Update");$contacts.activity=nullsub($G.bg.activity.type, "External
Update"))
```

Note: If using se.base.method to call the activity routines (RAD=sc.activity.update.check and/or RAD=sc.activity), you have the option of passing in, for the 2nd parameter, "update.no.activity", if it is desired to bypass the activity processing, but you still want to perform an update

If a RAD application other than se.base.method is used to update the record, add the following in the Process record's RAD tab after the call to the updating RAD application. If there are many Processes that update the record, it is recommended to create a new Process to create the activity record and call that new Process from the updating Processes.

Expressions before RAD (these expressions have to be entered before the call to the RAD application that writes the update to the record):

```
$L.update.save= journal.updates in $L.file; $L.stamp=str(tod())+"
("+operator()+"):"
if ($L.mode~="add" and nullsub($L.journal.contacts, true) and
lng(denull($contacts.update))>0 and denull($contacts.update)~={""}) then
(journal.updates in $L.file=nullsub(denull($contacts.update),
{""})+denull(journal.updates in $L.file); journal.updates in
$L.file=insert(journal.updates in $L.file, 1, 1, $L.stamp))
```

RAD Application: **sc.activity**

Condition: **$L.mode="update"**

Parameter Name	Parameter Value
file	$L.file
second.file	$L.file.save
text	$L.action
boolean1	true (or cust.visible in $L.file if customer visible flag is used)
record	$L.object

Expressions after RAD call:

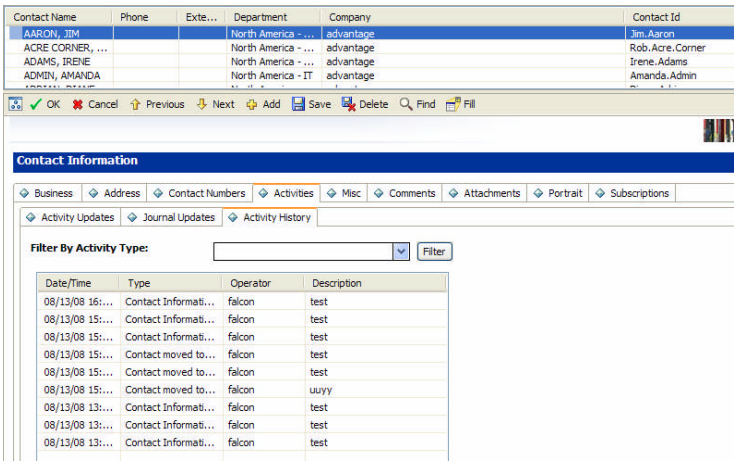
```
cleanup($contacts.update);cleanup($contacts.activity)
```

Step 10 – Test

Go to System Administration – Base System Configuration – Contacts.

Bring up any record and add a new activity.

Test different activity types, test the activity filter, and check the journal updates.



For more information

Please visit the HP OpenView support web site at:
http://www.hp.com/managementsoftware/support

This web site provides contact information and details about the products, services, and support that HP OpenView offers.

HP OpenView online software support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valuable support customer, you can benefit by being able to:

- Search for knowledge documents of interest
- Submit and track progress on support cases
- Submit enhancement requests online
- Download software patches
- Manage a support contract
- Look up HP support contacts
- Review information about available services
- Enter discussions with other software customers
- Research and register for software training

Note: Most of the support areas require that you register as an HP Passport user and sign in. Many also require an active support contract.

To find more information about support access levels, go to the following URL:

http://www.hp.com/managementsoftware/access_level

To register for an HP Passport ID, go to the following URL:

http://www.managementsoftware.hp.com/passport-registration.html

08/2008