Business transaction monitoring

Go back to Pandora FMS documentation index

Transaction monitoring

Introduction

Version NG 7 or posterior.

The transactional server component implemented in this version allows to execute tasks, which are dependent upon one another, following a user-defined design. This means that it is possible to coordinate different executions to test a target at a given time.

The following case study consists of tracking an order that goes through different stages, being able to keep track of the time in each of the steps, departments or stages:

A four-step graph will be defined:

1. *Portal:* Order reception.
2. *EMI01:* Order process.

3. *ORAMON:* Saved in database.
4. *Correo electrónico:* Logistics notification.

Pandora FMS will use this graph along with a series of control scripts to monitor the previously indicated process, visually showing the status in which each one of the parts that make up the business process is at all times.

Below there is an explanation on how to fully monitor a transaction process.

Operation

A **transaction** can be defined as a set of steps that make up a more complex task. These steps will be called **stages**.

The set of stages and their workflow (dependency relations) will define a **transactional graph**.

Pandora FMS, based on the transactional graph, will launch an order to execute control scripts in each of the previously defined stages. These control scripts will perform the monitoring tasks for each stage.

The most common points to check in control scripts that can provide information on the status of a transaction are:

- Entries in log files.
- Presence of temporary files.
- Direct database queries.
- Existence of mail in inboxes.

The transactional system is distributed, being able to deploy as many **transactional agents** in an infrastructure as needed, and it will be Pandora FMS server the one that will communicate with these agents, indicating the steps to follow and the time when they must run a control script.

Configuring a transaction

In order to configure and execute transaction monitoring, these will be necessary:

- Previous analysis of the business process to be monitored:
  - Workflow.
  - Points involved.
  - Control scripts.
- **Transactional agent** deployment in the required equipment to control the information flow.
  - Development and deployment of control scripts in the required transactional agents.
  - Configuration of transactional agents.
- From Pandora FMS console, create the transaction by entering the data in the forms.

Previous analysis

Let us analyze a regular case:

The transaction begins when an order is received on the **web portal**, where a transactional agent must be deployed, or at least on a machine capable of carrying out remote checks. During this first stage, a critical script is executed: the **transaction trigger**.

In the next stage, a virtualization of the order processing is launched in the *EMI01* machine, using different ID codes for the order placed in the previous stage. There must be another transactional agent installed on this machine, or, if not, on a machine capable of executing remote checks. The control script checks that the process has been correctly completed in this stage by searching for registry file entries, events or temporary files.

The third stage of the transaction is to save the processed order on an Oracle database, on the *ORAMON* machine. These machines are usually highly protected and it is difficult to install additional software on them, so a **transactional agent** should be deployed on a remote machine with permissions to launch data base queries.

The last stage is to confirm whether there is mail in the Logistic department's inbox in the *public* mail server. Since an agent cannot be deployed on this machine, the queries will have to be run from another machine that is able to access it. Pandora FMS classic plugins slightly modified can be used to check incoming emails.

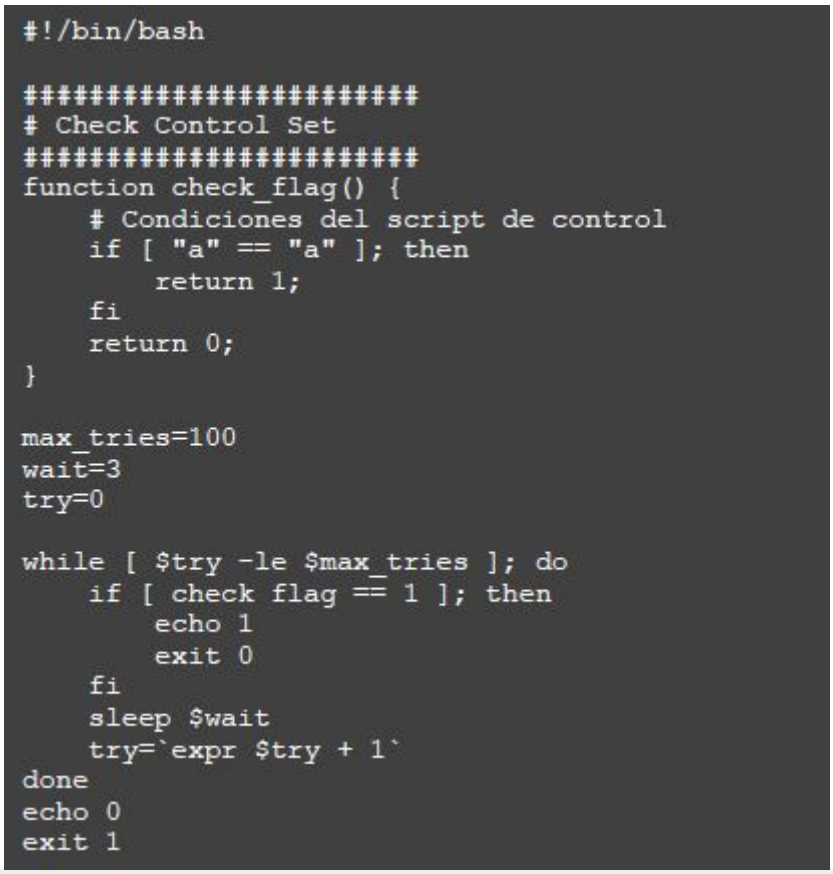
Deployment and development

In this example, there are four stages which require four scripts:

1. Transaction trigger: It makes a virtual order to the **web portal** in order to subsequently trace it.
2. Control script: It looks for a string in the log file.
3. Control script: It queries the data base.
4. Control script: It confirms whether the corresponding email reached the Logistics inbox.

In the above example, the transaction trigger leaves a file copy of an order on a specific point (for example through FTP).

Control scripts share a basic structure as in the following example:



```
#!/bin/bash

########################
# Check Control Set
########################
function check_flag() {
    # Condiciones del script de control
    if [ "a" == "a" ]; then
        return 1;
    fi
    return 0;
}

max_tries=100
wait=3
try=0

while [ $try -le $max_tries ]; do
    if [ check_flag == 1 ]; then
        echo 1
        exit 0
    fi
    sleep $wait
    try=`expr $try + 1`
done
echo 0
exit 1
```

```
#!/bin/bash
########################
# Check Control Set
########################
function check_flag() {
   # Control script conditions
   if [ "a" == "a" ]; then
      return 1;
   fi
   return 0;
}
max_tries=100
wait=3
try=0
while [ $try -le $max_tries ]; do
   if [ check_flag == 1 ]; then
```

```
      echo 1
      exit 0
   fi
   sleep
   $wait
   try=`expr $try + 1`
done
echo 0
exit 1
```

Once the scripts are working correctly, proceed to deploy transactional agents on the necessary machines and devices.

```
tar xvzf pandorafms_transactional.tar.gz
cd pandora_transactional
./pandora_transactional_installer --install
```

You may find the transactional agent in the Pandora FMS module library.

Just modify the agent configuration file (`/etc/pandora/pandora_transaccional.conf`) **to point to the Pandora FMS server that will direct the transaction process**.

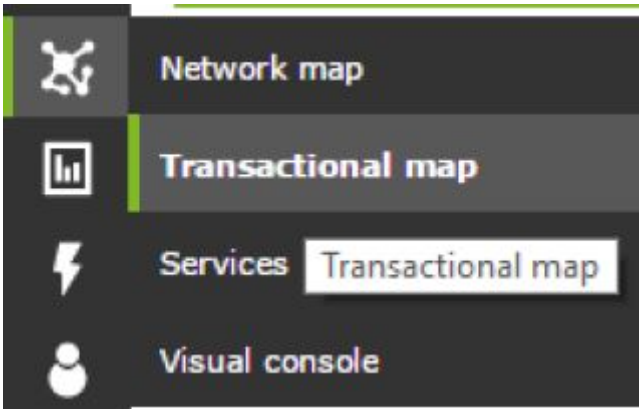
To start the agent's service:

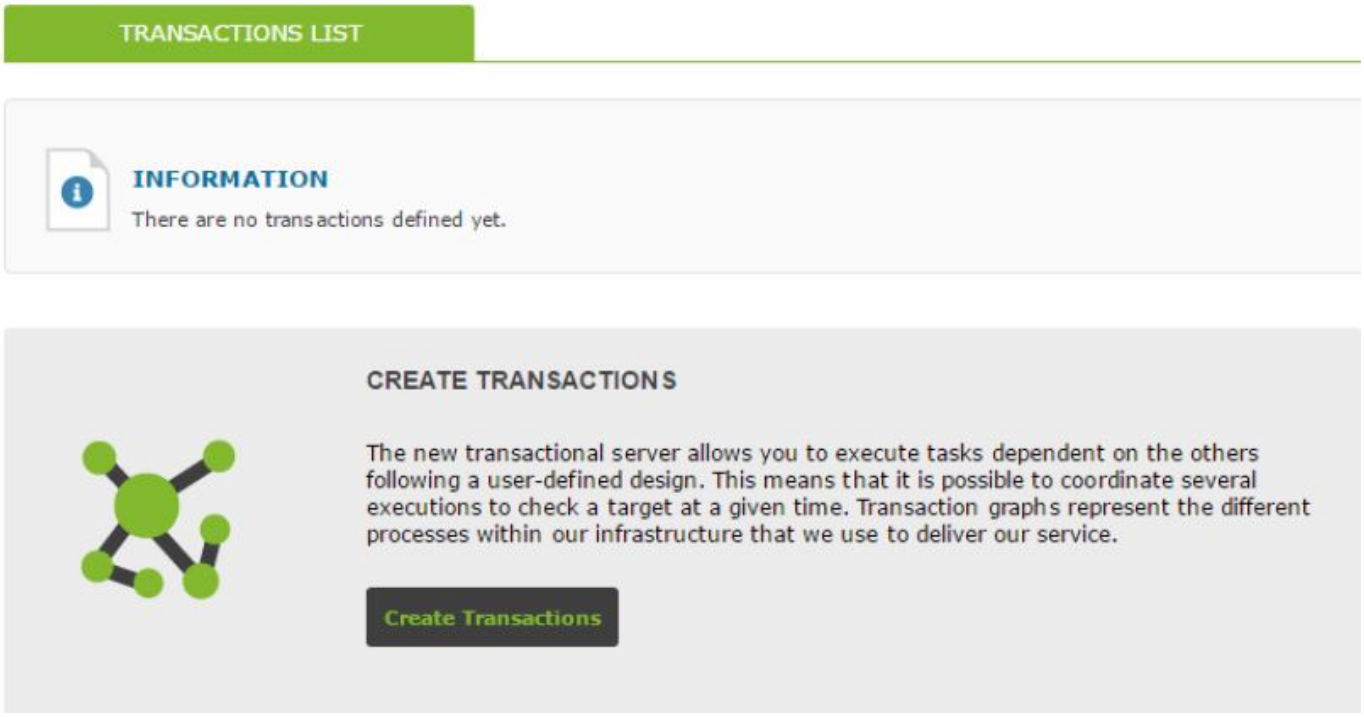
```
/etc/init.d/transactional_daemon start
```

Creating a transaction

Using the Pandora FMS web console.

Go to Maps → Transactional map.



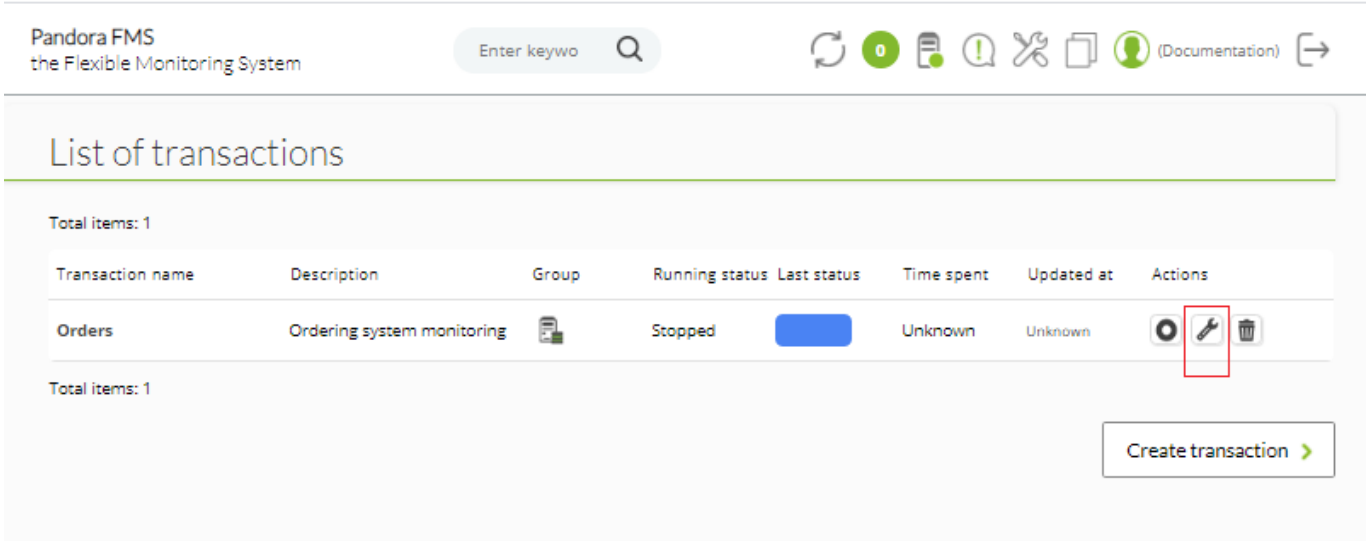


Click on **create new transaction** and fill in the required fields.

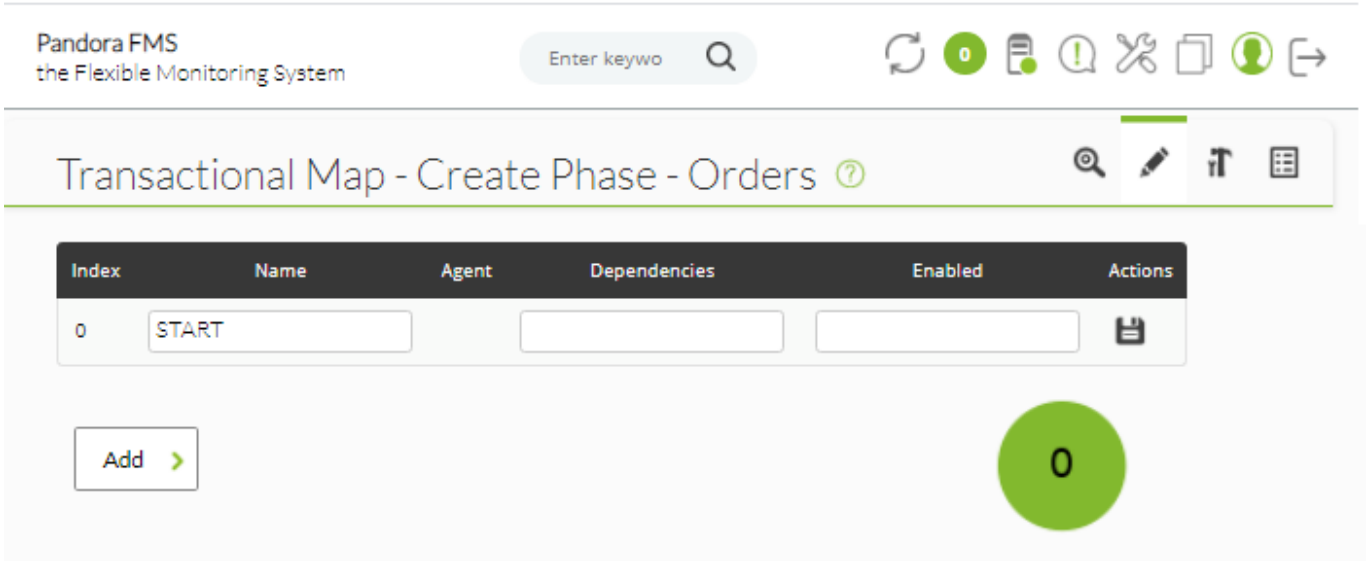
- **Agent**: Pandora FMS agent where the modules generated by the system will be saved (History).
- **Group**: to control visibility and access.
- **Loop interval**: waiting time before launching the transaction again.

Creating the stage tree

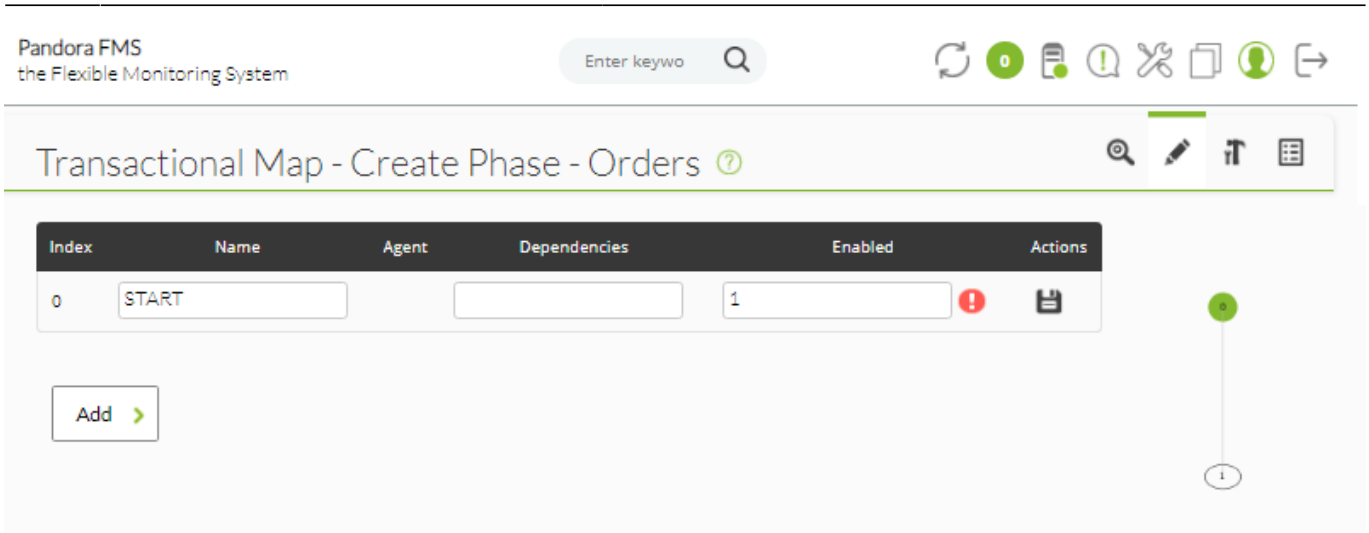
Once the transaction has been created, go to the list click on the settings icon (**Edit phases**) and proceed to shape the phase tree.



Enter the data for each stage:

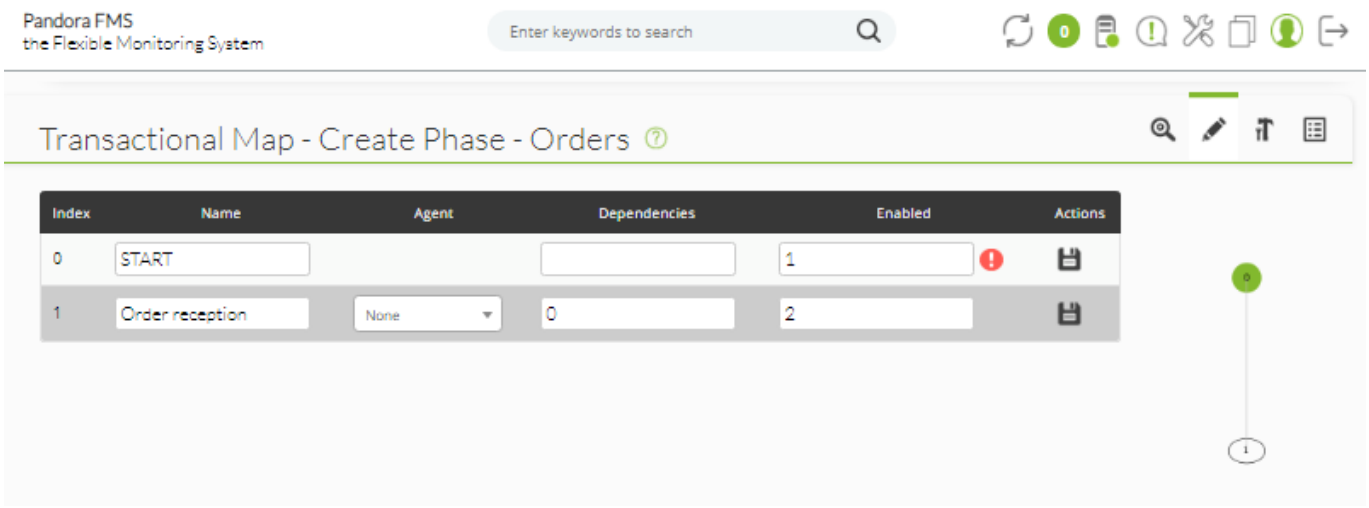


- **Index**: sole ID for the stage of the current transaction.
- **Name**.
- **Agent**: where the corresponding control scripts will be executed.
- **Dependencies**: prior stages that must be completed before that stage begins; indexes are separated by commas.
- **Enables**: stages that are enabled when a stage being edited is completed; the indexes of the stages are separated by commas.
- **Actions**: save changes.

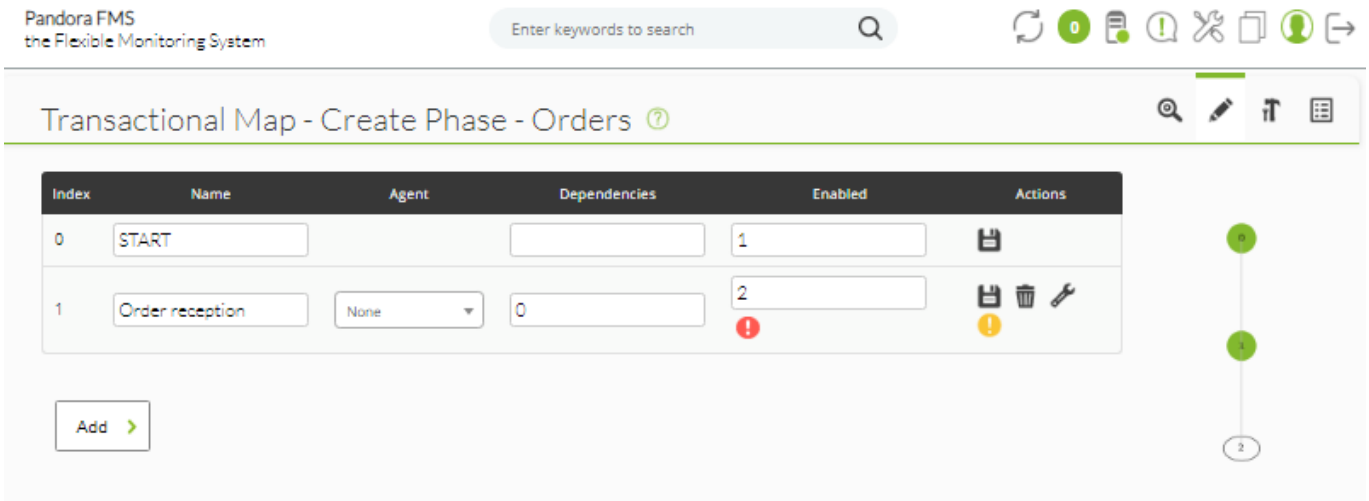


START is the default initial stage, used only to define which stages will be the first to be executed.

Click **Add** to insert the index of the first stage to execute (1) Order received.



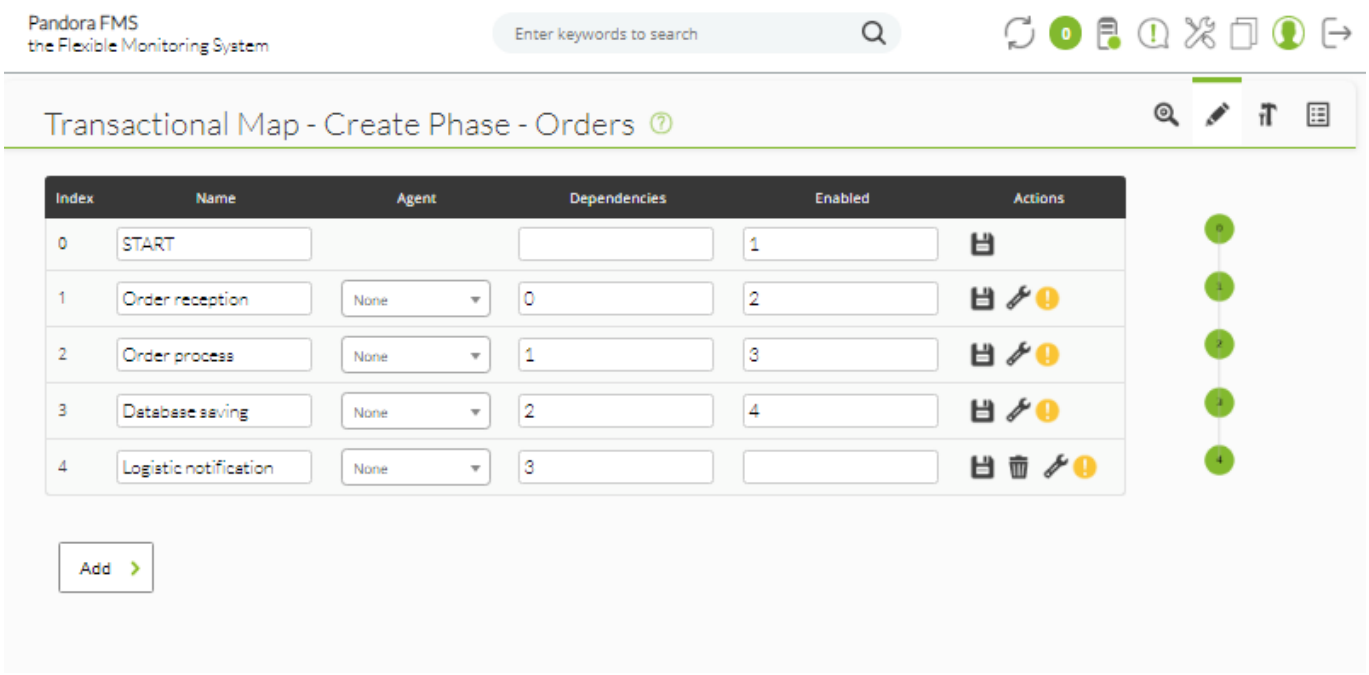
When saving the modification, the stage with error status is marked automatically. That happens because, in the definition, there is no stage with index 1. Click “Add” to create it:



Once the changes are saved, create the stage by updating the previewing of the transaction graph and correcting the previous warning:

Note that in stage 1, dependencies field '0' is shown to indicate that the stage will begin when the **START** stage is successfully finished.

Create all the stages following the same procedure:



The final stage does not enable anything, it means the transaction is finished.

In the screenshot, a yellow warning icon can be seen, indicating there is something yet to be configured (control scripts).

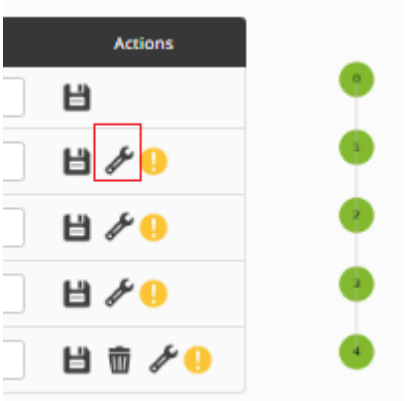
Control scripts configuration

Configure the control scripts associated with each stage, these must have been previously defined to perform the desired checks and maintain a specific basic structure.

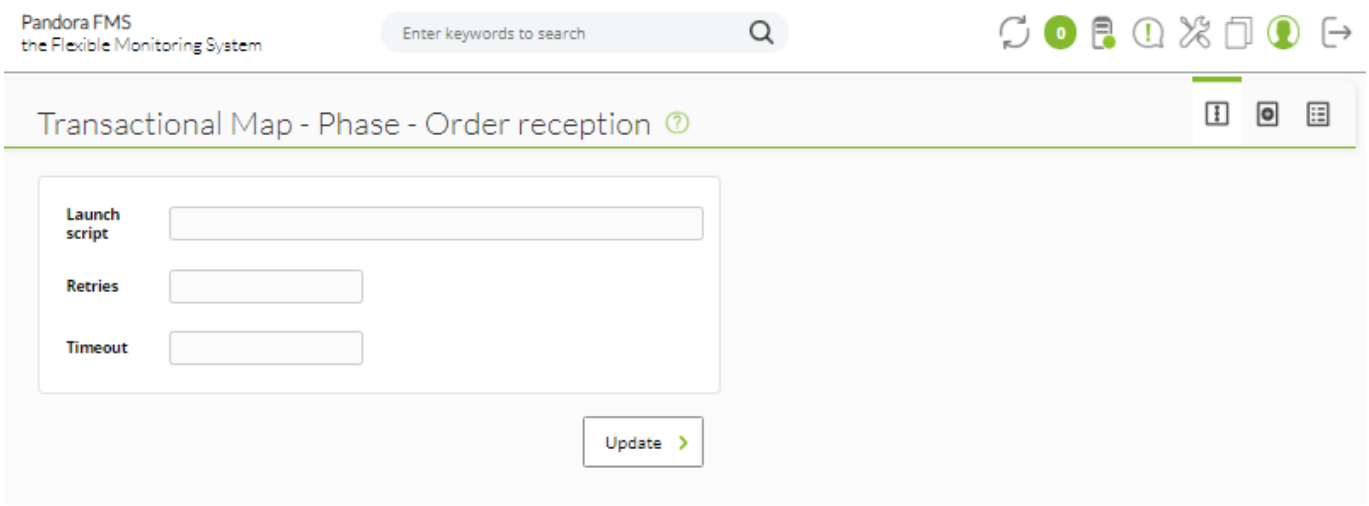
The standard output (STDOUT) of the script will determine the central value shown in the stage, while the status (correct or incorrect) will be determined by the *result of the execution* of the script itself.

The result of the execution (STDERR , also called **errorlevel**, or **execution code**) is different from the standard output (STDOUT) of the *script* (the value it returns by screen when executing it). The execution result or **errorlevel** can be checked executing `echo $?` in Linux systems and `echo %ERRORLEVEL%` in Windows® systems.

With this in mind, access the form using the edit icon:

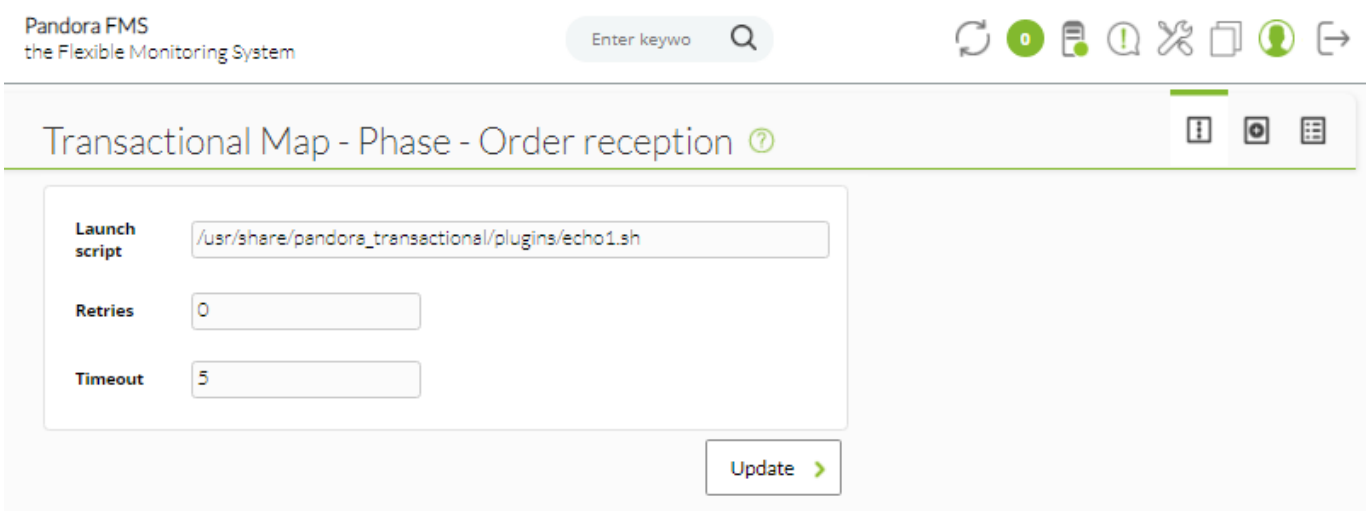


In the configuration form of the control scripts, the command to be executed, the number of retries and the maximum time of execution allowed for the indicated script:

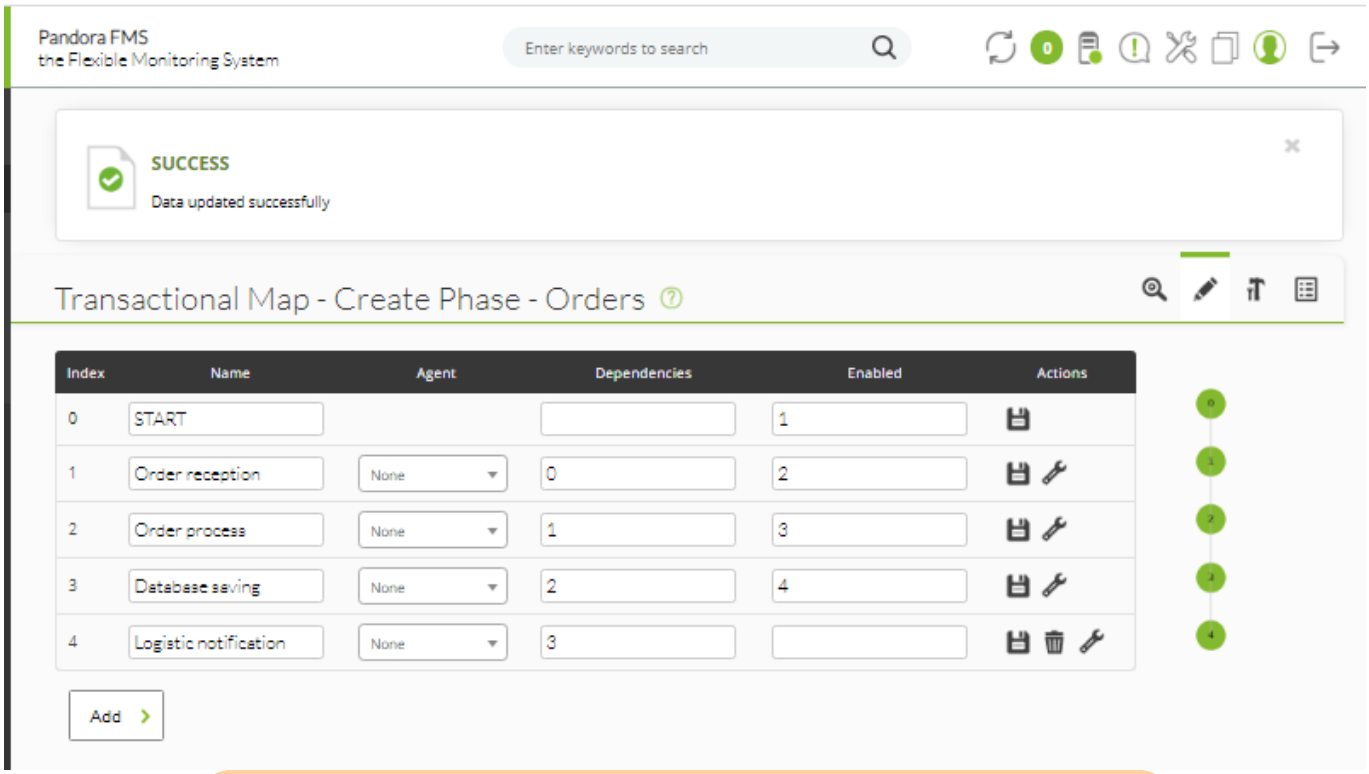


- **Launch script**: location or complete path of the control script, call, command, etc. The _name_ macro can be used to point out the name of the current transaction as an argument to the script.
- **Reattempts**: maximum number of execution retries until a positive response is obtained.
- **Maximum waiting time**: maximum value, in seconds, that the control script will remain running (feature not available for Windows transactional agent).

In this example, a custom script (`echo1.sh`) will be used to simulate the actions of the transaction trigger and control scripts for each stage:



Once the environment has been correctly configured, the transaction can be initiated.

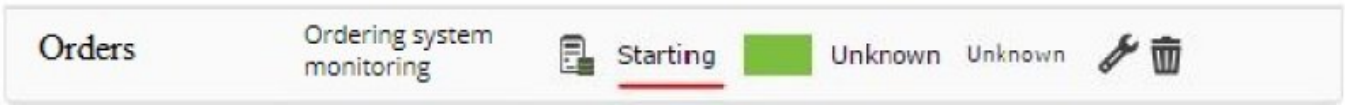


Any changes made to the transaction configuration will not be effective until the transaction is restarted.

Transaction control

Initiate a transaction

Go to the transactions list and click on the “initiate” icon (the black doughnut):



Once the transaction has been initiated the “stop transaction” icon will appear:

The status of the transaction can be displayed by clicking on the transaction's name:

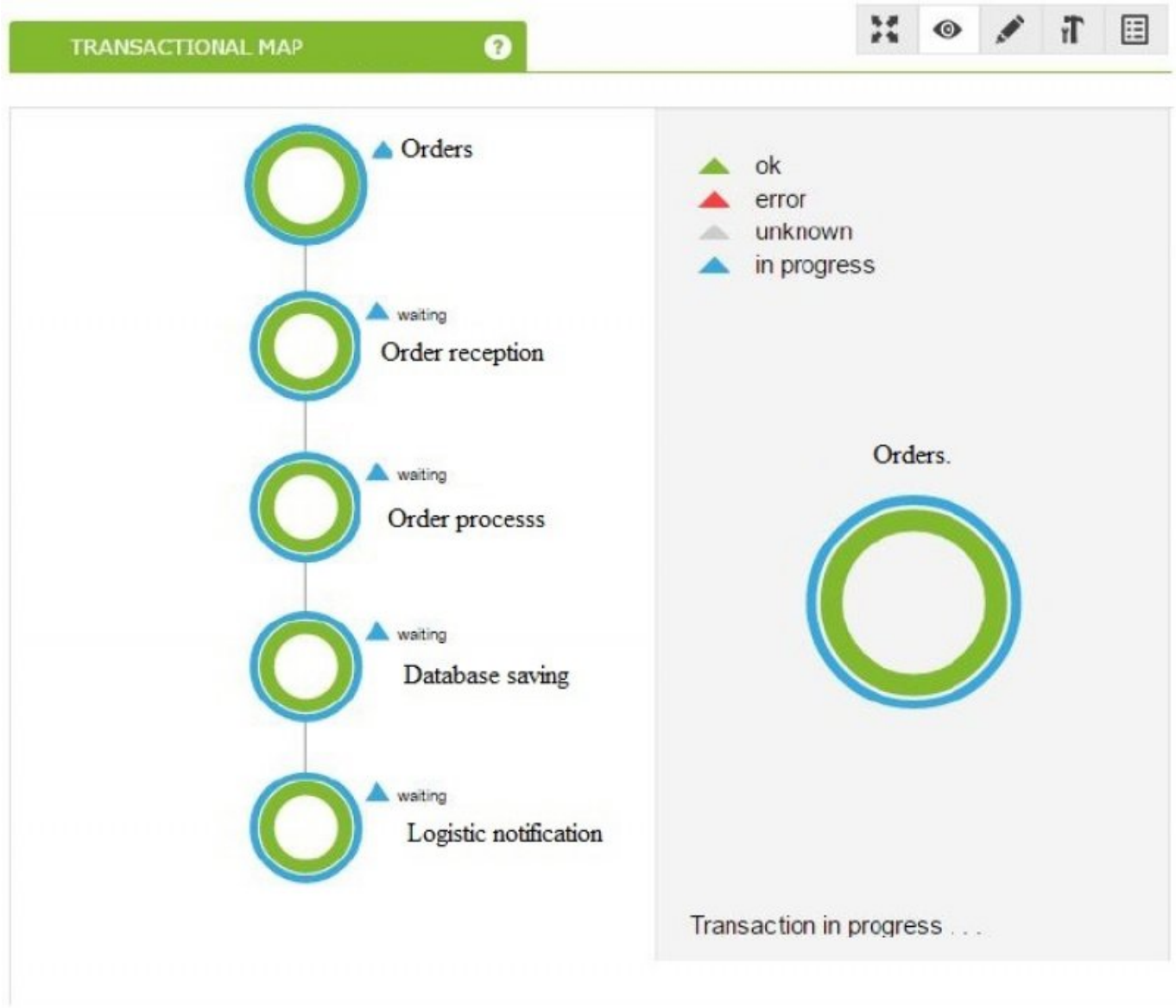
Stopping a transaction

To interrupt a transaction, click on the corresponding button, and, after a few seconds, click the “launch” icon to restart it.

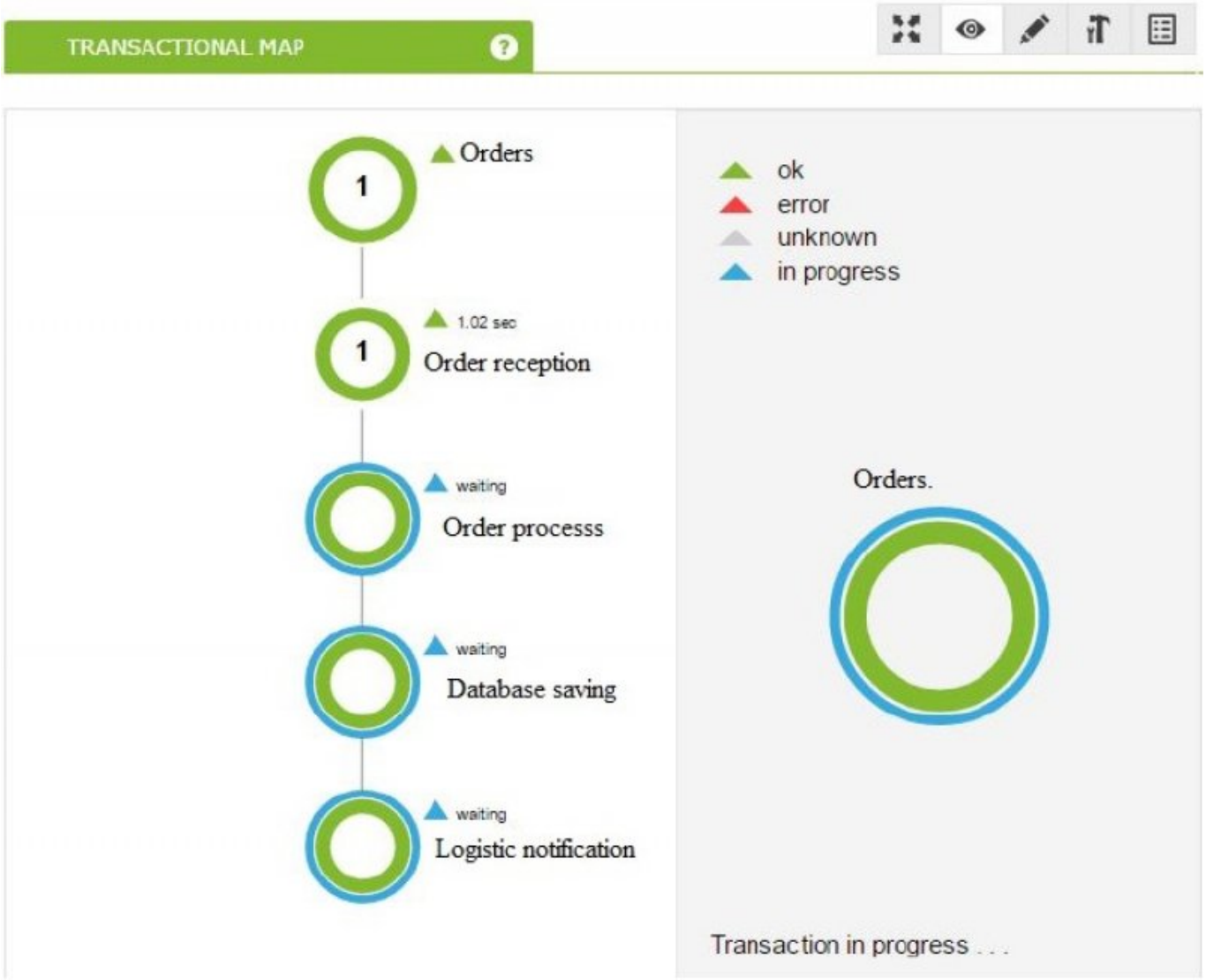
Viewing a transaction

Go to the view by clicking on the name of the transaction on the transaction list.

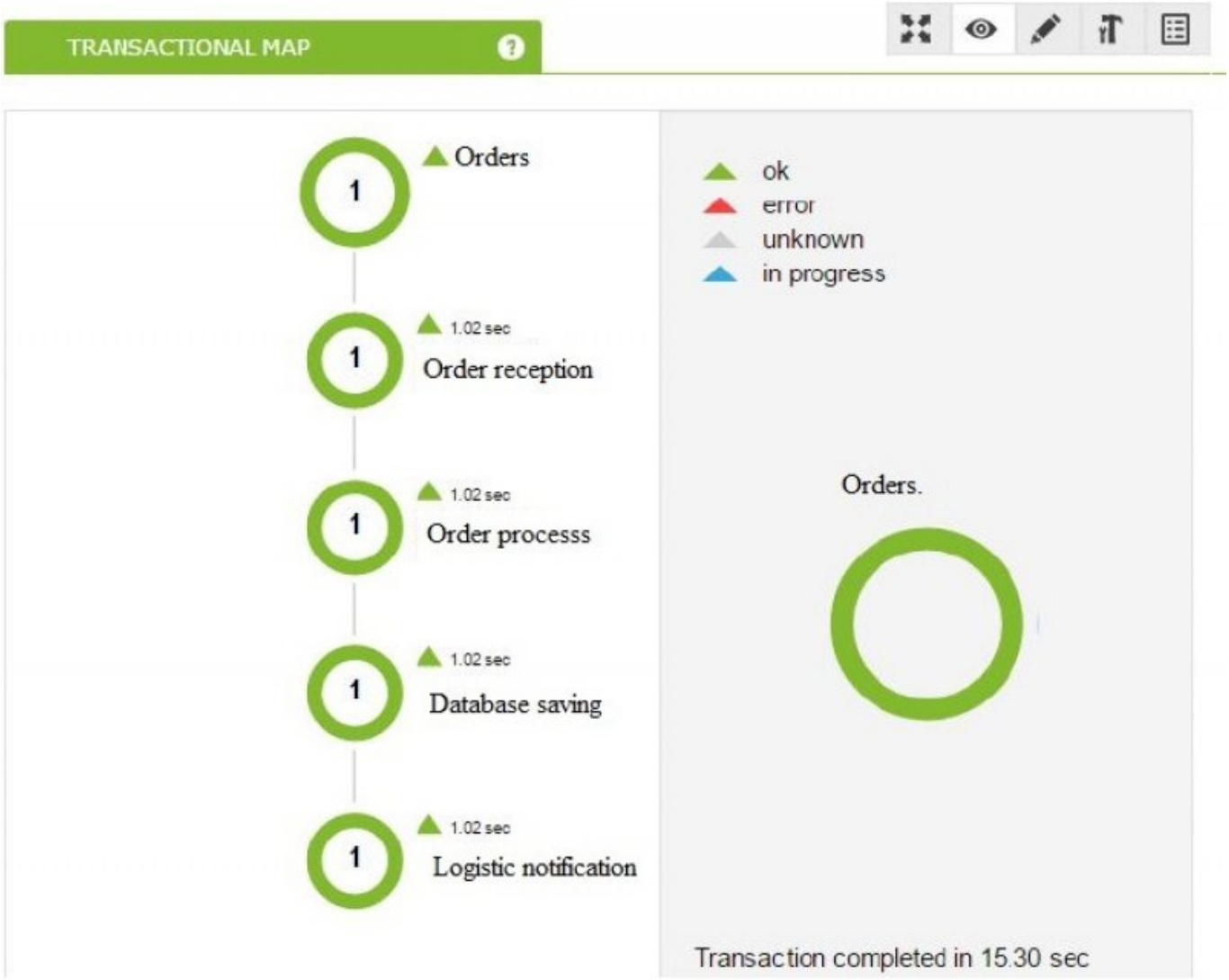
View of an execution in progress (initial phase of transaction):



View of an execution in progress (intermediate phase):



View of a completed transaction:



Configuration

Transaction server

The transaction server's configuration parameters are as follows:

```
transactionalserver 1

transactional_threads 1

transactional_threshold 1

# Work directory
# By default in icomingdir/trans
transactional_pool trans
```

- *transactionalserver*: starts the transaction server component upon starting the `pandora_server`.
- *transactional_threads*: added to the field by agreement. Only one internal thread is necessary to

manage active transactions from the server. The agent has a subsystem of dynamic threads to manage configured transaction executions.
- *transactional_threshold*: waiting time between status updates. This field specifies the time, in seconds, that the system will wait between changes in status of the elements which make up a transaction (recommended threshold: 4 seconds).
- *transactional_pool*: directory where `.lock` files are saved. The system uses these files to communicate among the different logical components (default location: `/var/spool/pandora/data_in/trans`).

For the system to be 100% operational, both the data server (`dataserver`) and the transaction server (`transactionalserver`) must be active. Go to tactical server view to check it:

Transaction agents

Transactional agents have a configuration file in their work directory, by default at:

```
/etc/pandora/pandora_transactional.conf
```

With the following content:

```
##################################################################
#  Base config file for Pandora FMS transactional agent
#  Version 2.0
#  Copyright (c) 2003-2016 Artica Soluciones Tecnologicas
#  http://www.pandorafms.com
##################################################################

server_ip = localhost
server_port =41121

# Temporal directory
temporal =/tmp
#temporal = C:\Program Files\pandora_transactional\tmp

# Log directory
log =/var/log/pandora/pandora_transactional.log

# Tentacle binary
tentacle_bin =/usr/bin/tentacle_client
```

```
 ##############################
 # Set the name of the agent
 ##############################
 #agent_name = transactional_agent
 agent_interval =300

 # Performance (in seconds) internal clock
 pause =5

 # Show all log messages (0 - show none, 1 - show script execution messages,
2 - show all)
 verbose =2
```

- *server_ip*: IP address or the name of Pandora FMS's transactional server.
- *server_port*: port where the tentacle server listens in.
- *temporal*: auxiliary directory for temporary work files.
- *log*: log file route.
- *tentacle_bin*: tentacle client's binary route.
- *agent_name*: optional, use a custom name to ID this agent. The name of the machine is the default ID.
- *pause*: default time, in seconds, between stages. It adjusts to the configuration received by the server to maintain synchrony.
- *verbose*: it controls the amount of information displayed on the log files. Any output in *STDERR* will overflow to this file by default:
  - 0: only show critical error messages.
  - 1: show control script executions.
  - 2: show all messages.

All transaction configurations will be published in the Pandora FMS server, and the agent is in charge of scanning these files and starting the necessary data structures to start the requested transactions.

In the face of any updates during an active transaction, the agent will interrupt the transaction, and start a new process to instantiate a new one. All previous progress will be lost.

The transaction system executes the complete transaction, which is to say, that if there is an error executing a script, the transaction will continue to be executed, **indicating the stage during which the error took place**. A transaction is considered failed when all stages are indicated as errors.

Download the transactional agent component from the plugin library
https://pandorafms.com/library/transactional-agent-2/

Go back to Pandora FMS documentation index