



vWorkFlow Manager v7.0.0

User and Admin Manual

Version: v1
Date: 16/11/2023

1	Change Control	5
2	Introduction	6
3	Compatibility and prerequisites	8
3.1	COMPATIBILITY	8
3.1.1	CRM	8
3.1.2	Browsers	8
3.2	PREREQUISITES	8
4	Installation	9
4.1	FIRST TIME INSTALL	9
4.2	UPDATE TO NEW VERSION	10
4.2.1	<i>New version of ValeDale Common Base</i>	11
4.3	ACTIVATION vWFM	11
4.3.1	<i>Schedulers</i>	11
4.3.2	<i>Management of logic hooks</i>	12
5	Basic concepts	14
5.1	PROCESS ENTITIES	14
5.2	PERSISTANCE IN DATABASE	14
5.3	COMMUNICATION BETWEEN vWFM INSTANCES	14
5.4	A PROCESS	15
6	Quickstart: Intro to creating a workflow	16
6.1	LISTVIEW	16
6.2	WORKFLOW EDITOR (WFE)	18
6.2.1	<i>Navigating</i>	19
6.2.2	<i>WF Editor : Symbols and their meaning</i>	22
7	Creating a Workflow: in detail.	23
7.1	WORKFLOW DESIGN	23
7.1.1	<i>Workflow instances</i>	23
7.1.2	<i>Triggers based on manipulation of data</i>	24
7.1.3	<i>Scheduled events</i>	24
7.1.4	<i>Activities with delay</i>	24
7.1.5	<i>Task execution</i>	25
7.1.6	<i>Task delays</i>	25
7.1.7	<i>Intermediate events</i>	25
7.2	vWFM-PROCESS	26
7.3	vWFM-EVENT	29
7.3.1	<i>Event type: Logic Hook</i>	30
7.3.2	<i>Event type: scheduled</i>	34
7.3.3	<i>Event type: Subprocess</i>	36
7.3.4	<i>Event type: local Subprocess</i>	37
7.4	vWFM ACTIVITY	38
7.5	vWFM-TASK	40
7.5.1	<i>Form tasks (not yet released)</i>	47
7.6	TASK DEFINITION EXAMPLES	48
7.6.1	<i>Send Email</i>	48
7.6.2	<i>PHP custom</i>	48
7.6.3	<i>vWFM-Task task_type=continue</i>	48
7.6.4	<i>End</i>	49
7.6.5	<i>Create Object</i>	49
7.6.6	<i>Modify Object</i>	49

7.6.7	Call Process	49
7.6.8	Add Custom Variables	50
7.6.9	Forms error message	50
7.6.10	Forms response	50
8	vWFM Variables	51
8.1	TYPES OF vWFM-VARIABLES	51
8.1.1	Module-data Variables	51
8.1.1.1	Main data (bean)	51
8.1.1.2	Email related variables	51
8.1.1.3	Data in related modules	52
8.1.2	WorkFlow related data	53
8.1.3	User-defined vWFM-Variables	54
8.1.4	Variable inheritance (in subprocess)	54
8.2	vWFM-VARIABLES USAGE	54
8.3	MODIFYING GLOBAL VARIABLES	55
8.4	VARIABLE GENERATOR	56
9	vWFM & module Forms (not released yet)	58
10	Exporting and Importing WorkFlows	60
10.1	EXPORT WORKFLOWS	60
10.2	IMPORT WORKFLOWS	61
10.2.1	Import WorkFlows without context	61
10.2.2	Import WorkFlows in context	62
10.2.3	Import WorkFlows Advanced	62
11	Bundled WorkFlows (examples)	64
11.1	LEAD ASSIGNMENT WITH VIEWS	64
11.2	LOGIN AUDIT	65
11.3	KEEP TRACK OF LAST TIME A CONTACT WAS APPROACHED	68
11.4	TASK ASSIGNMENT NOTIFICATION	70
11.5	SEND OPPORTUNITIES CHANGE LOG IN EMAILS	71
11.6	LEAD ASSIGNMENT	73
12	Admin tools	76
12.1	VALIDATE CONFIGURED FEATURES	76
12.2	MODULE DEPENDENCIES	76
12.3	cURL	77
12.4	FILES ACCESS	77
12.5	vWFM SCHEDULERS	77
12.5.1.1	vWFM_scheduled_task	77
12.5.1.2	vWFM_engine_crontab	77
12.6	CONFIG FEATURES	77
12.7	vWFM DATABASE-CLEANUP	79
12.8	vWFM MONITOR	81
12.8.1	Process Instances	81
12.8.2	Working Nodes	81
12.8.3	On Hold	85
12.9	vWFM REBUILD-LOGIC_HOOKS	86
12.10	VALIDATE WFs	87
12.11	vWFM-EVENT DUPLICITY	88
13	Known Issues	89



1 Change Control

Date	Version	Description	Author
10/05/2023	7.0.0	- Support for MSSQL	jfcc
11/01/2023	6.2.0	<ul style="list-style-type: none"> - Schedulers run now as jobs (improved efficiency) and easier to manage. - New WorkFlow example with provisioning (requires at least run time version of Forms&Views). - Used SuiteCRM convention for adding Entrypoints (this will make WFM better upgrade safe) - Login Audit List-view now shows last entry first. - Login Audit Work-flow did not work for login-failed. - Now no logic-hook is automatically activated (can be done manually in Admin section) - Clash in css styles made some buttons in task-tab not to appear. - Something went wrong with the positioning (information pop-up, in edit mode) (dialogue is pushed too high) when the information button is low on the screen. - In task custom variables, the "is global" mark was not generated correctly for each row. - For WFM with Initialize node, wrong check was made to see if initialize node was up and running (values could not be easily reset) - #5 Various look&feel enhancements 	jfcc
11/03/2022	6.1.0	Adapted self-referencing fix for SuiteCRM 7.11 PHP 7.4 supported	jfcc
24/08/2021	6.0.0	<p>Resolved crontab issues.</p> <p>Resolved SugarJob queue issues for processes.</p> <p>Improved examples.</p> <p>Local Subprocesses appear now when selecting subprocess in tasks.</p> <p>Solved issue when cloning from workspace.</p> <p>Basic import of call_process task did not update process_id and process_event.</p>	jfcc

2 **Introduction**

This document describes how to use Valedale's WorkFlow Manager (ValeDale WFM) module for SuiteCRM. The target audience are technically oriented users, with a certain knowledge of process definition, databases and at least a basic knowledge of PHP. The user also must have admin rights.

ValeDale WFM is part of the overall ValeDale initiative. "Vale Dale", Spanish expression for "OK GO" and pronounced as "BalleDalle", has the goal to create "out of the box thinking" solutions that will help people in their thought process pushing the limit from providing information to providing suggested actions and thus adding real tangible value.

ValeDale WFM, or in short: "vWFM", builds on top of the latest published Alineasol's WFM package, with the goal to evolve this in line with ValeDale's vision, while building on top of the expertise of people that have been involved in the creation of Alineasol's packages and are now part of ValeDale.

We plan to evolve vWFM specifically in the following areas:

- Usability
- Speed
- Integration with other packages

ValeDale WFM has been designed for SuiteCRM. As with all the ValeDale CRM modules, MySQL is required as database.

The goal of ValeDale WFM is to provide a powerful tool to set-up automated workflow processes within SuiteCRM.

Functionally, ValeDale WFM offers the possibility to execute certain automated tasks when events occur within the CRM. A basic example would be to send a notification (email) to a group of users when e.g. a new deal has been won. However, ValeDale vWFM supports considerably more complex scenarios. Conditional actions, multi-step checks, sub-processes, parallel execution, background processes, scheduled processes, global variables, use of external databases, etc. are all supported.

Of course, a graphical interface is provided so you can define your WorkFlow: The vWFM Editor.

With vWFM, you can define your own workflows in order to control most of SuiteCRM events. You can control the creation and modification of SuiteCRM objects (Accounts, Opportunities, etc). Also, you can trigger processes based on scheduled events. These processes can regularly scan your CRM database and based on certain values that you define yourself, take action.

You can also insert variables when sending emails (e.g. name of a contact), define global variables for your process instances, create tasks when certain conditions apply, define follow-up actions when tasks are closed, etc.

You can also define new tasks by using the PHP custom task.



The best way to start using vWFM is to check out the workflow examples that are bundled. Check out chapter 11 this manual. Also we recommend to read the documentation for each example to better understand its workings. You can then use these examples as templates.

3 Compatibility and prerequisites

3.1 Compatibility

3.1.1 CRM

- SuiteCRM latest LTS version

3.1.2 Browsers

- Chrome (recommended)

Should also work on:

- Firefox
- Microsoft Edge
- Opera
- Safari

3.2 Prerequisites

- MySQL database
- ValeDale Common Database module
- CRM patch module (also from ValeDale): `fix_Suitecrm_self_referencing`

4 Installation

4.1 First time install

If you have never used vWFM in your current CRM instance:

Ensure that you have already installed:

- **ValeDale Common module**
- **Self-referencing fix module**

If not:

- 1) Download ValeDale Common Base module zip to your PC
 - This module will have a name like: "ValeDale_Common_v1.0.3.zip"
- 2) Download self-referencing fix
 - This module will have a name like: "_fix_sugarcrm711_module_self_referencing_v7"
- 3) Login as a SuiteCRM admin user
- 4) Goto "Admin" section
- 5) Select "Module Loader"
- 6) For both the common base module and then the fix module:
 - a. Select the corresponding zip file and upload
 - b. Install the module
 - c. Confirm data

After that, you can install vWFM:

- 7) Download ValeDale WFM module zip to your PC
- 8) Login as a SuiteCRM admin user
- 1) Goto "Admin" section
- 2) Select "Module Loader"
- 3) Select the zip file of "ValeDale WFM", and upload
- 4) Install the module
- 5) Confirm data
- 6) Perform a Quick Repair & Rebuild. (Under admin, Repair)
- 7) If you experience issues with dropdown menus, also perform a "Repair Javascript Languages" (Under admin, Repair)

vWFM will modify the following files:

- “[Suitecrm_installation_path]/include/MVC/Controller/entry_point_registry.php” in order to add the entry_points used by vWFM. These are:
 - WFM_engine
 - WFM_export_workflows
 - WFM_layout
 - WFM_flowChart
 - WFM_flowChartActions
 - WFM_scheduled_task
 - WFM_variable_generator
- “[Suitecrm_installation_path]/custom/modules/[module]/logic_hooks.php” in order to tell vWFM what modules you want vWFM to listen to. The text added looks like this:

```
$hook_array['after_save'][] = Array(2, "WFM_hook", "custom/include/WFM_hook.php", "WFM_hook_process", "execute_process");
```

At installation NO hooks will be “activated”. This is done in order to optimize performance so that you only activate those hooks that you need. You can define what hooks will trigger vWFM in the WFM admin section.

4.2 Update to new version

First, you need to uninstall vWFM.

There are two options:

- a) Remove Tables
 - You will loose your workflows.
- b) Do Not Remove Tables
 - Your workflows remain in

database.

Recommended procedure:

- 1) Uninstall older version but **KEEP Tables**.
- 2) Install new version of vWFM
- 3) You may have to update *config_override.php* with your previous changes.
- 4) Perform a Quick Repair & Rebuild.
- 5) If you experience issues with dropdown menus, also perform a “Repair Javascript Languages”

(Under Repair)

Not recommended but should function: You can install a new version of vWFM without de-installing the new version. Your config data will be retained.

Between a vWFM version and another one, vWFM database table structure may have changed. If you do not remove tables and install a new version of vWFM then a database upgrade script will be executed. During uninstall, the system will clean-up previous entries (entry_point_registry.php, etc) .

4.2.1 New version of ValeDale Common Base

If you need to upgrade to a new version of ValeDale Common:

- 1) Uninstall first vWFM (keeping DB)
- 2) Uninstall Common Base
- 3) Install New Common Base
- 4) Install new vWFM

You can keep the fix for the relations.

4.3 Activation vWFM

Before you can use vWFM, you have to ensure that:

- schedulers are correctly set-up
- The logic hooks for the corresponding modules are activated
- Ensure that CRON is set-up (see your CRM installation manual)

4.3.1 Schedulers

You have to define 2 schedulers for vWFM to work properly. This is now based on jobs (not on URL calls as in older versions).



Scheduler	Interval	Range	Status
<input type="checkbox"/> new_wfm_scheduled_task	As often as possible.	01-01-2005 20:00 - perpetual	Active
<input type="checkbox"/> new_wfm_engine_crontab	As often as possible.	01-01-2005 20:00 - perpetual	Active

The **new_wfm_scheduled_task** scheduler is used to execute the scheduled actions defined in the workflows. You have to assign the job “vWFM Scheduled Tasks” as shown below.

CREATE

SAVE CANCEL

BASIC

Job Name: Status:

Job: Job URL:

Advanced Options:

Interval:

Every Day
 Monday
 Tuesday
 Wednesday
 Thursday
 Friday
 Saturday
 Sunday

The `new_wfm_engine_crontab` is used to execute any workflow action that is pending not due to a scheduled task, for example for asynchronous execution of some logic. Here you have to assign the job: “vWFM Pending Tasks”

Both schedulers should be set to be executed “as often as possible”, which in practice will mean every minute.

If you do not regularly set-up crontab, here are some indications:

Linux

Open a terminal:

- Type: `vi /etc/crontab` (or use your favorite editor)
- Add: `***** nobody cd /opt/lampp/htdocs/[instance name] && /opt/lampp/bin/php -c /opt/lampp/etc/php.ini -f cron.php > /dev/null 2>&1`

(for bluehost, cron.php might avoid non-cli calls):

`***** php-cli /home4/{username}/public_html/{subdomain}/cron.php`

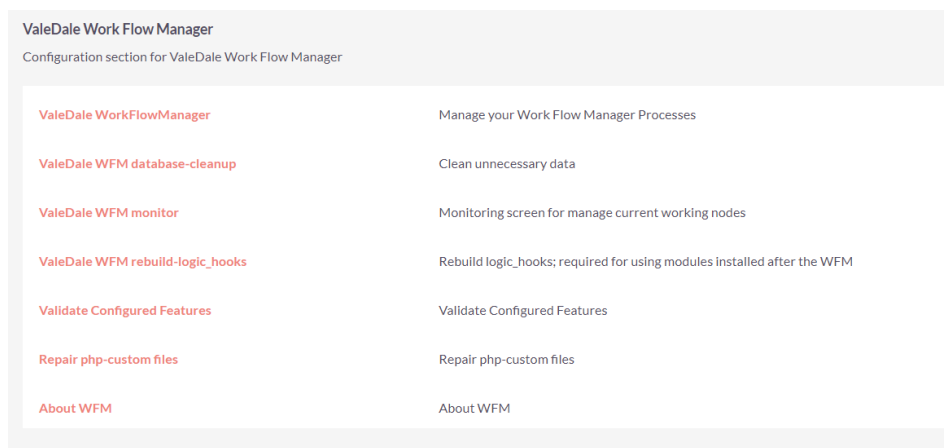
Windows

- Create a batch file to run using Windows Scheduled Tasks. The batch file should include the following commands:

```
o cd C:\[your directory]\htdocs\[instance_name] php.exe -f cron.php
```

4.3.2 Management of logic hooks

In the admin section, under ValeDale Work Flow Manager, you will find an entry to rebuild logic hooks.



Check the logic_hooks you want ValeDale V

action=execute_process (Required in order to use workflows with wfm-events of trigger_type=logic_hook)

<input type="checkbox"/>	Module	before_save	after_save	before_delete
<input checked="" type="checkbox"/>	Accounts	✘	✘	✘
<input type="checkbox"/>	Alerts	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	Projects - Templates	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	Project Task Templates	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	Index	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	Index Event	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	Knowledge Base	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	KB - Categories	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	Case Events	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	Case Updates	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	Reports	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	Scheduled Reports	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	Contracts	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

You have to cross (cross = active) the modules that you want to supervise.

NOTE: you have to select before_save also, if you want the after_save logic to work properly. You can select the three logic hooks at the same time by clicking on the entry box in the first column.

Make sure that you hit "Update" at the bottom of the page for the changes to apply.

5 Basic concepts

ValeDale WFM is inspired by BPMN (see: <http://www.bpmn.org/>), however, taking into consideration SuiteCRM architecture and typical user’s needs, a more simplified model was chosen.

To understand vWFM, let’s start with the concepts.

5.1 Process entities

ValeDale WFM introduces the following entities:

Process	A vWFM-process is the definition of an automated workflow. We will use here the terms Process and Workflows as the same concept.
Event	A vWFM-event triggers the execution of (part of) a vWFM-process.
Activity	A vWFM-process consist of vWFM-activities that are performed as part of the defined workflow.
Task	Every activity, consists of 1 or more vWFM-tasks. Tasks are executed in the same order as defined.

Note: In this document, a “vWFM-task” will be related to “vWFM-process's tasks”. “SuiteCRM tasks” will be used when referring to the standard SuiteCRM's built-in tasks.

Events and Activities are Flow-control components. All “do” things are defined as WFM-tasks.

5.2 Persistence in Database

In order to execute your workflows vWFM uses its vWFM-engine. The vWFM-engine is a state machine. To persist data, the status of processes is stored in the Database.

WFM-ProcessInstances	When a vWFM-process is instantiated (by the trigger of a vWFM-event), a vWFM-process_instance is created.
WFM-WorkingNodes	A vWFM-working_node is a pointer to vWFM-activity that is being executed. For each branch of activities defined, one working-node will be created.
WFM-OnHold	This table holds the context of a Process-branch where you want part of the process to be executed after a previously defined CRM task is closed or call is held. The specific conditions for this to apply are: encountering a task where delay_type=wait_on_finish_previous_task, preceded by a create task/call.

5.3 Communication between vWFM instances

Communication between vWFM-process_instances can take place in several ways:

- A vWFM-process can manipulate an object which may result indirectly in another vWFM- process being triggered
- A vWFM-process can call a sub-process.

Also, you can define intermediate events that can affect the execution within a given workflow. See also chapter 7.1.

5.4 A process

Below you can see how a process is depicted in the WorkFlow Editor.

You can see here a process that is being triggered by a logic hook (hence the hook-icon in the round event), followed by 3 activities, each consisting of various tasks. Note that normally the names of the elements will be more meaningful...



We normally will use the names WorkFlow and Process as being equivalent. Strictly speaking, a Process could consist of multiple Workflows, but currently, vWFM does not support the concept of bundled WorkFlows.

6 Quickstart: Intro to creating a workflow

The best way to create a Workflow is to start with an existing one. vWFM comes bundled with a number of useful Processes. You can find them in the zip file, in the folder “WF examples”.

We would suggest that you import the workflows to analyse them and use as a base for your wn Workflows. Once you import the workflows, you will see them in the “ListView” (note that the actual WFs available may be different than what the next picture shows).

6.1 ListView

Let’s go first through the information presented in ListView.

WFM PROCESS

Name	Status	Async	Data source	Form	Use Database	Module	Audit	Date Created	Date Modified	Created By	Modified By
Send Change log in an email	Inactive	Asynchronous - SugarCRM job queue	Database		CRM Native Database	Opportunities		14/05/2021 18:22	17/05/2021 16:47	admin	admin
Lead assignment	Active	Synchronous	Database		CRM Native Database	Leads		10/05/2021 16:28	14/05/2021 15:45	admin	admin
Task assignment notification	Inactive	Synchronous	Database		CRM Native Database	Tasks		10/05/2021 00:18	14/05/2021 18:25	admin	admin
Calls update related Contacts if conditions apply	Inactive	Synchronous	Database		CRM Native Database	Calls		06/05/2021 10:28	13/05/2021 13:12	admin	admin
WFM Login Audit	Active	Synchronous	Database		CRM Native Database	Users		06/05/2021 00:32	13/05/2021 16:05	admin	admin

View WFs Validate WFs Activate WFs Inactivate WFs Export WFs Import WFs Import WFs Advanced Delete WFs WFM Variable Generator

Note: “select all” does currently not work for mass operations. Select all in page, does work.

Going from left to right:

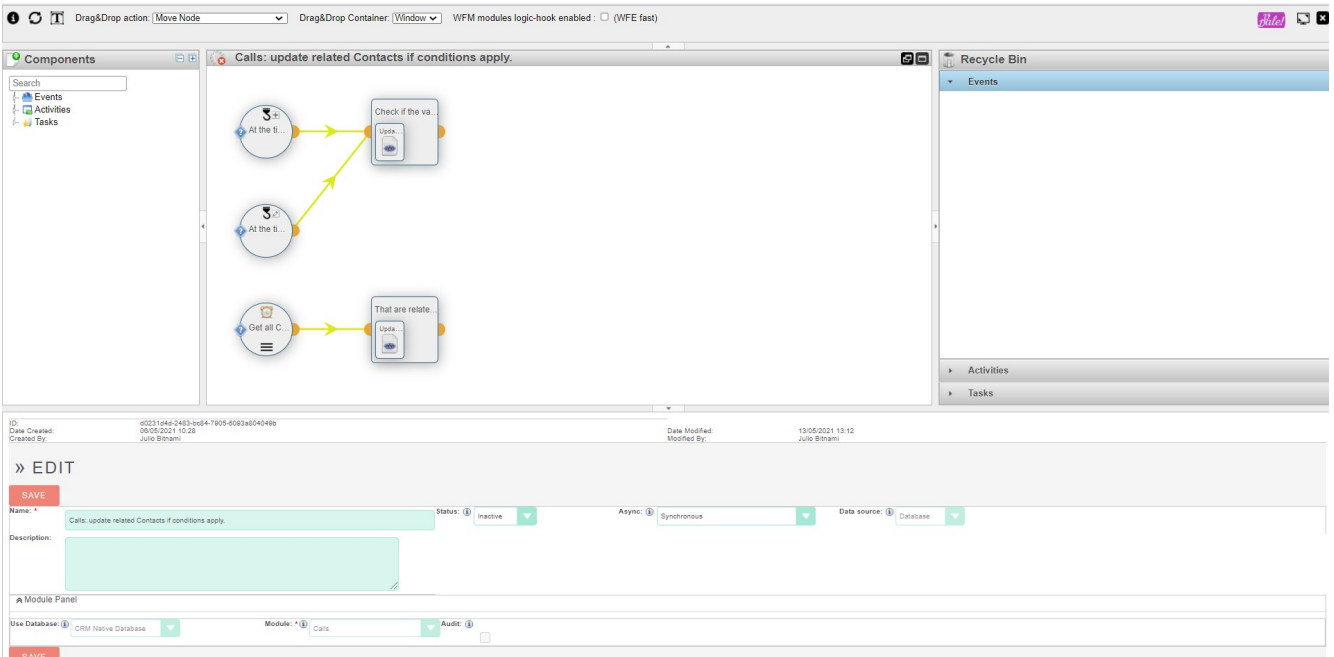
- Hot action buttons:
 - o Edit WorkFlow
 - o Launch WFE editor: launches a new window dedicated to editing the WorkFlow.
 - o Validate Workflow: this is a mini-process that will lead you through a basic check on your Workflow. It can check on:
 - Potential email templates that you use actually exist.
 - That your Workflow is active (else, it will do nothing)
 - That the logic hook for the given module is active
 - Note that currently validation does not check if you have only defined a scheduled event, in which case the logic hook does not have to be active for the WF to work.
- Name: name of your Workflow. If you click on it, you will launch the WF Editor in the same window.
- Control button: When in status Inactive, you will see the play button. Clicking on it will activate the Workflow. When in status Active, you will see the stop button to De-activate the WorkFlow.
- Status
- Async: The two main options are “Synchronous” and “Asynchronous”. The main difference is that Synchronous will execute your workflow immediately when the Events conditions apply (if you have selected logic hook as the trigger for the event) and Asynchronous will, in that same case, let you go on with your work while the flow will be executed in the background.
- Datasource: Can be Form (from ValeDale Forms&Views) or Database.
- Form: will show the name of the Form if Selected in previous item.
- Database: Can be the CRM Native Database, or if you have configured other databases, you can select those ones.
- Module: In case of CRM Native Database: the name of the selected module.
- Audit: will be marked if you have opted to work with the Audit table of the given module.
- Date Created
- Date Modified
- Created by
- Modified by
- Special action buttons:

- o Export WF: to share your WF with others (always good to also keep a back-up)
- o Import WF: to import Workflows. Note that they will be set to "inactive" at standard import. For security reasons...
- o Delete WF: Bye bye WorkFlow.

So, now that we know what the main actions are, you can click on a name of the WorkFlow to launch the WorkFlow Editor.

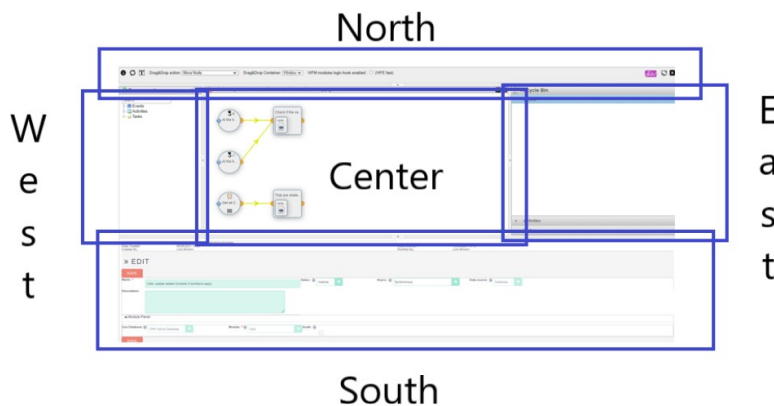
6.2 WorkFlow Editor (WFE)

The WFE allows you to create/view/modify workflows by means of dragging and dropping vWFM-components.



Your first impression could be that this is complex... But there is some order to it, and with some practice, this “all in one” view will shine.

There are 5 sections in the WFE:

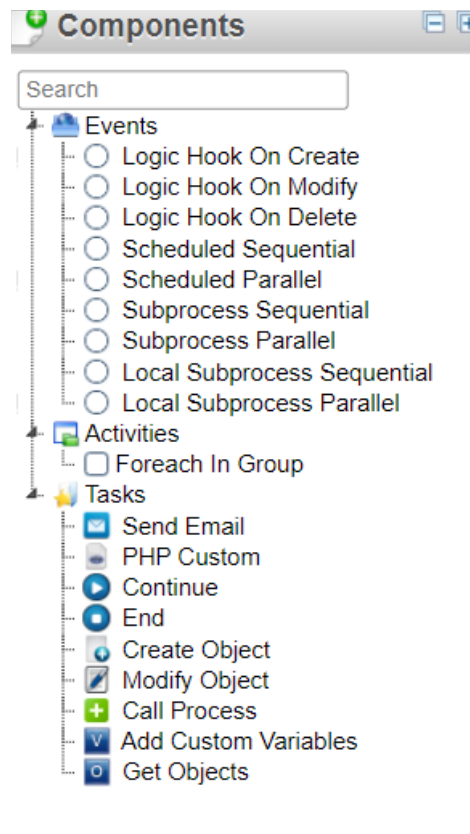


1. North panel: WFE's control panel.
2. South panel: object editor (modify vWFM-entities).

3. East panel: recycle bin (vWFM-entities stored here will not be executed).
4. West panel: components (create vWFM-entities, drag from west panel to center panel).
5. Center panel: workflow.

6.2.1 Navigating

In the Component panel you can find the elements that you can drop in the center panel, so that you can “draw” your workflow.



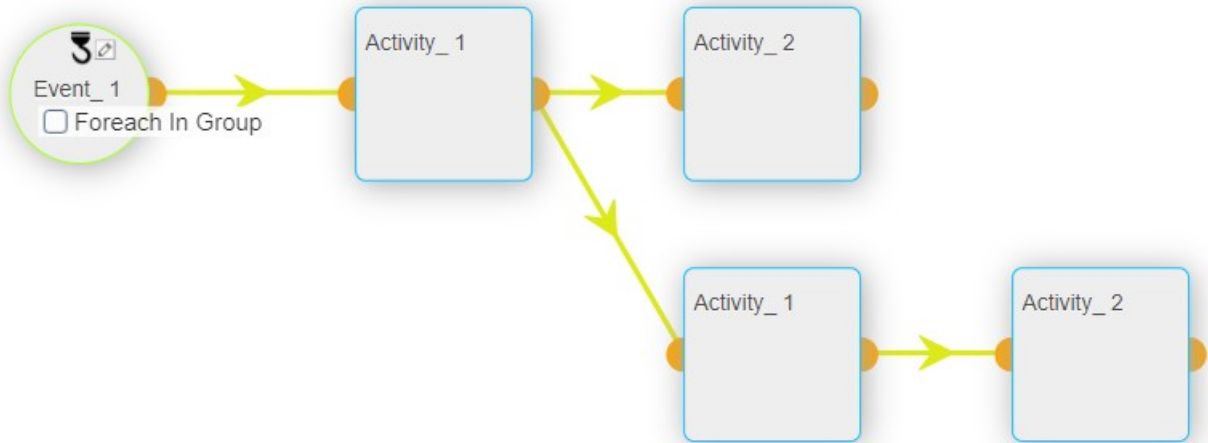
Any change you make in the Workspace is automatically saved ! (does not apply to the South Panel)

In the header section of the component panel you can see the compress (-) and expand (+) icons. You can select the expand to see all the elements, and the compress, to just see the categories. In the search box you can type part of the name of an element for filtering (it does not have to be the start of the name).

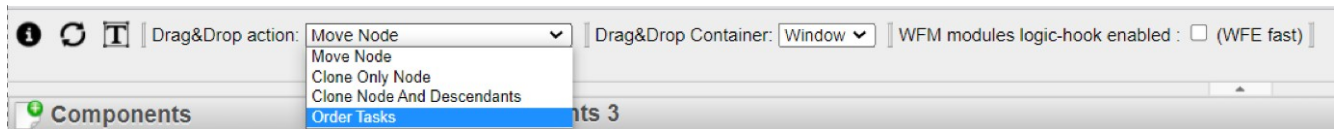
You can pick and drop any event into the workspace. You will notice that they will be ordered/grouped according to the type of event.

For activities, you can drop them on top of the element that you want to connect it to. As you drag them into the workspace, you will see that the elements already placed, get a blue (not connect) or

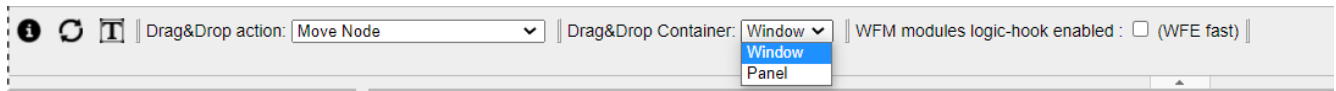
green border (you can drop it now to be connected to this element). You can replace an activity following the same principle. Note that everything after that activity will be moved along, i.e. you will be moving the whole branch that follows the activity.



Something similar applies to tasks. You can drop them within any activity. You can pick also existing tasks and move them to other activities. Note that if you want to reorder tasks within an activity, you have to change the “Drag&Drop action” from “Move node” to “Order Tasks”. This you can do in the control panel (North one).

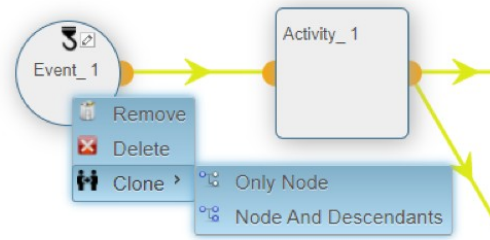


When you work with larger Workflows, you may want to change the Drag&Drop Container to “Panel”. The panel will then scroll along as you move an element.



You can use the right mouse button, when selecting an element to

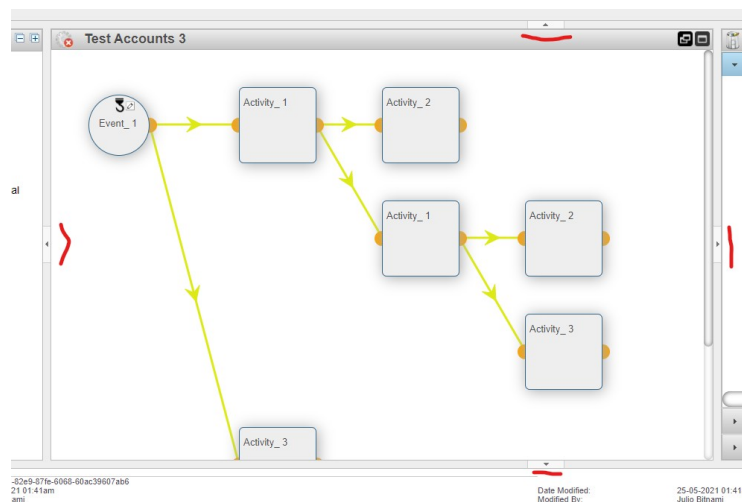
- Remove (sends element to the recycle Bin)
- Delete (bye bye element)
- Clone the element (copy and paste).



When you clone elements, they will appear in the recycle bin. So from there you can connect them where you want (you can quickly make impressive, non-functional Workflows this way ! ;-)

You can directly drag elements into the recycle bin for potential future use, and drag them back!

You can resize the panels to your heart’s desire. Note that you can also click in the centre part of the division line of the panels to quickly make more room for the centre panel.






You will notice though that most of the work will be done in the Workspace Panel and the Object editor (South Panel). When you click on the name of an element (from the Workspace), the data for that object will be loaded in the South Panel (**make sure to save your previous object before reloading a new one**). There are also some shortcuts to get more info on the object.

Shortcuts:

- If you hover the cursor over the component-items (icons and element-names) you will get extra info.
- Click on the condition-icon (question-mark) to get a summary of the conditions that apply.
- In the control panel, if you click on the “T” icon, you can switch between abbreviated and complete names (does not apply to tasks for visibility reasons)

6.2.2 WF Editor : Symbols and their meaning

	<p>conditions</p>
	<p>delay</p>
	<p>open subprocess (in call process task)</p> <p>When you click on this icon, the corresponding event of the sub-process will appear in the South-Panel. In this way you can easily read the illustrative description that for sure you have added to that event 😊 .</p>

7 Creating a Workflow: in detail.

Now that we know the basics of vWFM, we can dive into the details. Most of this chapter applies to the Object Editor (South Panel). Previous sections explained how to navigate around the full graphical editor.

In this chapter we will explain, amongst other “good to know”, the building blocks that you have at your disposal to create full-fledged automated processes.

First a few recommendations.

Check out the workflows bundled with vWFM. Basic (and not so basic) concepts are illustrated there, and probably you can use them as the basis for your developments.

We recommend that you build and test your new Workflows always first in a test environment. vWFM is a powerful tool, so you can also generate “powerful” unwanted behaviours. Wrong flows can result in garbage being created, performance impacts, unintended manipulation of objects, etc. Once you have tested the Workflows in a test environment, you can export them and import them into the production environment.

As in every lifecycle, once you create workflows you will want to enhance them at some moment in time. Please note that in general it is not best practice to make changes to active Workflows. It will depend on the actual Workflow and the changes that you want to perform to determine if this is a “must”, “preferred”, or “no prob”. Even if you deactivate the Workflow, take into account:

- Do other workflows depend on this one?
- Are there still actions to be processed (parts of the Workflow may be on hold, waiting for an event to happen: they still will execute, independent of the status of the Workflow).

You might consider going one step further and cloning the Workflow, perform the changes, and then deactivate the old version and activate the new one. This also can have implications, so you have to analyse if this is a better or worse solution. Concerning cloning: you can perform this by doing an export and then import it back.

7.1 Workflow design

vWFM can run your workflows in different ways. This (also) important section explains the fundamentals and will help you make design decisions and analyse how these will impact the user perceived behaviour (from “not acceptable” to “perfect”).

7.1.1 Workflow instances

First, let’s clarify that you can, of course, have multiple Workflows related to, for example, a modification in the Accounts module. This is logical, as you don’t want to create Mega-processes that will do everything (issues with maintenance, administration, etc.).

Therefore, you will define normally multiple workflows to automate your processes. Workflows are by definition independent, unless you explicitly or implicitly link them. See also section 5.3.

Every Workflow will have one or multiple Events defined (or else nothing could happen!). You can consider these “inputs” or “triggers”. When an event fires (i.e. conditions for the event apply), vWFM will take action:

- If the event is a “Start” event, a new “Workflow instance” is created.

For the technically oriented: these process instances do not result in separate (PHP) threads. However, they do have their own state, own variables, etc.

You may think that there are certain situations in which you would like to have global data to share between different workflow instances (of the same workflow). And you would be right! ☺ See in the Events section, “Initialize Event”.

“Intermediate events” will not result in the creation of new Workflow instances. See also events section.

Understanding the concept of WorkFlow Instances is important as you will have to take this into account when you design your datamodel (applicable only for more complex processes).

7.1.2 Triggers based on manipulation of data

Processes can run in Synchronous or Asynchronous modes. Synchronous means that it will run “in real time”. For example, if you change a value manually in an opportunity and you want another field of the opportunity to be automatically recalculated, you will want the changes to take place before you see the saved data back on screen (else it would be very confusing...).

Note: if you use the “edit field” option in your CRM, only the manually changed field will be refreshed on screen. Other fields, although they may have changed too, will not appear updated.

In Asynchronous mode, processes will not run in real time, but with a certain delay. For example, you may want to send a notification to someone when certain conditions apply because of a change. WFM supports two asynchronous modes:

- a) Using Curl: you need to have Curl support in your system.
- b) Using the SuiteCRM job-queue

The Curl option will launch a thread in the background to execute the flow. However, there will be a delay of one second (Curl idiosyncrasies) every time it is launched. This option is not appropriate for batch processes as every item will introduce a 1s delay in execution time.

The SuiteCRM job-queue option does not introduce a delay in execution time. The system will add the workflow to the SuiteCRM job-queue and it will be executed at the next Cron-batch. Normally, Cron should be set at 1 minute repeat intervals, so there will be a delay for the execution to start, but the user can continue to work without delay in parallel. This option will result though in potentially many entries in the jobs-queue. It has its pros and cons.

Note that by default the job-queue option is disabled (ou can still pick the mode at process definition, but the process will not run) This is done for a slight performance increase. Check out section 12.6, to enable this option (and if you do so, you might want to disable the CURL alternative). See parameters: WFM_enable_async_sugar_job_queue, WFM_disable_async_curl

7.1.3 Scheduled events

You can define events that will run based on some recurring time trigger, e.g. every day early in the morning. You may want to use these for clean-up actions, data preparation, etc.

7.1.4 Activities with delay

You may define a delay for an activity to execute. You can do this to:

- Allow for a certain action (it can be an action outside the flow, e.g. give a user some time to think about a potential issue) to take place.

- Execute an activity by default (if something does not happen earlier). You can “Cancel” the default action with a forced termination task
 - o An example would be to send a reminder as default, but “cancel” this if a task is closed.

7.1.5 Task execution

Tasks can run also synchronously or asynchronously. Only the job-queue is supported for the asynchronous mode. Typical examples for the usage of “async” could be “send-email” tasks, so that the user is not affected in the case of a slow email server.

For tasks, the async mode is always enabled (no need to configure anything).

7.1.6 Task delays

You can also define delays for the execution of tasks. There are a number of options here that will make your life easier:

- No delay
- On Creation: uses creation time of the applicable item as basis for the delay (e.g. 1 day after creation)
- On Modification: same as above but taking into account the last modified data.
- On Date: uses a date variable (check out variable section) to which you can add or subtract an off-set. The subtraction part only makes sense for variables that indicate future dates (e.g. planned meeting time).
- On Finish previous task: You can use this in a straightforward way, but for CRM-tasks and calls, this option will allow you to wait for the call or task to be closed (and send a “hooray !” message for example).

7.1.7 Intermediate events

When you design a process that will take certain time e.g. a process that includes 3 sequential notifications for payment due, you may have to cover events that may occur once that process has started, e.g. “payment made!” (happy scenario!).

For this type of scenarios, “Intermediate Events” are used.

Typical constructions would be:

- You define a “Start” event when an element is created, and define in your workflow an activity with a certain delay (e.g. notification)
- You define an “Intermediate” event when the same object is modified.

When the object is modified, the Intermediate event “fires” and gives you the opportunity to influence the already running process instance.

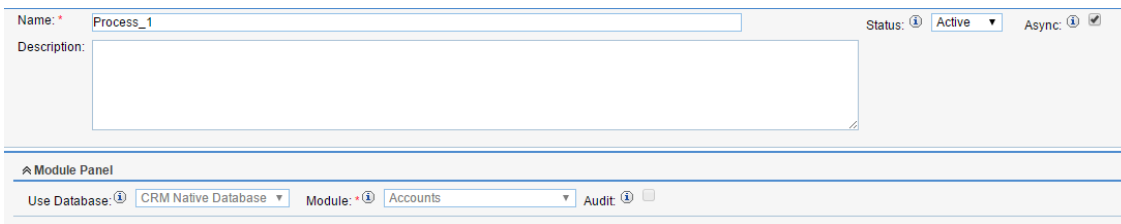
Typical activities related to the Intermediate event (not necessarily an exhaustive list) would be:

- You could send a notification and “kill” the process instance (i.e. the activities that were scheduled for a later time will be deleted).
- You could send a notification stating that something changed, and leave the pending activities be. *You could potentially do this in a separate Workflow using a Start event, but semantically, this is nicer.*
- You could change some global variable so that your pending activities can take this into account.

7.2 vWFM-Process



Assuming you have read the previous section on WorkFlow design, let’s check out the options available.

WFM-Process's EditView:



vWFM-Process's fields:

- Name
- Status

Status	Symbol	Description
Active		Fully active. If events occur, new instances will be created.
Inactive		No new process instances will be created, but elements already active will continue to run, i.e. they will gracefully finish...

- Async

Async	Description
Asynchronous	See also previous section.

	<p>Events with trigger_type=logic_hook will be executed in an independent php-script-instance from the CRM (so in case vWFM execution throws an issue, this issue will not affect the CRM execution).</p> <p>See previous sections about async-modes supported.</p> <p><i>NOTE: when executing in SugarCRM task mode, this will be done under "command mode". What does this mean? Well, your system may use a different php.ini for command mode, so you may see results like your email not working due to the openssl library not loaded. If you encounter issues: under windows go to command line and type "php --ini". This will show you what php.ini file is used for command line. You may need to adapt that configuration file too.</i></p> <p><i>NOTE: there are certain things you do not want to happen synchronously. E.g. changing the value of a module (e.g. calculated status) you want to do synchronous or else you will see the non-calculated value after saving (it will be eventually changed in the background, but you will be looking at the non-calculated value on screen).</i></p> <p><i>Still...If you just Edit a field (not the whole element, e.g. an account), even in synchronous mode, changes executed by vWFM may not appear on screen as the screen is not necessarily redrawn (just to take into account...).</i></p> <p><i>On the other hand, you do not want to wait for a bunch of email notifications to be sent when you perform a save, so this would be a scenario that would fit perfectly with async mode.</i></p>
<p>Synchronous</p>	<p>Synchronous execution.</p> <p>vWFM-events trigger_type=logic_hook and the CRM will be executed in the same php-process. This means that vWFM will execute the WorkFlow before the user gets the CRM response in his browser. Example: if you have defined a WorkFlow that modifies an object when this object is modified by the user then you want to make use of this mode.</p> <p>In case a php-error is thrown by vWFM (E.g.: the user has defined a vWFM- task php_custom that uses a php-function that is not defined), vWFM will handle this error (the browser will be redirected to the current object adding the following text '&_____vWFM_phpError_____' to the url, instead of showing a blank page).</p>

- Description

You may may want to include here information about how your process works and its dependencies. This will facilitate future maintenance.

- Use Database

Normally the CRM native Database option will apply. You can define access to external non CRM databases. For this, the database accesses must be configured in an array within the config_override.php file using “vWFM_AlternativeDbConnections”.

See Config section 12.6.

Although this is a strong feature that enables you to work with multiple databases, define correlations, maintenance processes, etc, be aware that no logic hooks are supported (this is restricted to the CRM database).

- Module

This is the trigger module for the Workflow.

When modifying a vWFM-process you cannot change the module because of security measures.

- Audit

Audit	Description
Yes	Instead of accessing the main module that you have chosen (e.g. accounts) you will access the audit table (in which the audited changes are saved).
No	Access to the chosen module module's data.

7.3 vWFM-Event

You can define multiple events in each WorkFlow. Events are the elements that “kick-off” the workflow.

EditView:

vWFM-Event's fields:

- Name
- Description
- Trigger Type

Trigger Type	Symbol	Description
Logic Hook		Event triggered by the manipulation (create, modify, delete) of objects. One object at the time. Note that this trigger type does not work for audit tables (i.e. it only works for the main data of the chosen module). You must have activated the corresponding logic hooks in the admin section for the events to work.
Scheduled		Event triggered by scheduled-tasks defined within the event. The CRM schedulers must be set-up and working. Also, you must have previously set-up a CRON job for the CRM scheduler.
Subprocess		Event triggered by other process (using the task call_process).
Subprocess local		Event triggered within same process (using the task call_process).

7.3.1 Event type: Logic Hook







Edit View:

Type:





Type	Description
Initialize	<p>This is a very special event that is used to initialize data that can be reused in the rest of the workflow by all WorkFlow instances. It will be triggered the first time that the WorkFlow is called (see for more detail Chapter 7.1.7).</p> <p>When this type of Event is triggered, vWFM will create a working_node of type=initialize. In order to keep the global-variables alive, you must set a delay in a vWFM-task after initializing your global-variables. As long as there are vWFM-working nodes that belong to the same WorkFlow instance, the initialize data will not be deleted from database.</p> <p>Once the delay task expires and there are no working nodes in memory for the process, the Initialize event will be re-executed once the conditions for it apply and before the rest of the workflow gets executed. You can see this as a “reset”.</p>
Start	<p>Triggers the starting point of a process. It initiates a process instance. This type of event is normally used “on creation” of an entity (e.g. an opportunity).</p>

Intermediate	Event that may influence its process when already running. i.e. it is only applicable to already existing process-instances. This type of event is normally used when you want to change something in an already instantiated workflow. E.g. when the field of an opportunity is modified.
Cancel	The cancel event is a specific type of Intermediate Event. This is the event that should be used (just to make the code more readable) when you want to cancel/end the process-instance.

- Trigger Event

Trigger Event	Symbol	Description
On Create		Triggered when creating an object.
On Modify		Triggered when modifying an object. The activities associated with this event will take place after the data is saved in the database.
On Delete		Triggered when deleting an object (before saving it to database).
Login		Only for Users module. Triggered when the user performs a login.
Logout		Only for Users module. Triggered when the user performs an explicit logout. (i.e. no event will be launched for automatic log-outs due to for example end of session).
Login Failed		Only for Users module. Triggered when someone performs a failed login.

The following trigger events require ValeDale Forms&Views:

On Submit		Only for data_source=form. Triggered when the user submits the form
On Init		Only for data_source=form. Triggered when the user loads the form.
After Submit		Only for data_source=form. Triggered after the submit operation is done in the server.
Before Submit		Only for data_source=form. Triggered before the submit operation is done in the server.

- Module Fields

If you have used vReports before, you will recognize this, and the next section as the “classic View of vReports”.

There are four types of fields:

1. (Standard Module) Fields
2. Custom-Fields
3. Related-Fields
4. Related-Custom-Fields.

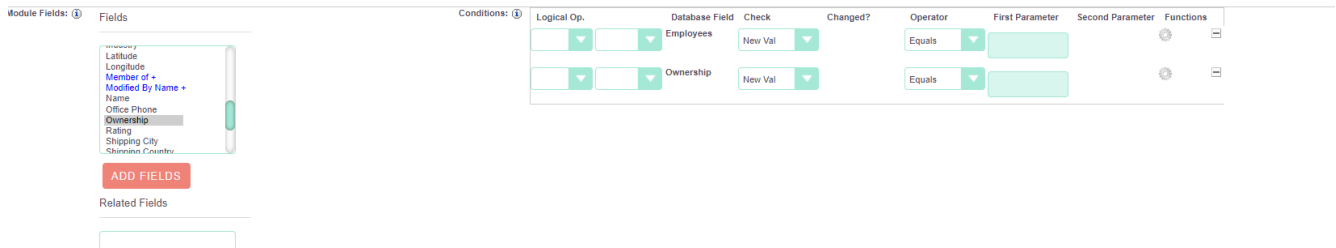
In order to add a Field or a Custom-Field, you need to click on the field you want to add and then click the button "Add Fields".

In order to add a Related-Field or a Related-Custom-Field you need to click on the Related-Field_Link(it is blue) and then click "Show Related" (or double-click over the Related-Field_Link). The Related-Field and the Related-Custom-Fields will appear. Finally, you have to click on the actual related field and click "Add Related Fields".

- Conditions

You can define the conditions that must apply for the Event to “fire”. If you have used vReports, you will recognize this as the classic Interface of vReports.

You can combine AND and OR logical operators. If you do not select a logical-operator for the condition, vWFM will assume it is a logical- AND by default



Note that you can use the brackets in the Logical Operator section to create more complex conditions.

- Database field: by default you will see the name of the label of the field. You can change this behaviour by setting the configuration variable "vWFM_TranslateLabels" to false: if so, you will see the database name. See Config section.
 - You can hover over the label in the list to get more information about the origin of the field (with this for example you can distinguish between the “modified by” of the main module and the “modified by” of a related module).
- Check: You can define if you want the check to be done using the “New Value”, “Old Value” OR just choose “Changed” if you want the condition to apply if the value of the field has changed.
 - In the case of New or Old Value:
 - You can define checks that should apply for the flow to proceed. The operators are type dependent.

7.3.2 Event type: scheduled

The screenshot shows the configuration interface for a scheduled event. At the top, the event name is 'Event_1', the trigger type is 'Scheduled', and the scheduled type is 'Sequential'. Below this is a 'Triggering Panel' containing a table of tasks:

Task Name	Execution Range	Day Value	Time Value	Execution End Date	Task State
Task 0	Monthly	01	02 : 05	17/03/2016	Active
Task 1	Weekly	Monday	04 : 10		Inactive
Task 2	Hourly		15	31/03/2016	Active

Below the tasks table is the 'WFM Event Condition Panel', which allows defining logical conditions. It includes a list of fields (e.g., Employees, Fax, ID, Industry) and a table for defining conditions with columns for Logical Op., Database Field, Check, Changed?, Operator, First Parameter, Second Parameter, and Functions.

The main difference with the Logic Hook Event, is that here you have to define time conditions for the event to be evaluated. "Evaluated" means that the other conditions will be checked to determine if the rest of the flow should be executed

- Scheduled Type

Scheduled Type	Symbol	Description
Sequential	≡	<p>The objects are executed sequentially. This means that a vWFM-activity is executed object by object before the flow gets to the next_activity. One process instance will be created to manage all objects that satisfy the conditions.</p> <p>Note that if you define a vWFM-task with delay, each object will wait for that delay. For example, if a vWFM-activity has only one vWFM-task with a 2 minutes delay and there are 10 objects, the flow execution will take (2 minutes * 10 objects)=20 minutes.</p>
Parallel		<p>The objects are executed in different process-instances, so their executions are independent. If n objects apply for the conditions then there are n process-instances each managing 1 object.</p> <p>It is recommended not to use the Parallel type if the number of</p>

		objects to be processed could be big.
--	--	---------------------------------------

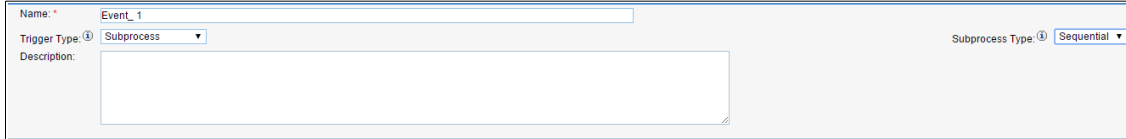
- Scheduled Tasks

You can define here multiple tasks. Note that each task is considered independently: if it applies, the other conditions (not other scheduled tasks) will be evaluated.

For example, if you want that every day at 1 a.m. a flow is executed, add a scheduled task and select:

- Execution_Range: Daily
- Time_Value=01:00

7.3.3 Event type: Subprocess



When trigger_type=subprocess, the Triggering Panel and the Event Condition Panel are not shown.

- Subprocess Type

Subprocess Type	Symbol	Description
Sequential	≡	This results in an atomic execution of the subprocess. I.e. if multiple instances of a process is calling the subprocess, this ensures that only one process at a time is executing this logic. Concurrent subprocess-instances are executed sequentially. This means that there are no conflicts if global-variables are read/modified by the subprocess.
Parallel		Subprocess can be executed in parallel by multiple process instances. Concurrent subprocess-instances are executed in parallel. This means that there could be throw some conflicts if global variables are read/modified by the subprocess.

7.3.4 Event type: local Subprocess

Local sub-processes behave very much like Subprocesses. The main differences are:

- Can only be called by the parent-process
- If you use a "Call Process" task, and you don't define a Sub Process, all the Events of type "Subprocess" will be executed, but NOT the "Local Sub Processes".

7.4 vWFM Activity

Activities are used to group tasks. In the case of sequential activities, all the tasks of one activity will be finished before moving to the next activity.

For an activity to be executed, all the conditions must apply (i.e. be true).

Connectivity rules:

- Events can connect to the same activities, i.e. different events can result in the same flow.
- Any activity can only be reached by only one other activity. If you want the same logic to be executed after different activities, you can model this using Sub Processes.

You can define conditions using a field of the selected module, a related field or a variable. Please see chapter on variables for more information.

The screenshot displays the configuration interface for a vWFM Activity. At the top, the activity is named "WFM-Activity 1". Below the name is a description field and a "Delay" dropdown menu currently set to "Minutes" with a value of "5". The "Type" is set to "Foreach In Group".

The "Activity Condition Panel" is expanded, showing a list of fields on the left and a table of conditions on the right. The fields are categorized into "Module Fields" and "Related Fields". The conditions table has the following structure:

Logical Op.	Database Field	Check	Changed?	Operator	First Parameter	Second Parameter
	account_type	New Value		Equals	Prospect	
	users.date_entered			Equals	22/08/2013	
	custom_variable_1			Equals		

At the bottom, there is a "Custom Variables" section with an empty input field and an "Add Custom Variable" button.

vWFM-Activity's specific fields

- Type

Type	Description
Foreach In Group	Objects received from previous flow-elements will be tested on the condition criteria and if passed, tasks will be executed for those objects.
[Futurable] For each in relation	

- Delay

You can choose the type of delay: minutes, hours, days, weeks, months, and the amount.

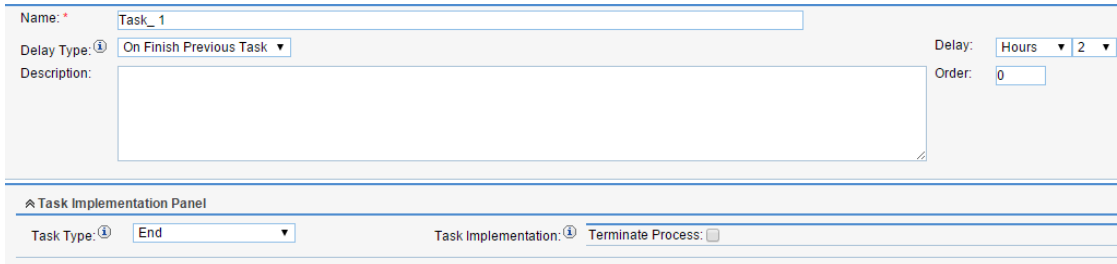
- Custom Variables



See variable section.

7.5 vWFM-Task

vWFM tasks are the “do-ers”: they make things happen. A WorkFlow without tasks has no real purpose.





vWFM-Task's fields



Delay Type	Description
No Delay	vWFM does not wait, it executes the task immediately.
On Creation	vWFM waits. Timebase = date_entered of the object that instantiated the process
On Modification	vWFM waits. Timebase = date_modified of the object that instantiated the process.
On Finish Previous Task	vWFM waits. Timebase = when the previous task is finished. If the previous task is a create_object and the object is a CRM-task or a CRM-call, vWFM will wait until the task is completed or the call is held (task/call is closed).
On Date	vWFM waits. Timebase = input-field Example variables that can be used: <code>\${current_datetime_db_format}</code> , <code>\${bean->datetime}</code> , <code>\${bean->start}</code> , <code>\${bean->end}</code> , 2017-06-01 12:15. This delay_type allows you to wake up the vWFM on an absolute datetime (instead of a relative datetime). In the input-field you must write a database-formatted datetime in GMT time-zone (vWFM- variables of type datetime comply so you can use them without adaptation). You can add/subtract an offset to this datetime (Example: You want to send an email 10 minutes before a Meeting is going to start).




- Order


Order of execution of this vWFM-task within the activity. Note that you can change this also graphically.

- Task Type

Task Type	Symbol	Description
Send Email		<p>The format is the same as for vReports.</p> <p>There are 4 tabs (Summary, TO, CC, BCC).</p> <p>In the Summary-tab you can choose an EmailTemplate, and a From-EmailAddress; also you can see (but not change) the TO, CC, BCC info. In the other three tabs you can set the destination-EmailAddresses. There are three ways to add destinations:</p> <ol style="list-style-type: none"> 1) Users (select the users you want to send an email to). 2) Roles (select the roles you want to send an email to, i.e, to all the users with these roles). 3) Distribution List (standard email address separated by “,”). <p>Variable examples:</p> <p><code>\${Users->created_by->email1}</code> : (Ex: trigger_module=Accounts, send email to the user that has created the Account)</p> <p><code>\${old_bean->email1}</code> or <code>\${bean->email1}</code> : (Ex: trigger_module=Users/Contacts/Leads, send email to the email-address defined in the User/Contact/Lead)</p> <p><code>\${old_bean->vde_email_list}</code> or <code>\${bean->vde_email_list}</code> : (Ex: send email to all email-addresses defined in the trigger-object)</p>
PHP Custom		<p>vWFM will execute the php-code you wrote. Be careful with this vWFM- task, it will be executed whatever the php-code is; you can even delete all your files in your hard-disk-drive.</p> <p>The PHP code goes within the php-tags. Example:</p> <pre><?php \$GLOBALS['log']->fatal('Hello World'); ?></pre> <p>This code will prompt a "Hello World" message in your Suitecrm.log file.</p> <p>You have access to these php-variables:</p> <pre>\$task_id \$alternative_database \$trigger_module \$bean_id</pre>

		<p>\$old_bean \$new_bean \$custom_variables \$current_user_id</p>
Continue		<p>Supportive construct to be used as final task within an activity when the previous task must end before starting the next activity. This will only have a differentiating behavior if the previous created object was a call or a CRM task. You have to define for this the appropriate delay (On Finished Previous Task)</p>
End		<p>Used to indicate the end of a process. This can result in different behaviour depending on related parameter. Terminate Process:</p> <ul style="list-style-type: none"> a) Checked -> terminates process-instance, i.e. all active branches of the process-instance b) Not Checked -> ends branch-execution, but allows process-instance to continue running. (in this case this task does nothing, but it can still look quite elegant ! ☺)

<p>Create Object</p>		<p>Creates a new object. All Mandatory fields will automatically appear so that they can easily be populated. Additional fields can be added and deleted by the user.</p> <p>Be careful if you use a this task and you include a generic event “on_create” in the same process. This will cause an infinite loop if you do not use conditions to avoid it. See variable: <code>#{c_var->bean_ungreedy_count}</code></p>
<p>Modify Object</p>		<p>Not available for “audit” modules, i.e. if audit is not marked in the module selection of the Process. Allows to modify fields of the trigger object.</p> <p>Be careful if you use a this task and you include a generic event “on_modify” in the same process. This will cause an infinite loop if you do not use conditions to avoid it. See variable: <code>#{c_var->bean_ungreedy_count}</code></p>
<p>Call Process</p>		<p>Call Process, as the name suggests, calls another subprocess from within the same or other process. For this, <u>you have to</u> first pick the Process and then select the corresponding event (which must be of type <u>Subprocess</u> or <u>Local Subprocess</u>).</p> <p><i>Note that if you don´t select a Process, you will not be able to select an event.</i></p> <p>You can pick the process and event id using the corresponding selectors, but you can also use a custom variable to define (max) one. (e.g.: <code>#{c_var->call_process1}</code>). Note that the value of the variable must be the <u>id</u> of the process or event respectively. If you use variables, the process-definition is optional.</p> <p>The following event execution criteria apply according to your selection:</p> <p>If you don´t select a process, the selected event will be executed. If you do select process, but no event, all process's events of type Subprocess will be executed (Local Subprocesses will not be executed).</p> <p>vWFM supports some trickier set-ups.</p> <ol style="list-style-type: none"> 1. You can call a Subprocess, based on a different module that the calling Process 2. You can pass objects to the Subprocess. <p>For this, you can find input fields where you can fill in a custom variable (you cannot fill in values directly). In the “Object Module” field you can define the module of the object you are going to pass. e.g. a variable with the value: “Accounts” (the value should not</p>

	<p>include the quotes). If you don't define the object module, vWFM will assume the module-type of the calling process. You can pass the object-ids also as a custom-variable. If not defined, then the id of the data being processed (<code>{bean->id}</code>) is assumed</p> <p>object_ids: If you only want to pass one object, just use a variable with the id of the object as its value. If you want to pass multiple objects: the typical scenario would be to obtain the object_ids using the task "get_objects". However you can also use PHP code to assign the valued to the custom variable. Basic example:</p> <pre><?php custom_variables['var_account']='260e366f-e546-f89e-0cd8-5ec27ca0a814\${pipe}999999999999999999'; ?></pre> <p>Note the use of single quotes and the specific separator "<code>{pipe}</code>".</p> <p>Note that in the call_task you would refer to the variable above as: <code>{c_var->var_account}</code> (see variable section)</p> <p>If you call a sub-process using a Module-type different from the Sub-process, you should NOT use conditions in the subprocess related to fields as they do not correspond,</p> <p>You can call various processes at the same time, using a vWFM custom variable. If so, you must use the following format in the variable value: <code>;;process_id_1;;process_id_2;;process_id_3;...;;process_id_n;</code> (do not use the quotes). I.e. make sure you start with a ";" and separate the ids of the processes with ";;".</p>
<p>Add Custom Variables</p>	<div data-bbox="432 1559 496 1626" style="float: left; margin-right: 10px;">  </div> <p>You can define variables (and change the values of previously defined ones) and then use them in another task or activity; the custom_variables are inherited from vWFM-activity to their Next Activities.</p> <p>Let's take a simple flow as an example to see the various options available. We will use some fun coding as example: Every time the name of an account is changed, we change "Annual Revenue" of the company.</p>



Conditions for Name Changed						
Logical Op.	Database Field	Check	Changed?	Operator	First Parameter	Second Parameter
	Name	Changed	Yes			

The condition above determines that something should happen when the name of the account changes.

Let's define a variable, based on Annual Revenue. Note that we define the name of the custom variable in the name column.

Name	Type	Module	Field	Functions	Is Global?
annual_revenue_inc	PHP ev	Annual Revenue			<input type="checkbox"/>

In the modify object Task, we will assign the value of this variable as the new value of Annual Revenue. Note that we have to use in the assignment as indicated below de full c_var... formulation.

Object Module: Accounts

Fields:

- Address
- Alternate Phone
- Annual Revenue
- Assigned User
- Billing City
- Billing Country
- Billing Postal Code
- Billing State
- Billing Street
- Campaign

Database Field: Annual Revenue

Value: `#{c_var->annual_revenue_`

`#{c_var->annual_reven`


Let's look at what we can do with the different types of definition.

PHP evaluation:



- Add Annual Revenue as field
- Select PHP eval as Type
- Define the function as: `$new_bean['annual_revenue'] + 1`

SQL:

You can achieve the same by defining the function as:
`#{this} + 1`

		<p>Every time that you change the name of the Account, the Annual Revenue of the company will be incremented by 1 ! (keeping your customers happy 😊 !)</p> <p>Or to make it bigger based on employees <code>\${this} + 30 + \${bean->employees}</code></p> <p>And to make that more robust; <code>ifnull(\${this} ,0) + 30 + ifnull(\${bean->employees},0)</code></p>
<p>Get Objects</p>		<p>With this task you can query the database for objects. The task will generate a custom variable using the name you defined. <code>\${c_var->get_objects->[name]}</code></p> <p>For example, <code>\${c_var->get_objects->top_opportunities}</code></p> <p>The vWFM_custom_variable is an Array with 2 elements: ('object_module'=>\$objectModule, 'object_ids'=>\$object_ids_string). So in order to use this custom variable in a call_process task you must use these two variables <code>\${c_var->get_objects->[name]->object_module}</code> and <code>\${c_var->get_objects->[name]->object_ids}</code></p>

7.5.1 Form tasks (not yet released)

<p>[Not yet released] Forms Response</p>		<p>This vWFM-task will immediatly quit the vWFM's execution. The vWFM will respond with json-instructions that Forms can understand and Forms will continue its execution. The response is: <code>\$custom_variables['sys_composite_forms_response']</code> This custom_variable is made by: <code>\$custom_variables['sys_composite_forms_response']['success']</code> <code>\$custom_variables['sys_composite_forms_response']['messages']</code> <code>\$custom_variables['sys_composite_forms_response']['actions']</code></p>
<p>[Not yet released] Forms Error Message</p>		<p>This vWFM-task will not quit the vWFM execution. (The opposite to Forms Response) If this vWFM-task is executed then <code>\$custom_variables['sys_forms_success']=false</code> (this custom_variables's default value is 'true'). Eachv WFM-task 'FormsError Message' will add messages to <code>\$custom_variables['sys_composite_forms_response']['messages']</code></p>

7.6 Task definition examples

7.6.1 Send Email

Task Implementation Panel

Task Type: Task Implementation:

Summary (fill TO-CC-BCC-data in other tabs)

Email Tpl: System-generated password email

From: from@test.com

Reply to: reply_to@test.com

Return path: return_path@test.com

TO:

Users: admin

Roles: role_1,role_2

Notification Emails:

Distribution List: to_1@test.com

CC:

Users: user_1,user_2

Roles: role_3

Notification Emails:

Distribution List: to_2@test.com,to_3@test.com

BCC:

Users:

Roles:

Notification Emails:

Distribution List: to_4@test.com

7.6.2 PHP custom

Task Implementation Panel

Task Type: Task Implementation:

```

1 <?php
2
3 function log($text) {
4     $GLOBALS['log']->debug($text);
5 }
6
7 for ($i=1; $i<10; $i++) {
8     log($i);
9 }
10
11 ?>
    
```

7.6.3 vWFM-Task task_type=continue

Task Implementation Panel

Task Type: Task Implementation:

7.6.4 End

Task Implementation Panel

Task Type: Task Implementation:

7.6.5 Create Object

Task Implementation Panel

Task Type: Task Implementation:

Fields	Database Field	Value
Assigned User	Duration Hours *	1
Created By	Start Date *	\$(current_datetime_db_K) + Y: M: 01 D: 03 h: 02 i: 10
Date Created	Status *	Planned
Date Modified	Subject *	Son of \$(bean->name)
Description		
Direction		
Duration Hours *		
Duration Minutes		
Email reminder sent		
Email Reminder Time		

Relationships	Relationship Name	Relationship Value
Account (Accounts) [account]	Account (Accounts) [account_calls]	\$(bean->id)

7.6.6 Modify Object

Task Implementation Panel

Task Type: Task Implementation:

Fields	Database Field	Value
Billing Street	Type	Competitor
Campaign	Annual Revenue	300
Created By	Description	\$(bean->description) mod
Date Created		
Date Modified		
Description		
Employees		
Fax		
Industry		
Member of		

Relationships	Relationship Name	Relationship Value
Assigned to User (Users) [ac]		
Bugs (Bugs) [accounts_bugs]		
Calls (Calls) [account_calls]		
CampaignLog (Campaign Lc)		
Campaigns (Campaigns) [ca]		
Cases (Cases) [account_cas]		
Contacts (Contacts) [account_cas]		
Created by User (Users) [acc]		
Documents (Documents) [doc]		
Email Address (Email Address)		

7.6.7 Call Process

Name:

Delay Type: Order:

Description:

Task Implementation Panel

Task Type: Task Implementation:

Process's Event:

Object Module: Object Ids:

Execute subprocess immediately and wait until is done:

7.6.8 Add Custom Variables

The screenshot shows the 'Task Implementation Panel' with 'Task Type' set to 'Add Custom Variables' and 'Task Implementation' set to 'Accounts'. On the left, there are two lists of fields: 'Fields' and 'Related Fields'. The 'Fields' list includes items like 'Alternate Phone', 'Annual Revenue', 'Assigned User', etc. The 'Related Fields' list is currently empty. On the right, a table lists various fields with their names, types, module fields, functions, and whether they are global.

Name	Type	Module Field	Functions	Is Global?
account_type	SQL	Type		
annual_revenue	SQL	Annual Revenue		
a_rev	SQL	Annual Revenue		
sic_code	PHP eval	SIC Code		
description_test	SQL	Description		
users_user_name	SQL	Users.User Name		
accounts_account_type	SQL	Accounts.Type		
bugs_assigned_user_id	SQL	Bugs.Assigned To		
calls_created_by	SQL	Calls.Created By		
contacts_alt_address_state	SQL	Contacts.Alternate Address State		
documents_doc_url	SQL	Documents.Document Source URL		
from_relationship_1	SQL	Email Address.Date Create		
from_relationship_2	SQL	Email Address.Date Create		
from_rel_a	SQL	Emails.Date Sent		
from_rel_b	SQL	Emails.Date Sent		
not_a_field	Literal			
not_a_field_2	PHP eval			

7.6.9 Forms error message

The screenshot shows the 'Task Implementation Panel' with 'Task Type' set to 'Forms Error Message' and 'Task Implementation' set to 'Response Generator'. Below the header, there is a table with columns for 'Scope', 'Field', 'Color', and 'Value'. One row is visible with 'Field' set to 'label_date_2', 'Color' set to 'FF0000', and 'Value' set to '[t must be today]'.

Scope	Field	Color	Value
Field	label_date_2	FF0000	[t must be today]

7.6.10 Forms response

The screenshot shows the 'Task Implementation Panel' with 'Task Type' set to 'Forms Response' and 'Task Implementation' set to 'Response Generator'. The configuration includes sections for 'Success', 'Messages', and 'Actions'. The 'Success' section has a dropdown set to 'sys_forms_success'. The 'Messages' section has a table with columns for 'Scope', 'Field', 'Color', and 'Value'. The 'Actions' section has a table with columns for 'Operation', 'Target', and 'Parameters'.

Operation	Target	Parameters
Load	Self	Forms Form_2

8 vWFM Variables

vWFM supports variables. With this you can create more complex Work Flows. You can create variables using the task "Add custom variables" and change the values using the same task or directly in PHP using the PHP Custom task.

Depending on where you want to use the variables, the formulation may differ. This is explained also in this chapter.

8.1 Types of vWFM-Variables

8.1.1 Module-data Variables

8.1.1.1 Main data (*bean*)

These variables are created by vWFM. You should not override them.

CRM data is mainly accessed through "Beans". There are two sets of data: the original data for that account and the data with all the changes performed. You can access the "original data", using:

```
#{old_bean->[field_name]}
```

And the data with all the performed changes, using:

```
#{bean->[field_name]} (you can also use "new_bean" instead of "bean")
```

Check out "`#{c_var->modified_new_bean->[custom_variable_array_key_name]}`" (later in this chapter) for an additional scenario.

The set of field_name that you can use are different depending on the object's module (trigger_module).

Examples:

```
#{bean->name}, #{bean->account_type}, #{bean->amount}, #{bean->date_closed},  
#{bean->date_start}, #{bean->status}, #{bean->is_admin}.
```

You can use the [Variable Generator](#) to generate these vWFM-variables.

8.1.1.2 Email related variables

- 1) `#{bean->email1}` for example when you want the first email of a User.
- 2) `#{bean->vde_email_list}`, for example when you want all emails of a User separated by

commas.

These two vWFM-variables cannot be used in all modules.

8.1.1.3 Data in related modules

`${[related_module]->[related_field_name]->[field_name]}`

Allows you to access the related_module-object's fields through the related_field_name.

Examples:

- trigger_module=Accounts: `${Users->assigned_user_id->user_name}`. This will get the user_name of the User with id=`assigned_user_id`.
- trigger_module=Accounts : `${Users->created_by->user_name}`. This will get the user_name of the User with id=`created_by`.
- Example 3: trigger_module=Tasks → `${Contacts->contact_id->phone_work}`. This will get the phone_work of the Contact with id=`contact_id`.

8.1.2 WorkFlow related data

`#{c_var->[custom_variable_name]}`

➤ `#{c_var->num_objects}`

Number of objects that meet the conditions as defined in the workflow. If

`trigger_type=logic_hooks` → `num_objects=1` If

`trigger_type=scheduled` → `num_object=n`

➤ `#{c_var->iter_object}`

`iter_object {0,1,...,n-1}`

➤ `#{c_var->bean_ungreedy_count}`

This is a counter that you need to include in your conditions (for certain scenarios) to ensure that you are not creating infinite loops in your workflow. For example, if you modify data in an account, and in your workflow you modify again account data you should include in the conditions: `bean_ungreedy_count=0`. This will ensure that your workflow will not be executed twice.

➤ `#{c_var->trigger_event}`

Gives you vWFM-event's `trigger_event`.

Complex data:

`#{c_var->[custom_variable_array_name]->[custom_variable_array_key_name]}`

➤ `#{c_var->server->[custom_variable_array_key_name]}`

Built from `$_SERVER` php-global-variable.

Examples: `#{c_var->server->HTTP_HOST}` , `#{c_var->server->HTTP_USER_AGENT}` , `#{c_var->server->client_ip}` (special, created by vWFM).

Note that this information may not always be available.

➤ `#{c_var->request->[custom_variable_array_key_name]}`

Built from `$_REQUEST` php-global-variable.

Example: If you are at the Suitecrm login page → `#{c_var->request->user_name}`

➤ `#{c_var->current_user->[custom_variable_array_key_name]}`

Built from `$current_user` php-global-variable.

You can access:

- `#{c_var->current_user->id}`
- `#{c_var->current_user->is_admin}`
- `#{c_var->current_user->roles}`

➤ `#{c_var->env->[custom_variable_array_key_name]}`

Built from `$_ENV` php-global-variable.

Examples: `#{c_var->env->COMPUTERNAME}`, `#{c_var->env->OS}`, `#{c_var->env->PROCESSOR_IDENTIFIER}`, `#{c_var->env->USERNAME}`

➤ `#{c_var->modified_new_bean->[custom_variable_array_key_name]}`

The vWFM-variable `#{bean}` is loaded at the beginning of the workflow-execution and it is not updated when a `modify_object` is executed. So, you can access updated data by means of `#{c_var->modified_new_bean}`, because this vWFM-variable is updated when a Suite-object is modified by vWFM-task `modify_object`.

8.1.3 User-defined vWFM-Variables

You can define your own vWFM-variables. After the definition, you can assign new values using the same task: `add_custom_variables`.

All user-defined vWFM-variables are `custom_variables`, you can access them by means of the `#{c_var}` construct.

8.1.4 Variable inheritance (in subprocess)

You can access in a subprocess the bean data from the calling process through a predefined custom variable. For example, (the `parent_process` variable `#{bean->start}` equals to the subprocess variable `#{c_var->parent_process_bean->bean->start}`).

8.2 vWFM-Variables usage

- Email Templates: you can use vWFM-variables in your emails, both in the email-body and in the email-subject.
- vWFM-Tasks:
 - `create_object`, `modify_object`, `add_custom_variables`.

Examples: `#{bean>name}`, `#{c_var->bean_ungreedy_count}`, `#{c_var->current_user->id}`, etc.

- `php_custom`, you can use vWFM-variables in this vWFM-task, but with a different nomenclature:

Examples:

- Do not use `#{bean->name}` → you must use `$new_bean['name']`
- Do not use `#{c_var->current_user->id}`
 - → you must use `$custom_variables['current_user']['id']`
- Do not use `#{g_c_var->myVariable}`
 - → you must use `$custom_variables[['GLOBAL_CVARS']][' myVariable ']`

- vWFM-Event and vWFM-Activity: you can use `custom_variables` as conditions. But here

you must use a simplified way to write vWFM-variables.

Example: Do not use `#{c_var->bean_ungreedy_count}` → you must use `bean_ungreedy_count` (the name of the variable).

8.3 Modifying Global Variables

Global variables can be kept in memory for a user-definable period of time. This can be done by just defining the global variables then letting the “Initialize branch” go to sleep for a certain period of time.

If the administrator wants to change these global variables, currently, he/she would have to change the corresponding values in the workflow and then delete the working-node that keeps the branch asleep. At the next execution, vWFM will notice that the Initialize branch is not executed yet and will do so before checking on the other events.

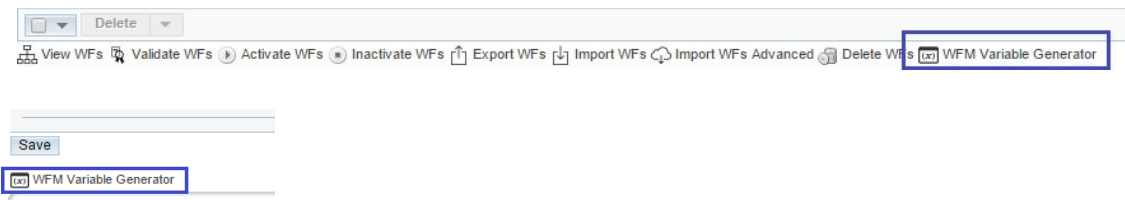
Name	Type	Process Instance Id	Priority	Event	Current Activity	Trigger Module	Executing Object	Iter Object	Current Task	Delay Wake-up Time	Status	Date Created	Date Modified	Created By	Modified By
w_n_4794a78-627f-71af-cde8-532fe49c664	Start	p_i_477cdf79-bd24-04ea-756f-532fe39a0f5	-1		next_user™	Leads	ToM ToM6	0			Terminated	24.03.2014 10:44	24.03.2014 10:44	Administrator	Administrator
w_n_47929c3f-830-9cda-08fe-532feda9610	Initialize	p_i_477cdf79-bd24-04ea-756f-532fe39a0f5	-1			Leads	ToM ToM6	0	Keep variables alive	25.03.2014 10:44	Delayed By Task	24.03.2014 10:44	24.03.2014 11:11	Administrator	Administrator

Note that the name that appears in the vWFM monitor, Working nodes list, is the name you have given the task that sent the branch to sleep.

8.4 Variable Generator

The vWFM Variable Generator, as its name implies, is used to generate “variables” (Eg: \${bean->name}). Based on the input parameters you define, it will generate the variable that you can use. Note though that depending on where you use the variables, you may have to adapt the format.

You can find the launch button for the variable generator in the vWFM-process module ListView and in the EditViews (at the bottom) of vWFM-event, vWFM-activity and vWFM-task.



First, you have to define what variable type you want to generate.

WFM variable types
<input checked="" type="radio"/> Custom variable predefined (system)
<input type="radio"/> Custom variable user defined
<input type="radio"/> Dates
<input type="radio"/> Data source is "database"
<input type="radio"/> Data source is "form"

num_objects ▼ +

Depending on your choice, additional options will be presented. For example, if you want to generate a variable related to the CRM database:

WFM variable types

<input type="radio"/> Custom variable predefined (system)
<input type="radio"/> Custom variable user defined
<input type="radio"/> Dates
<input checked="" type="radio"/> Data source is "database"
<input type="radio"/> Data source is "form"

Object Module: Accounts ▼ Audit:

Fields

Address
 Alternate Phone
 Annual Revenue
 Assigned User +
 Billing City
 Billing Country
 Billing Postal Code
 Billing State
 Billing Street
 Campaign +

Related Fields

Users
 Users.Address City
 Users.Address Country
 Users.Address Postal Code
 Users.Address State/Province
 Users.Address Street
 Users.Assigned To
 Users.Authentication Key
 Users.Date Entered
 Users.Date Modified

New Bean
 Old Bean

Once you have selected a module, its fields and related fields will be shown.

if you wish to generate a variable for a field:

1. select your field
2. choose either 'New Bean' or 'Old Bean'
3. click on 'Add Fields'.

If you want to generate a related_field variable, you just need to select your desired related_field and click on 'Add Related Fields' (note that the related field is independent of the type of bean). In order to reach a related field, you first have to select the "link" field (in blue) and then click on the 'Show Related' button. You can also double-click on the field in blue.

The variable will appear in its corresponding box. You can copy the value from there for reuse.

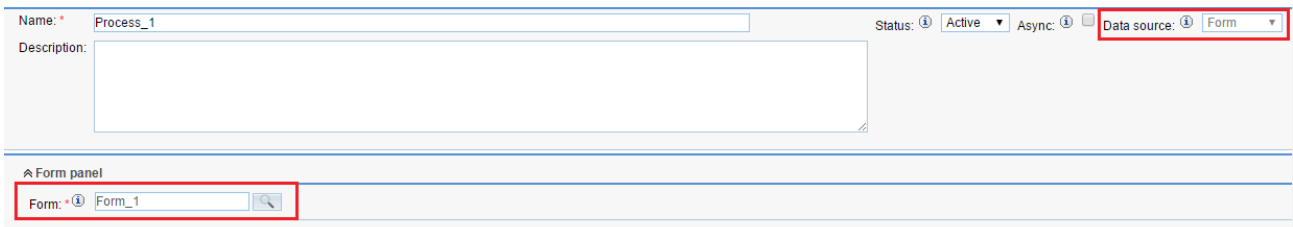
For user defined custom variables, you have to type yourself the name of the variable. Currently vWFM does not collect this information.

9 vWFM & module Forms (not released yet)

If you have installed ValeDale's Forms module in your SuiteCRM instance then you can use the vWFM in order to generate responses for the Forms module.

For example: you have a form that has an editable field, in this case a contact's name, when you submit this form you need to check if this contact's name has already been stored in database (the contact exists). So, if the contact does not exist the vWFM creates the contact and shows to the user a success message in his/her browser, but if the contact does exist then the vWFM shows to the user a validation error message in the same form.

In order to create a workflow that manages the response for a form, you need to create a vWFM- process with "Data source" equals to "Form" and then select this form in the "Form panel".

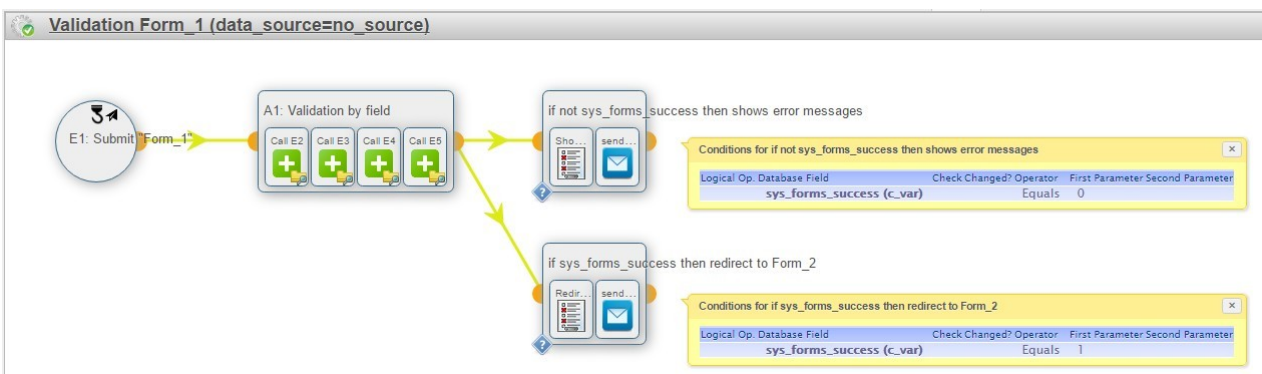


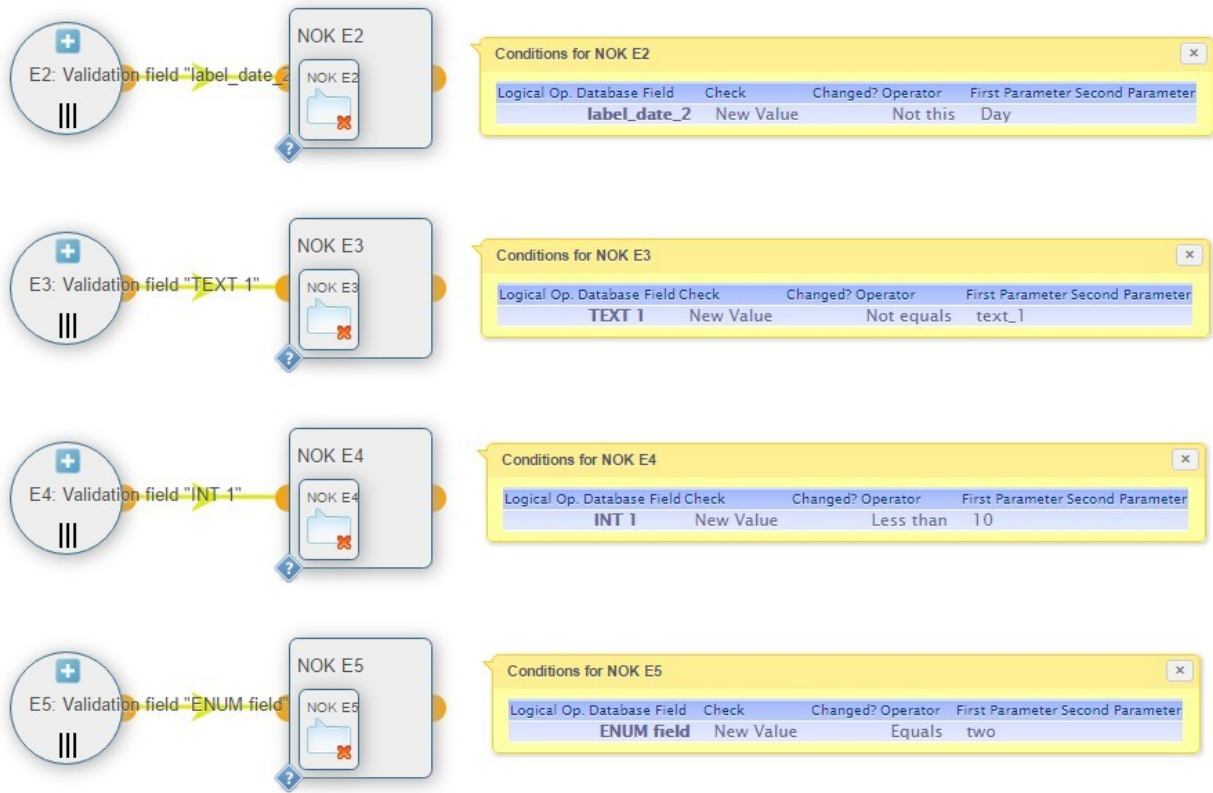
If you want the vWFM to generate a response for Forms then you need to choose "Synchronous" mode ("Async" parameter).

In a workflow with "Data source" equals to "Form" you can add events specifically designed to manage form's responses, these are "On Submit", "On Init", "Before Submit", "After Submit". The more important is "On Submit". If you need the vWFM to check some validations for some fields and then return a response for Forms module then you should add a vWFM-event to the workflow with these properties {"Trigger Type" = "Logic Hook", "Trigger Event" = "On Submit"}. In order to perform validations you should add conditions to vWFM-activities. Finally, there are two vWFM-tasks designed for module Forms, these are {"Forms Response", "Forms Error Message"}, with these vWFM-tasks you can generate the response for module Forms.

If you have designed a form with a lot of fields and you need to perform validations for every field then you need to design a workflow in a scalable way. We suggest you to use subprocesses. Each subprocess will be a validation for a field. For each validation you should use a vWFM-task "Forms Error Message". These will conditionally add error messages. After you call all subprocesses you should add a vWFM-activity with a condition over the custom-variable "sys_forms_success", within this vWFM-activity you need to add a vWFM-task "Forms Response".

Workflow example:





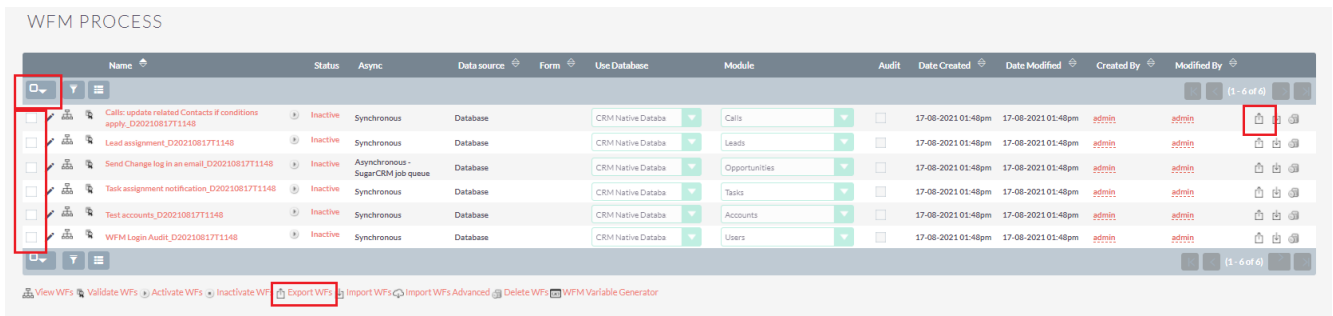
This workflow has been made for the form:

The screenshot shows a form editor interface with a top toolbar and a main workspace. The workspace contains a grid of form fields organized into three sections. The first section has two panels: 'Edition Buttons 1' and 'Function Button 1'. The second section has two panels: 'DATE 2' and 'DATE 1', and 'TEXT 1' and 'VARCHAR 1'. The third section has two panels: 'INT 1' and 'DECIMAL 1', 'BOOL 1' and 'ENUM field', and 'BOOL 8'.

10 Exporting and Importing WorkFlows

10.1 Export WorkFlows

You can export a workflow individually using the action button to the right of the WorkFlow entry, or you can select multiple ones and select the 'Export WFs' option at the bottom. A save-dialog will open then so you can choose where you want to store these workflows.



10.2 Import WorkFlows

First some advanced topics:

In case you decide to use global libraries in Workflows (using sub-processes), you will always have to use “advanced imports” or “import with context” for all those sub-processes (libraries) and Processes that call the libraries. This is due to the references in the “call process” tasks for processes and events.

Basic import does not keep the “global” ids, but still, basic import should work when:

- You only use local libraries (i.e. calls to sub-processes in the same process)
- You import the libraries and other processes in the same file, making sure that the libraries are imported first (they are imported in the same order as they were exported).

There is a peculiarity when importing a vWFM-task of task_type=php_custom. Usually vWFM- task are only stored into database when importing a workflow; but with the php_custom vWFM- task, the vWFM will store the php-script into a disk-file (besides into database). If this disk-file somehow (maybe due to wrong file permissions) were not created, then the workflow will not work. If this happens, you just need to go to vWFM-task's EditView and save it, this operation will create the disk-file.

10.2.1 Import WorkFlows without context

This feature clones workflows:

- Clone vWFM-entities (new ids).
- vWFM-process->status = inactive

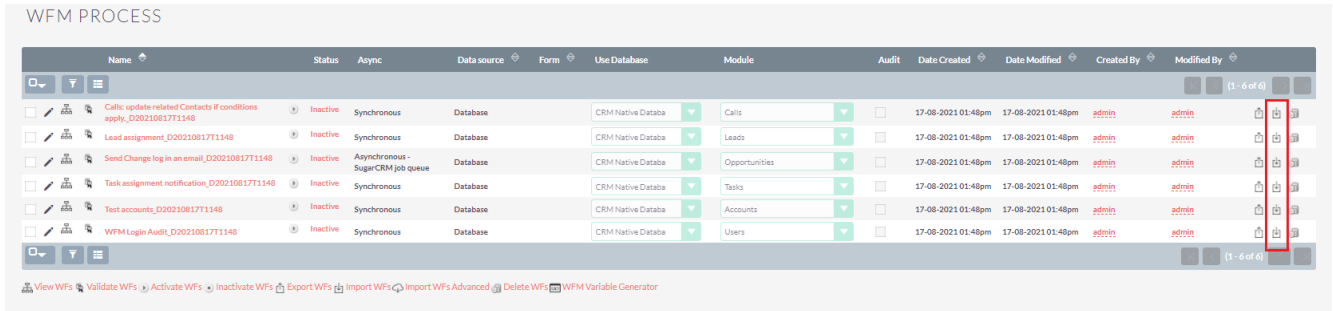
The screenshot shows the 'WFM PROCESS' interface. It features a table with columns: Name, Status, Async, Data source, Form, Use Database, Module, Audit, Date Created, Date Modified, Created By, and Modified By. The table lists several workflow processes, all with a status of 'Inactive'. Below the table, there is a navigation menu with options: View WFs, Validate WFs, Activate WFs, Inactivate WFs, Export WFs, **Import WFs**, Import WFs Advanced, Delete WFs, and WFM Variable Generator. The 'Import WFs' option is highlighted with a red box.

You must choose file and click on 'Import WF'.

10.2.2 Import WorkFlows in context

This feature replaces workflows:

- Replace vWFM-entities (same ids).



You must choose file and click on 'Import WF':

10.2.3 Import WorkFlows Advanced

You can find the advanced import action at the bottom of Process list view.

This feature allows to choose among several options.

Step 1: Check if WorkFlows already exist.

Import WorkFlows Advanced (admin-users only)

Step 1: Check if WorkFlows already exist.

Choose File Check

If WorkFlows do not exist:

Import WorkFlows Advanced (admin-users only)

Step 2: WorkFlows do not exist. Import.

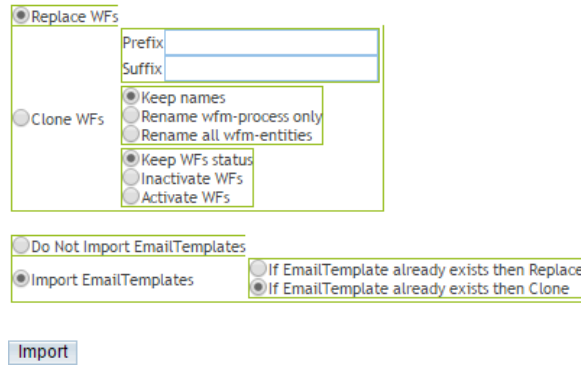
Do Not Import EmailTemplates
 Import EmailTemplates
 If EmailTemplate already exists then Replace
 If EmailTemplate already exists then Clone

Import

If WorkFlows exist:

Import WorkFlows Advanced (admin-users only)

Step 2: WorkFlows already exist. Choose your options and Import.



Replace WFs

Prefix

Suffix

Clone WFs

- Keep names
- Rename wfm-process only
- Rename all wfm-entities

- Keep WFs status
- Inactivate WFs
- Activate WFs

Do Not Import EmailTemplates

Import EmailTemplates

- If EmailTemplate already exists then Replace
- If EmailTemplate already exists then Clone

If the WorkFlow does not exist, the IDs of the Workflow will be maintained.

11 Bundled WorkFlows (examples)

wWFM is bundled with an example library of Processes. It is highly recommendable that you analyse them and potentially use them as the basis for your own Processes.

You can import the examples from the zip file. You can find the single WorkFlows, or the combined set (“ZZZ Exported_WorkFlows...”).

In order to make the examples work:

- Change the default sender email in “send email” tasks
- Activate the appropriate logic hooks if not already activated
- Activate CRON and scheduler if not already done
- Activate Workflow

Check out each WorkFlow for potential additional requirements.

action=execute_process (Required in order to use workflows with wfm-events of trigger_type=logic_hook)

Module	after_login	before_logout	login_failed
Users	✘	✘	✘

11.1 Lead Assignment with Views

Check out bundled Views for this Workflow that facilitates provisioning. Supported by RunTime Version of Forms&Views.

MANAGEMENT AUTOMATIC LEAD ASSIGNMENT

Status WF: Active Data Not Initialized ↕

SAVE CANCEL

Routing Data

Routing Type: Roles Agent List Default+Fixed

Use Audit Login: No ▼

Fixed/Default

default_user: admin

Lead Agents

Agentlist (username separated by ,): agent.sales

Roles

Roles (name separated by ,): agent

Support Lookup Reports

» List Users

» List Assigned Roles

For functionality, refer to section 11.6.

11.2 Login Audit

Check out bundled reports for Login Audit that come with vReports.

vWFM is bundled with a straightforward but very useful feature: Login audit. Note that this is a module on its own: it has its own data. This feature keeps track of who has logged in and out (explicitly) and when failed logins take place. A default presentation of the activity is provided.

With vReports (from the ValeDale productivity suite) you can then create your own dashboards. You can potentially also create your own workflows to launch alarms when too many failed login attempts take place.

The following triggers are monitored (these should be set to active: see admin section) (by default they are NOT active)

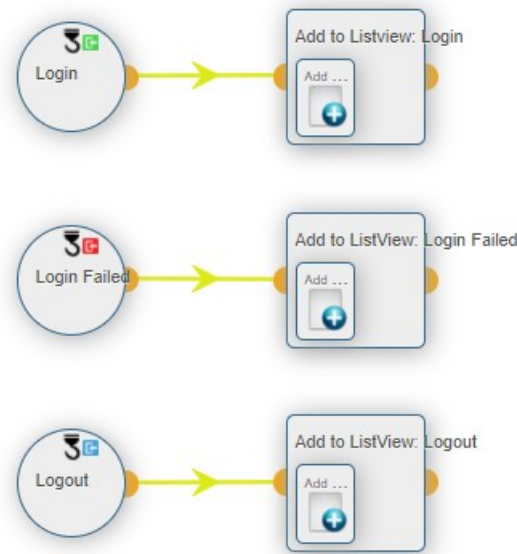
- after_login (Login)
- before_logout (Logout)
- login_failed (Login Failed)

Three events are defined, each resulting in the execution of one task. All tasks just add the corresponding log-in information about the performed action.

Note: although the menu item is automatically created when you install vWFM it is not automatically activated.

To do so:

1. Import the Workflow itself
2. Activate the corresponding logic hooks (see above)
3. Activate in the WFM list view, the workflow (click on the play button) (by definition an imported Workflow will be inactive).



See the extensive use of system custom variables to fill-in the entry for the successful log-in entry in the example below. Note that if you hover over the variables in the graphical WFE, you will see their full name.

Task Implementation: 1 Object Module: WFM Login Audit

Fields	Database Field	Value
Action *	Action *	Login
Action Time	Action Time *	\$(current_datetime_db_foi) + Y: M: D: h: i:
Assigned User Id	Ip Address *	\$(c_var->server->client_ip)
Created By	Is Admin? *	\$(c_var->current_user->is)
Date Modified	User ID *	\$(c_var->current_user->id)
Description	User Roles *	\$(c_var->current_user->ro)
Ip Address *	Username Tried *	\$(c_var->request->user_n)
Is Admin? *		
Name	Relationship Name	Relationship Value
User *		

The default presentation looks like this (this can be improved with vReports).

User	Username Tried	Is Admin?	Ip Address	Action	Action Time	User Roles
Administrator	admin	1	127.0.0.1	Login	28/08/2013 10:59	
Administrator		1	127.0.0.1	Logout	28/08/2013 10:59	
.	user_wrong	-	127.0.0.1	Login Failed	28/08/2013 10:56	-
user2		0	127.0.0.1	Logout	28/08/2013 10:55	role_1,role_3
user2	user2	0	127.0.0.1	Login	28/08/2013 10:55	role_1,role_3
.	user2	-	127.0.0.1	Login Failed	28/08/2013 10:55	-
user1		0	127.0.0.1	Logout	28/08/2013 10:55	role_1
user1	user1	0	127.0.0.1	Login	28/08/2013 10:55	role_1

As you can see, you have access to the user that has performed both a successful login and or logout. Failed logins only provide limited data.

Login failed can be triggered by two cases:

- a) Wrong username
- b) Wrong password

Other information you have access to: `username_tried`, `is_admin`, `ip_address`, `action`, `action_time` and `user_roles` (some of this data may not be available depending on your set-up).

11.3 Keep track of last time a contact was approached

This process is defined for the module calls, but in practice a similar flow should be defined for meetings.

The main part checks if the call that was just marked as “held” is the last hit a contact got. If so, it updates a timestamp (in this case the field “date_review” is used in the contact module – this is a standard field in the CRM; if you are already using it, change this to another field).

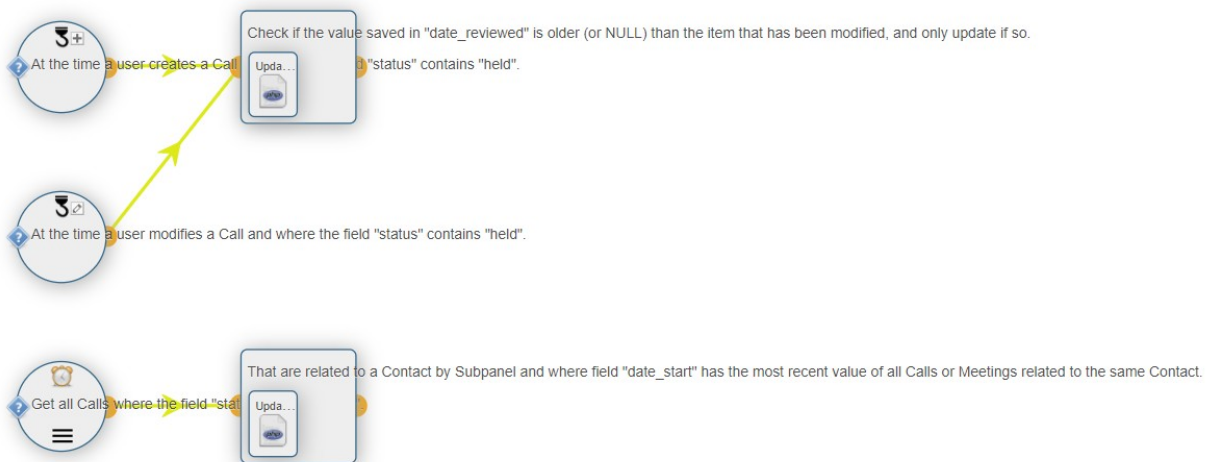
In this way one can supervise that contacts are regularly “contacted” (alerts can be set with for example vReports).

There is also a scheduled event that you can use for various purposes:

- You could run it once to fill-in the data per contact, based on already held meetings.
- You can run it every night to ensure that all calls really have been taken into account for example in the last week (just in case the vWFM was not active then).

The scheduled process also uses the contact custom field “subject_last_meetcall_held_c”. This field is set to the “subject” of the last call (if that was the last interaction). Note that you will have to define this field with “Studio” for the scheduled process to work (you could enhance the process by saving the id of the call or meeting to access the potential notes easily).

The process consists of three events, and some PHP code to do the magic.



The first 2 events are triggered by newly created and modified calls that are marked as “held”. The last event is the scheduled event that looks at passed calls and checks both calls and meetings to pick the latest “hit”.

Depending on what you want to achieve, you may want to run the process less often and take into account potentially a smaller back-log.

Note that as the picture below shows, the event is triggered every day at night and it checks all calls that are held, and that had a start date within this week, or passed week.

A Triggering Panel

Scheduled Tasks: 1 Tasks

Task Name	Execution Range	Day Value	Time Value	Execution End Date	Task State	ADD NEW TASK
Task 0	Daily	21	: 30		Active	

Trigger Module: Calls Audit:

A WFM Event Condition Panel

Module Fields: 1 Fields Conditions: 1

Fields

- Assigned User +
- Created By +
- Date Created
- Date Modified
- Description
- Direction
- Duration Hours
- Duration Minutes
- Email reminder sent
- Email Reminder Time

ADD FIELDS

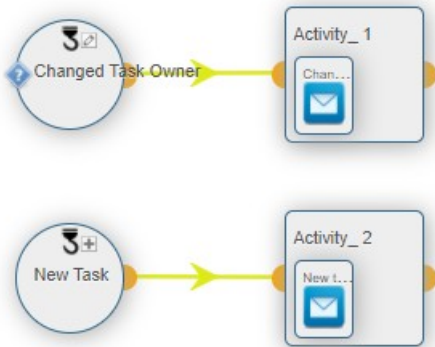
Logical Op.	Database Field	Check	Changed?	Operator	First Parameter	Second Parameter
(AND	Status	New Val	Equals	Held	
(OR	Start Date	New Val	This	Week	
-))		Start Date	New Val	Last	Week	1

11.4 Task assignment notification

This process sends an email to the user that got a task assigned. Note that this mimics partly standard CRM behaviour.

The process consists of two events:

- Changed owner for existing task
- New task



The first event applies when the assigned_user changes.

The notification email is sent to “`${Users->assigned_user_id->email1}`” using a specific email template (you may want to put a little bit more design juice into that template).

11.5 Send Opportunities change log in emails

This process sends regularly an email to a predefined distribution list with selected audit changes to Opportunities. The way it is defined, one email is sent for each opportunity. Most of the logic comes from custom PHP code that you can reuse.

It makes use of the SugarCRM job queue for the process, so you may have to change this to Asynchronous mode, or activate it through configuration (`WFM_enable_async_sugar_job_queue`).



The sole event is a scheduled event.

The screenshot shows the configuration for a task named "Get all Opportunities". The Trigger Type is set to "Scheduled" and the Scheduled Type is "Sequential". Below this is a "Triggering Panel" and a "Scheduled Tasks" table. The table has columns for Task Name, Execution Range, Day Value, Time Value, Execution End Date, and Task State. One task is listed with the name "last 1", a "Daily" execution range, a time value of "06:00", and an "Active" state. There is an "ADD NEW TASK" button and a "Trigger Module" dropdown set to "Opportunities".

The typical scenario would be to run this for example on a daily basis, early in the morning (6 am in the example).

The screenshot shows the "Conditions" configuration panel. It has a table with columns: Logical Op., Database Field, Check, Changed?, Operator, First Parameter, and Second Parameter. The first row has "OR" as the logical operator, "Date Modified" as the database field, "New Val" as the check, and "This" as the operator. The second row has "Date Modified" as the database field, "New Val" as the check, and "Last" as the operator. The "First Parameter" is "Day" and the "Second Parameter" is "1".

In this example, we are only taking into account those opportunities that have been modified the same day or the previous day (see also filtering parameter in the PHP custom code). This first filter is just to limit the number of opportunities that need to be handled. Also, note that this is just a coding example, as you could perform the same process with vReports in a simpler way.

In the PHP code you can filter a bit finer using the parameter:
`$hours = 24; // Show last $hours of the change-log`

Note that if you expand this, you may have to expand also the condition filter in the event.

You can define what fields should not be included in the report.
`$avoid_fields = Array('probability', 'type');`

For the distribution time, the timezone of the default admin user is used (`id='1'`):
`$theUser->retrieve('1');`

A table is generated with all corresponding changes and this is loaded in a custom variable:
\$custom_variables['last_changes'] = \$result;

In the email template, the name of the opportunity (`{bean->name}`) is included in the name of the email-template (subject of the email). And as main-body, the custom variable "last_changes" is used. Note the different naming convention for PHP and email-templates for the variables.

EDIT

SAVE CANCEL

Name: * [WFM][Notifications][Opportunit] Type: --None-- * Indicates required field

Assigned to: [] [] []

Description: []

Insert Variable: Account: [] Name: [] \$account_name

Subject: [WFM][Notifications][Opportunities]{\$bean->name} Change log

Width Default: 600

Body:

Add your headline here..

Change log (last 24 hours):
`${c_var->last_changes}`

EDIT PLAIN TEXT

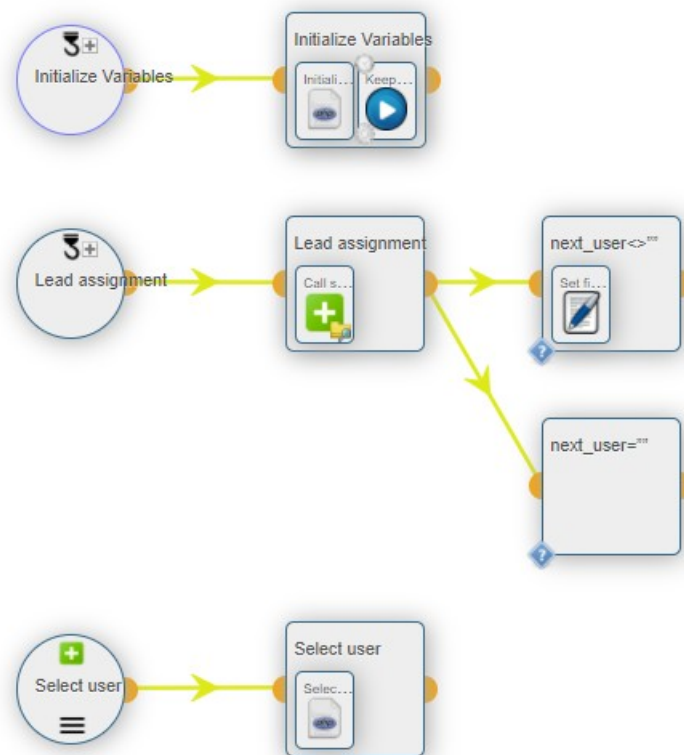
Attachments: [Seleccionar archivo] Ningún archivo seleccionado

11.6 Lead assignment

This is a more sophisticated Process. It will assign new leads to users (agents) based on different types of definable algorithms. For each case, a group of potential agents is identified and then the lead is assigned based on a round robin (circle around) process. The system will check if the agents have logged in during the day before assigning the lead. The login_audit process must be also active (you can potentially deactivate this requirement: see later in this section).

The agent selection algorithms available are:

- Static list: you can define the list of agents by using their username.
- Role: just assign one specific role to the agents (yes, that's all).
- Postcode: You can assign group of agents to certain postal code (group)s. The system supports (and assumes) hierarchical postcodes. Examples are provided later.



The process consists of three events:

- Initialize variables: this part takes care of the initialization of the global variables. Selection of algorithm, selection of agents, etc. These variables can be configured by the administrator.
- Lead assignment: this is the main flow that is triggered when a lead is created. It calls a subprocess that selects the “next agent” to be picked.
- Select user: this subprocess selects the next user, based on the settings and current status.

Initialization

In this example, we have put all the data definition in a php custom task, instead of using multiple “add custom variable” tasks. Therefore we define first a new array: “GLOBAL_CVARS”, within \$custom_variables.

You have to define what algorithm should be used to select the next agent. In the code below, “static list” is chosen.

Note that for agents (including the default one) the username is used.

Also, you have to fill in your own data and comment/de-comment to select the data you want to use.

```
<?php

wfm_utils::wfm_log('vde_debug', '$lead_routing initialized', __FILE__, __METHOD__, __LINE__);

$custom_variables['GLOBAL_CVARS'] = Array();

// comment and de-comment according to your needs. Only one lead_routing algorithm will be used
//i.e. only one should not be commented.

$custom_variables['GLOBAL_CVARS']['default_lead_routing_c'] = 'static_list';
//$custom_variables['GLOBAL_CVARS']['default_lead_routing_c'] = 'role';
//$custom_variables['GLOBAL_CVARS']['default_lead_routing_c'] = 'postalcode';

//Keeps task selected agent.
$custom_variables['GLOBAL_CVARS']['pointer'] = 0;

//used for static_list
$custom_variables['GLOBAL_CVARS']['user_list'] = Array('Agent1', 'Agent2', 'Agent3', 'Agent4', '1');

//if no agent can be selected, the default user will be assigned
$custom_variables['GLOBAL_CVARS']['default_user'] = 'admin';

//list of roles that should be taken into account to select agents.
$custom_variables['GLOBAL_CVARS']['applicable_roles'] = Array('Agent');
/*

//data used for postalcode_mapping. Define as needed.
/*
$custom_variables['GLOBAL_CVARS']['postalcode_mappings'] = Array(
    '73207' => Array('Agent3'),
    '732' => Array('Agent2'),
    '73' => Array('Agent4', 'Agent1'),
);
*/
?>
```

Lead assignment

This is a simple process-branch. It calls the sub-process to select the next agent. If none has been chosen, it does nothing (potentially it could be expanded to launch an alarm every now and then...) or otherwise it assigns the lead to the selected agent. Note that in the flow we now use the variable in the vWFM format: “\${c_var->GLOBAL_CVARS->next_user}” (see variable section for more data).

Selection of user

The code is straightforward. Based on the selection algorithm:

- Agent-id list is created
- If there are selectable agents
 - The first one in the list, counting from the previously selected agent (identified by “pointer”), that HAS logged in that day and has not explicitly logged-out, will be selected.
- If there are no selectable (or available) agents, the default agent is chosen (that can be empty).

If you want to delete the dependency from the login_audit process, change the function:

```
function appliesLoginConditions($test_user) {

    $user_logged_in_today = checkUserLoggedInToday($test_user);
    // wfm_utils::wfm_log('vde_debug', '$user_logged_in_today=[.var_export($user_logged_in_today,
true).]', __FILE__, __METHOD__, __LINE__);
    if (!$user_logged_in_today) {
        return false;
    }

    $user_last_action_is_not_logout = checkUserLastActionIsNotLogout($test_user);
    // wfm_utils::wfm_log('vde_debug',
'$user_last_action_is_not_logout=[.var_export($user_last_action_is_not_logout, true).]', __FILE__,
__METHOD__, __LINE__);
    if ($user_last_action_is_not_logout) {
        return true;
    }

    return false;
}
```

To:

```
function appliesLoginConditions($test_user) {

    return true;
}
```

There are shorter ways to do this, but, you may want to include your additional criteria in the future.

12 Admin tools

You can find the admin tools in the CRM administrator page, in the vWFM section.

ValeDale Work Flow Manager
Configuration section for ValeDale Work Flow Manager

ValeDale WorkFlowManager	Manage your Work Flow Manager Processes
ValeDale WFM database-cleanup	Clean unnecessary data
ValeDale WFM monitor	Monitoring screen for manage current working nodes
ValeDale WFM rebuild-logic_hooks	Rebuild logic_hooks; required for using modules installed after the WFM
Validate Configured Features	Validate Configured Features
Repair php-custom files	Repair php-custom files
About WFM	About WFM

12.1 Validate Configured Features

Performs a number of checks on how vWFM is configured.

ValeDaleWFM Module Dependencies (-required modules)

- ValeDale Common Base*	ValeDaleCommonBase v1.0.3 is installed correctly [OK]	- ValeDale Domains:	ValeDaleDomains is NOT installed [NotSet]
- ValeDale Ajax Post Requests IE10 Compatibility:	ValeDaleAjaxPostRequestIE10Compatibility is NOT installed [NotSet]	- ValeDale Publish HomePage:	ValeDalePublishHomePage is NOT installed [NotSet]
- fix_sugarcrm_module_selfReferencing_bug*:	fix_sugarcrm_self_referencing v4 is installed correctly [OK]		

PHP Function Dependencies

- curl_init:	- curl_init [OK]
--------------	------------------

ValeDaleWFM Files Access

- ValeDale WFM's CSS Files:	- ValeDale WFM's JavaScript Files:	
- vde_activity_style.css [2000 OK]	- vde_events.js [2000 OK]	- vde_activity.js [2000 OK]
- vde_events_style.css [2000 OK]	- vde_events_style.js [2000 OK]	- vde_events.js [2000 OK]
- vde_jspostrequests.css [2000 OK]	- module_fields.js [2000 OK]	- module_fields.js [2000 OK]
- tabs.css [2000 OK]	- activatables.js [2000 OK]	- activatables.js [2000 OK]
- codeeditor.css [2000 OK]	- common_event_activity_task.js [2000 OK]	- common_event_activity_task.js [2000 OK]
- docs.css [2000 OK]	- jquery-rc.js [2000 OK]	- jquery-rc.js [2000 OK]
- jquery.ui.min.css [2000 OK]	- jquery.ui.plumb.min.js [2000 OK]	- jquery.ui.plumb.min.js [2000 OK]
- jquery.yella.min.css [2000 OK]	- jquery.yella.min.js [2000 OK]	- jquery.yella.min.js [2000 OK]
- flowChart.css [2000 OK]	- cfile.js [2000 OK]	- cfile.js [2000 OK]
- vde_process_style.css [2000 OK]	- codeeditor.js [2000 OK]	- codeeditor.js [2000 OK]
- vde_task_style.css [2000 OK]	- css.js [2000 OK]	- css.js [2000 OK]
	- htmlmin.js [2000 OK]	- htmlmin.js [2000 OK]
	- jasonor.js [2000 OK]	- jasonor.js [2000 OK]
	- matchbrackets.js [2000 OK]	- matchbrackets.js [2000 OK]
	- php.js [2000 OK]	- php.js [2000 OK]
	- vml.js [2000 OK]	- vml.js [2000 OK]
	- jquery.ui.min.js [2000 OK]	- jquery.ui.min.js [2000 OK]
	- jquery.yella.min.js [2000 OK]	- jquery.yella.min.js [2000 OK]
	- ZeroClipboard.js [2000 OK]	- ZeroClipboard.js [2000 OK]
	- vde_task.js [2000 OK]	- vde_task.js [2000 OK]

ValeDaleWFM Schedulers

- wfm_scheduled_task:	- wfm_engine_crontab:	- wfm_engine_crontab [OK]
-----------------------	-----------------------	---------------------------

ValeDaleWFM Community Features

- WFM_site_url:	- WFM_site_url [NotSet]	- WFM_site_login_username_password:	- WFM_site_login_username_password [NotSet]
- WFM_translateLabels:	- WFM_translateLabels [NotSet]	- WFM_get_fields_from_db:	- WFM_get_fields_from_db [NotSet]
- WFM_sugarcrm_emailTemplate_charset:	- WFM_sugarcrm_emailTemplate_charset [NotSet]	- WFM_MAX_loops:	- WFM_MAX_loops [NotSet]
- WFM_MAX_working_nodes_executed_in_one_php_instance:	- WFM_MAX_working_nodes_executed_in_one_php_instance [NotSet]	- WFM_changeLogLevelFromveto:	- WFM_changeLogLevelFromveto [NotSet]
- WFM_nonVisibleFields:	- WFM_nonVisibleFields [NotSet]	- WFM_CanSeedDomainField_Roles:	- WFM_CanSeedDomainField_Roles [NotSet]
- WFM_use_alternative_ListView:	- WFM_use_alternative_ListView [NotSet]	- WFM_disable_wfmhook:	- WFM_disable_wfmhook [NotSet]
- WFM_disable_wfm_completely:	- WFM_disable_wfm_completely [NotSet]	- WFM_disable_workFlowManagerEngine:	- WFM_disable_workFlowManagerEngine [NotSet]
- WFM_disable_wfmScheduledTask:	- WFM_disable_wfmScheduledTask [NotSet]		

The main configurations are explained in the following sections.

12.2 Module Dependencies

vWFM requires ValeDale Common base and the self-referencing bug-fix modules.

12.3 cURL

cURL utility must be installed unless vWFM is configured not to use it.

12.4 Files Access

vWFM uses both css and javascript files. If any of these files is missing then the vWFM might not work as it should be.

12.5 vWFM schedulers

The vWFM needs two schedulers in order to execute the workflows.

12.5.1.1 vWFM_scheduled_task

Needed for workflows with vWFM-events of trigger_type=scheduled. If this scheduler is not setup then you cannot use scheduled-tasks.

12.5.1.2 vWFM_engine_crontab

This scheduler will execute the vWFM. It will look into the vWFM-working-memory and it will execute the working-nodes. For example: if you have defined a workflow with delays then you need vWFM-engine-crontab scheduler in order to execute delayed vWFM-working_nodes.

Note: the vWFM-working-memory will be executed also if a logic_hook is triggered.

12.6 Config Features

There are a number of advanced features that can be set in the config_override.php file.

An example of config_override.php is included in the zip package. Note that you should keep any existing elements in your config_override.php, so do not overwrite your existing file with this one. You can just copy the relevant sections and add them to the existing file.

Note that vWFM will work without defining these parameters (using some default values).

Config parameter	Usage
WFM_site_url	vWFM will use the CRM's default site_URL. This may not work for Virtual Hosts set-up, using cURL. <pre>\$sugar_config['WFM_site_url'] = 'https://[your_server_url]/[your_sugarcrm_instance_name]';</pre> For example: <pre>\$sugar_config['WFM_site_url'] = 'https://www.ValeDale.com/mysugarcrm';</pre>
WFM_site_login_username_password	You need to define this, if you use basic authentication for your site. <pre>\$sugar_config['WFM_site_login_username_password'] = '[site_login_username]: [site_login_password]';</pre> Leave out the brackets "[]"
WFM_SystemCurlUsage	Only use this in Linux Instead of using curl through php-function 'curl' (lasts: 250 ms at least, at the most 1 s), you can use curl through command-line shell. It takes: 1 ms. Default is false. <pre>\$sugar_config["WFM_SystemCurlUsage"] = false;</pre>

WFM_TranslateLabels	<p>If you don't want the labels of the module fields to be translated, set this to false. This will slightly improve performance while editing. Default is true.</p> <pre>\$sugar_config['WFM_TranslateLabels'] = true;</pre>
WFM_get_fields_from_db	<p>Some fields are not included in the bean by default by the CRM. (For example: module 'Cases' has the field 'case_number', this field has the property autoincrement. So when you save a record, you do not have the case_number in the bean, the case_number is autoincremented in database, and it is not sent back to the bean. You can define here what fields should be retrieved to make them accessible to vWFM.</p> <pre>\$sugar_config['WFM_get_fields_from_db'] = array ('[moduleTable1]' => array ('[field1]', '[field2]'), '[moduleTable2]' => array ('[fielda]', '[fieldb]', '[fieldc]'),);</pre>
WFM_sugarcrm_emailTemplate_charset	<p>SugarCrm 641 uses ISO-charset for emailTemplates. As of SugarCrm 656, the CRM uses UTF-charset. If you experience that your emails show some weird characters use the following \$sugar_config:</p> <pre>\$sugar_config['WFM_sugarcrm_emailTemplate_charset'] = 'ISO';</pre>
WFM_MAX_loops	<p>In order to avoid infinite loops. Infinite loops can for example occur when the user defines a process that performs an action that triggers its execution (On Modify, On creation, ...). The system assumes a default value of "10".</p>
WFM_MAX_working_nodes_executed_in_one_php_instance	<p>This setting is to avoid error "Query limit of 1000 reached for Home module.", which makes the system crash. Although you can increase the CRM limit (check out Internet blogs), we also have included this protection. Default is 10. The value 'unlimited' is also supported (but not recommended).</p> <p>When the limit is reached, vWFM will send a curl request to continue the wfm-execution and so avoid the error. This does result in a slight performance impact.</p>
<p>Deprecated: WFM_changeLogLevelFromvdeTo WFM_changeLogLevelFromFlowDebugTo WFM_log_to_independent_file</p>	<p>vWFM uses the function "wfm_utils::wfm_log" to send data to log-files. function wfm_log(\$logLevel, \$logText, \$file, \$function=null, \$line=null) The settings are used to change: \$loglevel for the cases 'flow_debug' and 'vde_debug' to anything else than "Debug" (i.e. as default Sugar settings must be set to debug for logging to take place in case of those two values) If "WFM_log_to_independent_file" is set to true, logs will be sent to wfm.log instead of "sugarcrm.log".</p>
WFM_NonVisibleFields	<p>If you do not want to present some fields in the module_fields selection boxes.</p> <pre>\$sugar_config['WFM_NonVisibleFields'] = array("[module_name_1]" => array("[field_1]", "[field_2]"), "[module_name_2]" => array("[field_1]", "[field_2]", "[field_3]"),);</pre>
<p>WFM_development_mode WFM_development_mode_allowed_emails WFM_development_mode_notAllowedEmails_textAddedToEmailAddress</p>	<p>If you are using vWFM in a development environment, you don't want users to get email notifications. In order to do so, you have to set: \$sugar_config['WFM_development_mode'] = true;</p> <p>You can now define email addresses explicitly that should be excluded from the email manipulation: \$sugar_config['WFM_development_mode_allowed_emails'] = Array('name1@server1.com', 'name2@server2.es');</p> <p>All other email addresses will be transformed to: name3@server.com -> name3@XWFMnotAllowedEmailXserver.com</p> <p>You can also define your own translate string using the setting: WFM_development_mode_notAllowedEmails_textAddedToEmailAddress And setting this to another string than 'XWFMnotAllowedEmailX'</p>
WFM_useTinyMCE	<p>Some installations change the textarea of some vWFM fields to WYSIWYG fields. If you experience these kind of issues set:</p> <pre>\$sugar_config['WFM_useTinyMCE'] = false;</pre>
WFM_sugarcrmEmailTemplateBody_doNotExecute_nl2br	<p>Textarea encodes line-feeds as \n, but html encodes line-feeds as
. If a field(wfm-variable) that is a textarea is used in an email_template, wfm needs to execute the php-function nl2br.</p>

	<p>Sometimes textarea uses directly
. {ckeditor, tynimce, etc}, in this case you need to set this sugar_config to true.</p> <pre>\$sugar_config['WFM_sugarcrmEmailTemplateBody_doNotExecute_n12br'] = false;</pre>
WFM_disable_wfm_completely	Allows you to disable vWFM.
WFM_disable_wfmHook	Processes will not be executed
WFM_disable_wfmScheduledTask	ScheduledTasks will not be executed.
WFM_disable_workFlowManagerEngine	vWFM engine is not executed.
WFM_enable_async_sugar_job_queue WFM_disable_async_curl	When you import a large number of objects, you may encounter performance issues due to delays in the use of cURL. Set these two parameters to true to avoid this. Note that async_curl cannot be used while you have the second parameter set to true. If you are using async_curl in your workflows, ensure that you perform these temporary changes when no WFs are executed.
External databases	<p>You can define access to multiple external databases. Domain array should always be empty (supported for backwards compatibility). See also vReports for more information.</p> <pre>\$sugar_config['WFM_AlternativeDbConnections'] = array(0 => array("vdeReportsDbAddress" => '192.168.0.X', //Ip address "vdeReportsDbUser" => "root", //Database access username "vdeReportsDbPassword" => "", //Database access password "vdeReportsDbName" => "ExternalDb_A", //Database name "vdeReportsDbPort" => "3306", //Port "vdeAllowedTables" => array("instance" => array("tableA", "tableC"), "domain" => array(),), "vdeForbiddenTables" => array("instance" => array(), "domain" => array(),),), 1 => array("vdeReportsDbAddress" => '192.168.0.Y', "vdeReportsDbUser" => "root", "vdeReportsDbPassword" => "", "vdeReportsDbName" => "ExternalDb_B", "vdeReportsDbPort" => "3307", "vdeAllowedTables" => array("instance" => array(), "domain" => array(),),),);</pre>
\$sugar_config['resource_management']['default_limit']	<p>This standard CRM setting is related to: 'WFM_MAX_working_nodes_executed_in_one_php_instance'</p> <p>You may want to increase this setting, but in general it is recommended not to do so.</p> <pre>\$sugar_config['resource_management']['default_limit'] = 1000;</pre>

12.7 vWFM database-cleanup

In your development environment you may encounter situations where your WorkFlow breaks, leaving elements “in the air”. This, of course can also happen in an active environment.... ☹

This section describes how and what you can clean. Please be especially thoughtful before you perform and clean-up action.

Go to CRM administrator page and click on “ValeDale vWFM database-cleanup”.

Clean WFM

Choose.

<input checked="" type="checkbox"/> Clean WFM definitions
<input type="checkbox"/> Clean deleted=1 wfm-entities ({wfm-process, wfm-event, wfm-activity, wfm-task}) and relationships between them.
<input type="checkbox"/> Clean unrelated wfm-entities (example: wfm-activity does not belong to any wfm-event).
<input checked="" type="checkbox"/> Clean WFM working tables
<input type="radio"/> Clean broken working-nodes (status=executing and older than 1 hour).
<input type="radio"/> Clean wfm working-tables ({wfm-proces_instances, wfm-working_nodes, wfm-on_hold}).
<input checked="" type="checkbox"/> Others
<input type="checkbox"/> Clean deleted=1 login_audit.
<input type="checkbox"/> Clean deleted=1 email_templates.

Clean WFM [Warning: You should remove temporarily (not mandatory) wfm-modules' logic-hooks ({wfm-process, wfm-event, wfm-activity, wfm-task}). Go to Administration->"AlneaSol WFM rebuild-logic_hooks".]

Examples of “bad scenarios”:

You have defined a workflow that sends an email. If the email_server does not reply to the vWFM request for sending an email, the vWFM-working_node that controlled the workflow-execution-time will end up in a status=executing, forever. With other words, if some resource the vWFM uses does not work properly, the workflow-execution might not end up well. Note that this will not affect other activities. So, in order to clean this garbage, you can use this facility.

If you choose 'Clean vWFM working-tables' this facility will remove all the vWFM-working- memory, both crashed-workflow-executions and correct-workflow-executions. If you want to remove only the crashed-workflow-executions you need to go to the vWFM monitor. If you have access to your DB, you may also remove them with your own DB-manager (phpMyAdmin, etc).

12.8 vWFM monitor

Go to CRM administrator page and click on “ValeDale vWFM monitor”.

This facility will allow you to monitor the vWFM-execution-time, as well as removing those workflow-execution records that you like.

The vWFM-monitor covers three modules.

12.8.1 Process Instances

Below you can see an example of Process Instances:

Search WFM Process Instances [+ Create](#)

Name	Process	Parent Process Instance Id	Bean ungreedy_count	Date Created	Date Modified
p_i_3d53fd3b-d6a6-5acc-fa61-52318fe8c6b0	P2		0	12/09/2013 11:56	12/09/2013 11:56
p_i_9accdea5-20bc-89f7-6816-52318fee3f93	P1		0	12/09/2013 11:54	12/09/2013 11:54
p_i_53803e3c-64d5-4ea4-870e-522a00e197bc	P1		0	08/09/2013 18:18	08/09/2013 18:18

vWFM-Process_Instances fields:

- Name (internally generated)
- Process

This is the name of the Process (Workflow) you gave the process.
- Parent Process Instance Id

When One (parent) process calls a Sub Process, the Parent-process instance ID of the parent will appear in the entry of the sub-process.
- Bean ungreedy_count

This counter is increased one by one when a workflow-execution results in another workflow-execution. It is used by the vWFM to avoid infinite-loops.

Example increments:

 - A parent process calls a sub-process.
 - trigger_event=on_modify, task_type=modify_object
 - trigger_event=on_create, task_type=create_object with object Module=trigger_module

12.8.2 Working Nodes

Below you can see an example of working_nodes:

Search WFM Working Nodes + Create

Status: Not Started Executing Delayed By Activity Delay By Task In Progress Terminated Search Clear Advanced Search ?

Name	Process Instance Id	Priority	Event	Current Activity	Trigger Module	Executing Object	Iter Object	Current Task	Delay Wake-up Time	Status	Date Created	Date Modified
<input type="checkbox"/> w_n_3d58534a-972d-a21f-1d8e-52318f66da6c	p_l_3d53fd3b-06a6-5acc-fa61-52318fe8c6b0	-4	E1 P2	A1 E1 P2	Opportunities	Opp2	0	-	12/09/2013 11:57	Delay By Task	12/09/2013 11:56	12/09/2013 11:56
<input type="checkbox"/> w_n_5384cd3f-6c87-1f85-4b2c-522a006148c2	p_l_53803e3c-64d5-4ea4-870e-522a00e197bc	0	E1 P1	A1 E1 P1	Accounts	Acc1	0	T2 A1 E1 P1	-	Held	06/09/2013 18:18	06/09/2013 18:18
<input type="checkbox"/> w_n_9ad1a563-7527-52dc-40cd-52318f77e04a	p_l_9accdea5-20bc-897-6816-52318fee3f93	0	E1 P1	A1 E1 P1	Accounts	Acc1	0	T2 A1 E1 P1	-	Terminated	12/09/2013 11:54	12/09/2013 11:56
<input type="checkbox"/> w_n_9adc41d8-f9c2-802c-9308-52318fd32762	p_l_9accdea5-20bc-897-6816-52318fee3f93	0	E1 P1	A2 E1 P1	Accounts	Acc1	0	-	12/09/2013 12:54	Delayed By Activity	12/09/2013 11:54	12/09/2013 11:54

vWFM-Working_Nodes fields:

- Name
- Process Instance Id

This is the id of the process_instance that owns the working_node.

- Priority

```
//
vWFM_working_node_priority
['logic_hook']['start'] = 0;
['subprocess'] = -1;
['logic_hook']['intermediate'] = -2;
['logic_hook']['cancel'] = -3;
['scheduled'] = -4;
```

- Event

The vWFM-event that triggered the execution of vWFM-process (creation of a process_instance).

- **Current Activity**
Current vWFM-activity in execution.
- **Trigger Module**
Executing Object's module.
- **Executing Object**
We need to differentiate two cases:
 - a) `trigger_type=logic_hook`
The `executing_object` will be the object that triggered the execution of the workflow. Ex: The user modifies an Account and there is a workflow with `trigger_type=logic_hook` and `trigger_event=on_modify`.
 - b) `trigger_type=scheduled`
Several objects will be "executed" (actually the object is not executed, the object is being processed). Each object that complies with the applicable conditions will be executed one by one.
- **Iter Object**
Indicated the nr of the object being processed. If n objects meet the condition, then the value can be {0,1,...,n-1}
- **Current Task**
Current vWFM-task in execution.
- **Delay Wake-up Time:** When a vWFM-activity or a vWFM-task has a delay, the vWFM stops the workflow-execution and once the delay time is elapsed the vWFM will resume the workflow-execution. So, the `delay_wakeup_time` will show you when the vWFM will resume the workflow-execution.
- **Status**

Status	Description
not_started	When a working-node is created, vWFM sets status=not_started.
executing	vWFM is working with this working-node. You should not see this status in the vWFM-monitor. If you see a working-node with this status, it is very likely that the vWFM faced a problem with its execution. This problem could be that the email server is not responding to the vWFM requests, but it could be other causes too. If a working-node is in status=executing for long then feel free to remove this working-node.
delayed_by_activity	If a vWFM-activity has a delay then the vWFM will set status=delayed_by_activity.
delayed_by_task	If a vWFM-task has a delay then the vWFM will set status=delayed_by_task.
in_progress	The vWFM will set this status in order to resume the workflow-execution.

	<p>Ex1: If vWFM-task executed is the last one and there is only one next_activity.</p> <p>Ex2: When a working-node is in status=held and then the user clicks on close-button in a Suitecrm-task.</p>
terminated	This working-node has finished its execution.
held	When a vWFM-task needs to be put on hold (only if vWFM-task is a create object of a CRM task/call and the next vWFM-task's delay_type=on_finish_previous_task).

12.8.3 On Hold

Below you can see an example of on_hold nodes:

Search WFM On Hold [Create](#)

Working Node Id Search Clear [Advanced Search](#) ?

(1 - 1 of 1)

Name	Trigger Module	OnHold Object	Process Instance Id	Working Node Id	Date Created	Date Modified
<input type="checkbox"/> o_h_184d6dea-a59f-ef83-e597-522a00f0a1d	Tasks	Son of [Acc:1]	p_i_53803e3c-64d5-4ea4-870e-522a00e197bc	w_n_5384cd3f-6c87-1b5-4b2c-522a006148c2	06/09/2013 18:18	06/09/2013 18:18

(1 - 1 of 1)

OnHold fields:

- Name
- Trigger Module
OnHold Object's module.
- OnHold Object

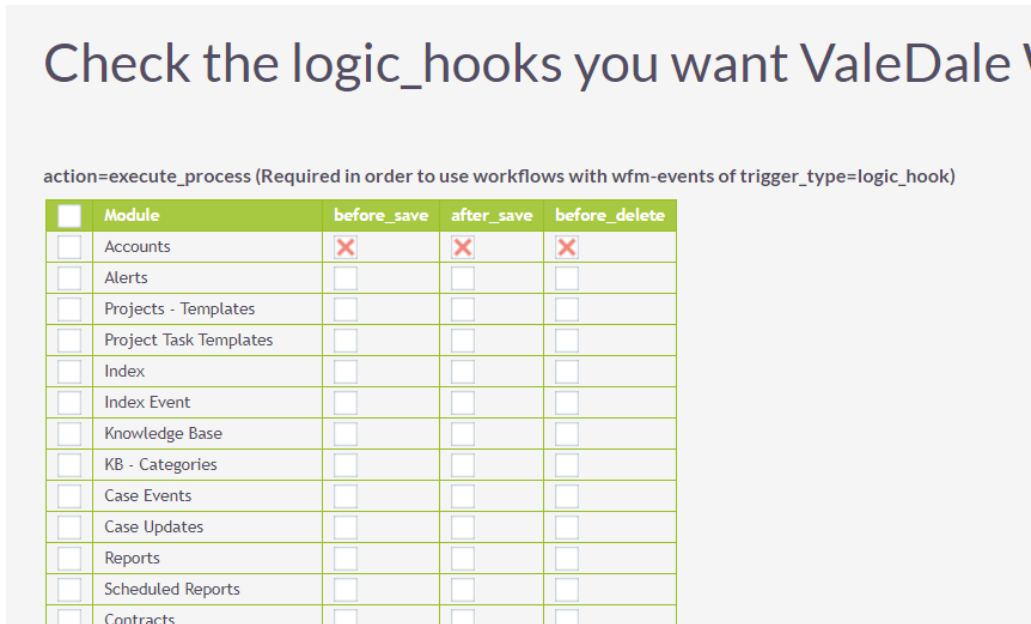
You need to click on its close-button in order to let the working_node continue its workflow-execution. The next php-condition is checked:

```
((($object_module == "Calls")&&($bean->status == "Held")) || (($object_module == "Tasks")&&($bean->status == "Completed")))
```

- Process Instance Id
This is the id of the process_instance that owns the onHold-node.
- Working Node Id
This is the id of the working_node that owns the onHold-node.

12.9 vWFM rebuild-logic_hooks

Go to Suitecrm administrator page and click on “ValeDale vWFM rebuild-logic_hooks”. The browser will show you the rebuild-logic_hooks facility.



Check the logic_hooks you want ValeDale

action=execute_process (Required in order to use workflows with wfm-events of trigger_type=logic_hook)

<input type="checkbox"/>	Module	before_save	after_save	before_delete
<input checked="" type="checkbox"/>	Accounts	✗	✗	✗
<input type="checkbox"/>	Alerts	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	Projects - Templates	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	Project Task Templates	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	Index	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	Index Event	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	Knowledge Base	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	KB - Categories	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	Case Events	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	Case Updates	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	Reports	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	Scheduled Reports	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	Contracts	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

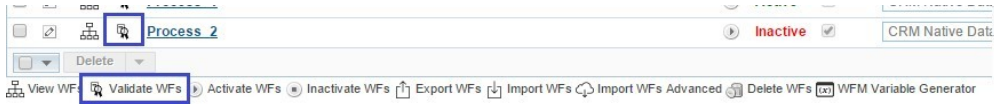
You have to cross (cross = active) the modules that you want to supervise.

NOTE: you have to select also before_save, if you want the after_save logic to work properly. You can select the three logic hooks at the same time by clicking on the entry box in the first column.

Make sure that you hit “Update” at the bottom of the page for the changes to apply.

12.10 Validate WFs

If some workflow does not work then this feature can help you to find out what is wrong.



Validate WorkFlows

Step 1: Choose validations.

<input checked="" type="checkbox"/> All
<input checked="" type="checkbox"/> Task SendEmail references existing EmailTemplate
<input checked="" type="checkbox"/> WorkFlow is active
<input checked="" type="checkbox"/> Logic Hook is set

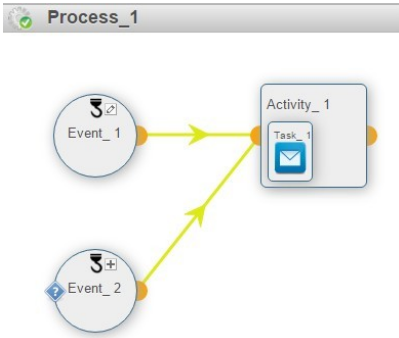
WorkFlow: Process_2

Validation	Result
Task SendEmail references existing EmailTemplate	OK
WorkFlow is active	NOK
Logic Hook is set	OK

Note that this is only a rough indication.

12.11 vWFM-Event duplicity

When one activity is triggered by more than one event (at the same time), the activity will only be executed once.



13 Known Issues

- When selecting an event in "call-process" editview, if filters are used, followed by "Search", the pop-up no longer returns the value of the event. I.e. search mode is now useless. (workaround: don't use search)
- In the graphical editor: icons in the south-panel are sometimes not correctly loaded. (refresh of the south-panel makes them appear)
- Need to send the main process id in vde_task.js when open_popup is called (around line 387) (to limit access to local sub-processes)
- Need to migrate in Variable Generator zeroclipboard (uses Flash) to another "copy to clipboard tool" (e.g. clipboardjs) (now you have to copy manually)
- "Select all" in ListView does currently not work for mass operations. Select all in page, does work.
- For SQL tips: default MySQL information is shown (e.g. Task: "Add Custom Variables"). To get MSSQL tips, copy "`..\modules\vde_Process__common_WFM\mySQLFunctions\msSQLFunctions.nl_nl.html`"
-