Task behavior

Introduction

This page describes the status a task can have and what the behavior is per status runtime.

Possible task statuses are:

OPEN	The task has the status `open` and is available to be performed.
STARTED	The task has the status `started` and is being performed.
COMPLETED	The task has the status `completed` and is finished.
CANCELED	The task has the status `canceled` and is not relevant any more for the process.
EXPIRED	The task has the status `expired` and is not performed in time and not relevant any more for the process.

On this page:

Definitions

- Precondition - Business rules that must be evaluated and can result in TRUE or FALSE.
- Applicable - All preconditions are met, the status is set to open and the task is available to be performed.
- Automated - Can only be performed by a system.
- Manually - Can only be performed through human interaction.
- ProcessDAO - Blueriq process database which contains the current process state of an application.
- TASKS table - A table in the process database with the current state of all tasks for all process instances.
- `application.properties` - A text file with server properties used by a Blueriq application.
- Runtime - An application that is executing Blueriq models.
- Process - A process instance on the runtime.
- Case stage - A stage in the reference process for a business function. Typically a process model of type phase which has a milestone as precondition and its own milestone as postcondition.

Process init

When a process is triggered by a message event, a process instance is made. The init state of the process instance is decided by 2 components:

1. The line that flows out of the message event. The first node the process flows to will get the status `open`**.** If this is a task, then this task will be required and `open` and gets an entry in the TASKS table.
2. The data in the message event will be used to decide what ad hoc nodes are applicable. All applicable ad hoc nodes will get an entry in the TASKS table.

Open

A task gets the status `open` when the task is applicable within a process. This means that the task has met its precondition. A precondition can be a business rule or a trigger. A business rule is logic that is evaluated by Blueriq and can be mapped to the process engine to give the precondition a value. A trigger can be another task or an event like a timer or a message.

When a task is applicable, then it will get an entry in the TASKS table of in the processDAO. The start date is the date the task is applicable, and the initial status will be set to `open`**.** Such `open` tasks will be shown in a worklist and can be executed if the user has permission to start the task.
When an automated task is applicable then it will start automatically. So an automated task will not be shown in a work list and changes status from `open` automatically to `started`.
When an `open` task is required then it MUST be performed before exiting the process. When an `open` task is not required but is applicable than the task MAY be performed.
When an ad hoc task is not applicable anymore and is not required it will be `canceled` and is not available on the worklist.

At the moment when an ad hoc task is applicable and required then it has to be performed. Even when the precondition changes back to FALSE. The task stays on the work list because it is required and MUST be performed.

Current behavior - Blueriq 9.7

	Precondition	Required
MAY be performed	TRUE	FALSE
MUST be performed	TRUE	TRUE
Not relevant	FALSE	TRUE
Not relevant	FALSE	FALSE

MUST be performed	TRUE -> FALSE	TRUE

There is an improvement that will change this behavior, namely that the precondition overrules the required value. When the improvement is implemented the next table can be used as reference:

	Precondition	Required
MAY be performed	TRUE	FALSE
MUST be performed	TRUE	TRUE
Not relevant	FALSE	TRUE
Not relevant	FALSE	FALSE
Not relevant	TRUE -> FALSE	TRUE

Be careful when modeling a dynamic process with required ad hoc tasks in combination with repeatable is set to `TRUE`. The process will only continue when the precondition of the task is evaluated to `FALSE` after completing the task. If the precondition stays `TRUE` then the repeatable setting will kick in and in combination with the required setting the task `MUST` be performed again. Required can be conditional, so there are multiple scenarios how to handle this.

Started

A task gets the status **`started`** when someone or a system starts to execute the task. A precondition to start a task is that the task must be **`open`** and thus applicable. The task will keep the status **`started`** during the entire time it is performed.

Manual tasks
When a task is started manually and cannot be finished due to an error it will do a roll back. This means that there is no transaction to the database and the task status will be set back to **`open`.** So a manual task that receives an error can be executed again via the worklist.
When a manual task is started it is also locked for other users. When **`started`** the task is not available in the worklist.

Automated tasks
When an automated started task gets an error it will stay in the status **`started`**. There will not be a transaction to the database, but the task cannot be triggered again without intervention. The reason an automated task does not get the status **`open`** is to prevent an infinite loop. First the cause of the error needs to be fixed before the task can be set back to **`open`.** Setting an automated task back to **`open`** can be done by the run time API. Once the automated task is set from **`started`** to **`open`** it will trigger itself on the next tick of the portal timer. The portal timer is a server setting in a server property.

Cancelling tasks
It is possible to cancel a **`started`** task, which will set a task back to **`open`.**

When the implementation has a cancel event and this event is used for example by pressing a cancel button the task does a roll back and gets the status **`open`.** The task is available on the worklist.

Sometimes it is needed to intervene and cancel a task. For example when someone closes his browser. Opening the browser again will open the dashboard, but will not show the task in the worklist, because the task still has the status **`started`**. There are 3 ways to cancel **`started`** tasks and set it back to **`open`.**

- Use the server setting `blueriq.processengine.cancel-started-tasks=true`. This will cancel all tasks when...:
    - ... the Runtime starts and an application is used for the first time
    - ... in development, the reload project/projects/branch/settings buttons are clicked
    - ... a new version of the application is published using the publisher
- Reloading the project in the developer dashboard.
- Use the run time API to change a task status from **`started`** to **`open`**. Information about the run time API can be found here.

An automated task will trigger itself when the status is set back to **`open`.**
A manual task will be available again in the worklist.

Completed

A task gets the status **`completed`** when a task is successfully performed. The task is not part of work lists any more. If the task is repeatable, then after completing the task a new entry will be created in the `TASKS` table with the status **`open`** and thus part of the worklist. The finished task has the status **`completed`**.

A **`completed`** task cannot be performed again if it is not repeatable. So make sure that all information is available after finishing the task.

Canceled

A task gets the status `canceled` when it is not relevant any more. For example, there exists a non-required ad-hoc task with status `open`**.** There is enough data within the case to finish the sub-process, then all `open` tasks in the sub-process are canceled because they are not relevant any longer. A required task cannot be canceled and should always be performed unless it is not applicable any more (not implemented yet!).

Expired

A task gets the status `expired` when its timer exit evaluates TRUE.

There is a property that sets the Timer interval on a server. This property can be set in the Settings/Timer on the developer dashboard or in the `application.properties` file. At every interval all relevant timer nodes will be evaluated. When a timer exit is evaluated TRUE then the timer exit event will be flowed (the process engine will follow the line from the timer exit event). The task gets the status `expired` and is not available on the work list and thus cannot be performed any more.

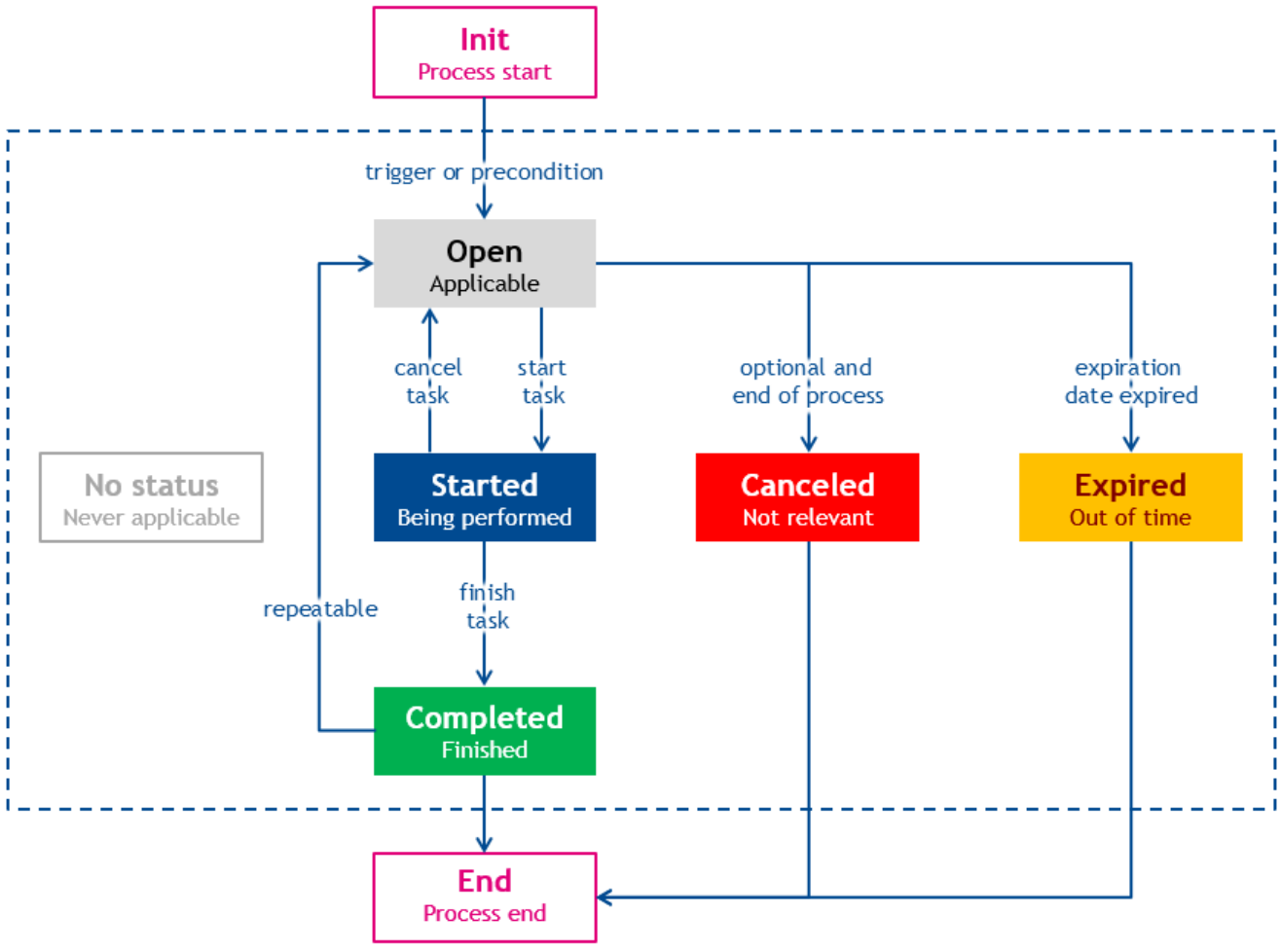
Be careful with timeout dates in dynamic processes, because when a task timed out the taken action cannot be undone. You must take into consideration the level of empowerment you give the system in telling a user he or she cannot perform certain actions anymore. It is better to use due dates on events and let an event notify a user, or add new information to the case so that new decisions can be made which can steer the process to a different direction if needed.

There are patterns for time management in a dynamic process which do not use the expire date.

Process Ends

All not required tasks will get the status `canceled`. Required tasks must be performed before the process can end.

Schematic overview



Fail-safe scenarios

This page has a few scenarios that describe behavior of the process engine in dynamic process handling with failures and how to fix the failures.