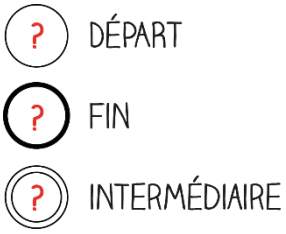
C1.5 – Les évènements

A présent, nous allons voir les évènements plus en détail. Dans un modèle de processus, il existe trois catégories d'évènement :

- les évènements de départ,
- les évènements de fin,
- les évènements intermédiaires.

Les évènements peuvent également être de différentes natures, comme par exemple la réception d'un message.

Un des principes de base de BPMN est de pouvoir ajouter des pictogrammes aux différents concepts, afin de spécifier leur nature. Sur un évènement, ces icones s'inscrivent à l'intérieur du cercle. Notons qu'il n'est pas obligatoire de préciser la nature de l'évènement et que le cercle peut rester vide.



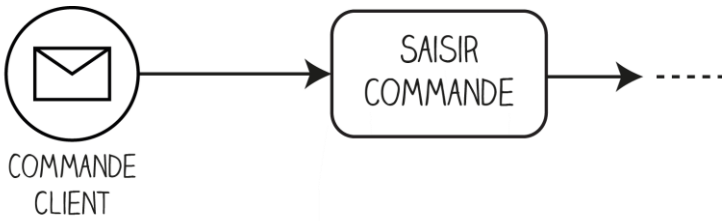
Nous allons ici définir quelques-uns de ces types d'évènements, parmi les plus utilisés.

1. EVENEMENTS DE DEPART

Ces évènements déclenchent le processus. Ce sont des évènements de type « ***catch*** » c'est-à-dire des évènements dont notre processus est le destinataire ou l'attrapeur.

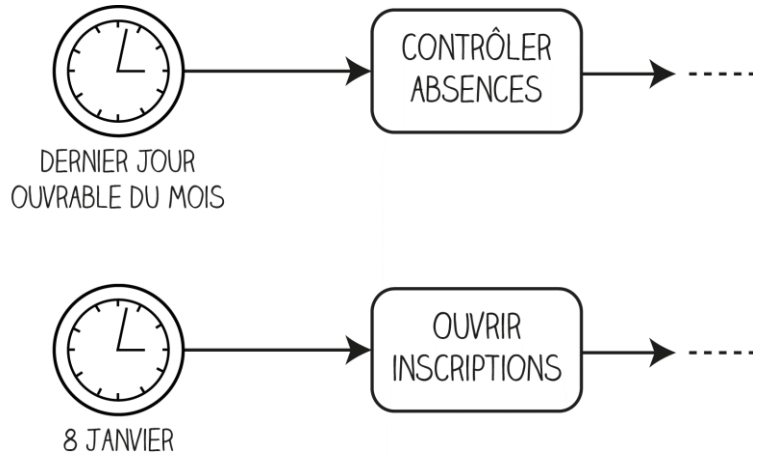
La star des évènements, c'est le **message**, représenté par une enveloppe. Le processus démarre suite à la réception d'un message provenant de l'extérieur du processus, peu importe sa forme (oral, courrier, email, etc…).

Par exemple, la saisie de la commande démarre lorsque le client donne sa commande.



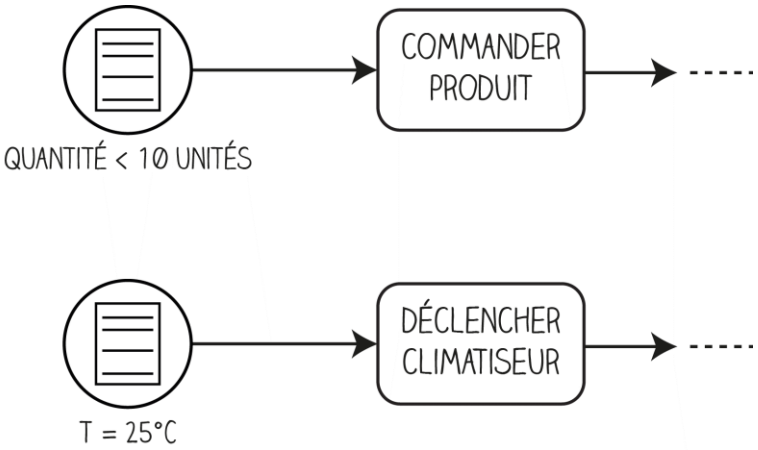
Autre tête d'affiche, le **timer**, dont le pictogramme est une horloge. Cet évènement correspond à une indication temporelle comme une date, une heure ou une périodicité. Le processus démarre lorsque la condition temporelle est vérifiée.

Par exemple, la réalisation du processus de paye démarre le dernier jour ouvrable du mois ou encore l'ouverture des inscriptions démarre le 8 janvier.



La condition peut également porter sur une donnée. Le type d'évènement conditionnel permet de formaliser le déclenchement d'un processus suivant une règle de gestion.

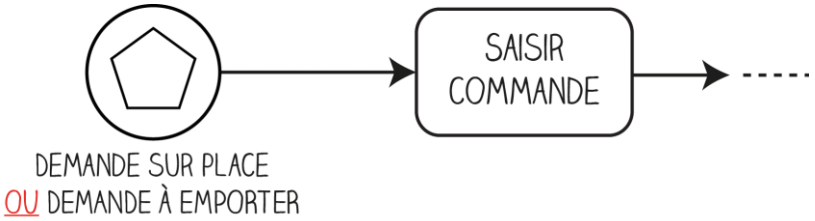
Par exemple, le processus de réapprovisionnement démarre lorsque la quantité du produit est inférieure à 10 unités ou encore lorsque la température atteint 25° cela déclenche le processus de régulation de la température.



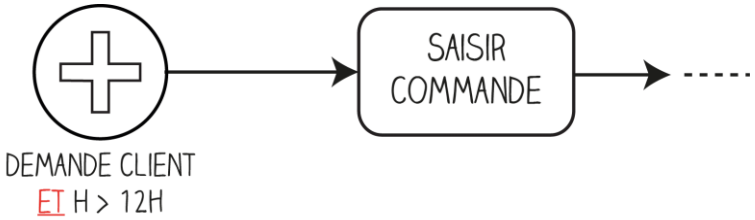
Lorsque plusieurs évènements déclenchent le processus, on utilise alors le type **multiple**. On spécifie dans le libellé les différents évènements. Deux formalismes existent correspondant à deux modes de déclenchement différents.

Lorsque seule l'occurrence d'un des évènements démarre le processus, on utilise un pentagone.
Lorsque l'occurrence de tous les évènements sont nécessaires au déclenchement du processus, on appelle cela un multiple-parallèle et le signe utilisé est un +.

Par exemple, la saisie de la commande peut démarrer pour une demande de plat sur place ou une demande de plat à emporter.

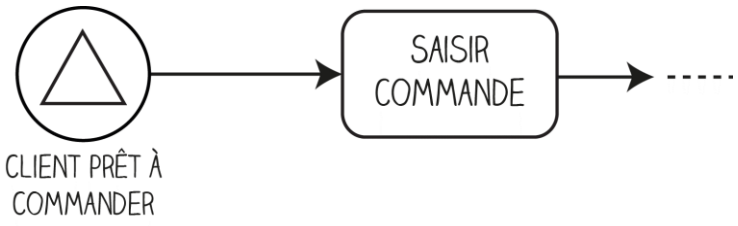


Autre contexte, la saisie de la commande pourrait démarrer par la demande du client mais également s'il est midi passé. Il faut que tous ces évènements soient réunis pour passer commande.



Enfin, le **signal**, représenté par un triangle. Le processus se déclenche après avoir capté un signal d'un autre processus.

Par exemple, notre processus de service client dans un restaurant pourrait commencer par le signal du client qui ferme sa carte, lève la main ou scrute la salle à la recherche d'un serveur disponible.
Ce n'est pas un message destiné à un serveur en particulier mais un signal qui sera capté par un des serveurs et déclenchera une instanciation du processus. Nous reviendrons plus en détail sur ce dernier concept avec les évènements intermédiaires.



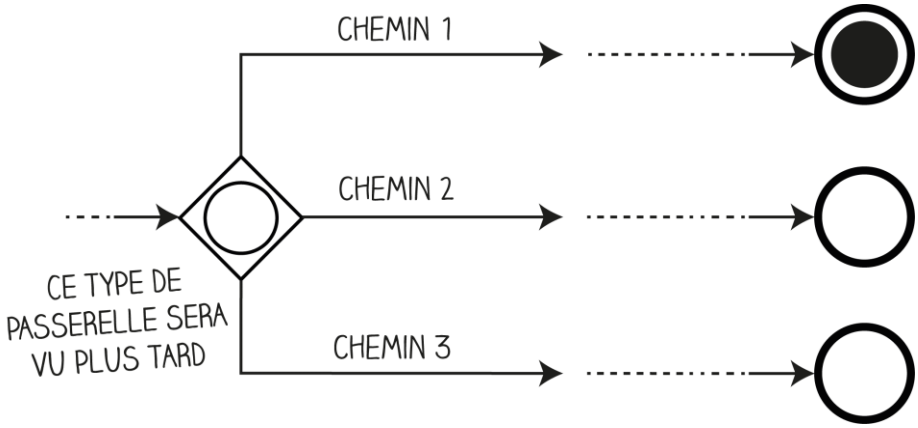
2. EVENEMENTS DE FIN

Ce sont des évènements de type « throw » c'est-à-dire dont notre processus est l'émetteur ou le déclencheur.

Nous retrouvons donc le **message** et le **signal** en mode throw, c'est-à-dire destiné à un autre processus. Il existe également l'évènement **multiple** mais peu utilisé.

Nous présentons en plus l'évènement de terminaison ou **terminate**, formalisé avec un remplissage du cercle. Rappelez-vous, grâce aux passerelles, il est possible d'avoir plusieurs chemins parallèles d'exécution dans un même processus et donc potentiellement plusieurs évènements de fin.

Cependant, si un chemin se termine par un évènement de type terminate, cela met fin à l'ensemble du processus, stoppant les flux parallèles qui pourrait être en cours.

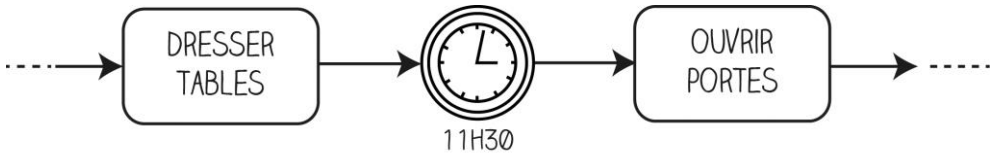


3. EVENEMENTS INTERMEDIAIRE

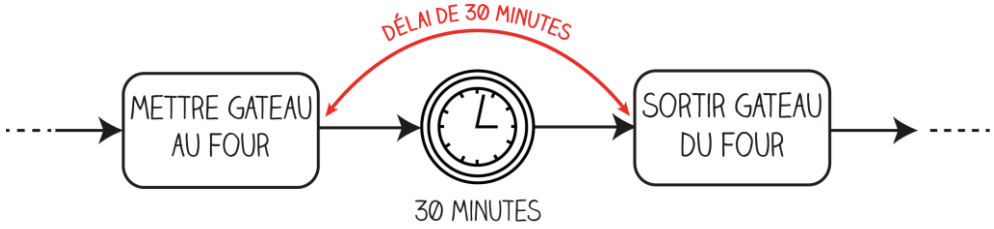
Les évènements intermédiaires se produisent eux, au cours d'un processus. Ils peuvent être de type « catch » ou « throw ». Le formalisme des pictogrammes va donc devoir spécifier pour certaines natures d'évènement si notre processus est le déclencheur ou le destinataire.

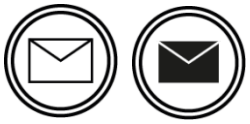
Nous retrouvons tout d'abord le **timer**, l'un des plus utilisés qui permet de modéliser un délai d'attente ou une échéance entre deux activités.

Par exemple, dans le processus de préparation de la salle de restauration, il faut attendre 11H30 avant d'ouvrir les portes du restaurant, peu importe si le dressage des tables a été fini avant.



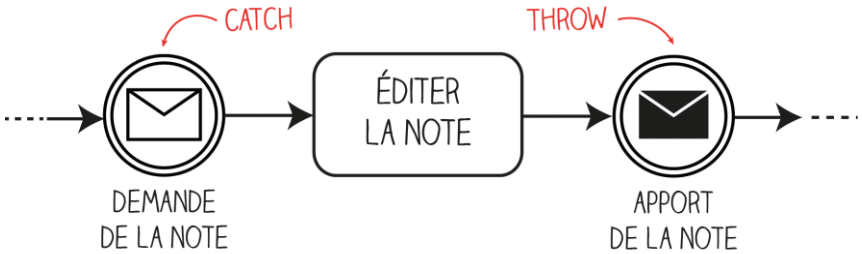
Ou encore, il faut attendre 30 minute entre la mise au four et l'enlèvement du gâteau et encore 2H avant de le déguster. Notez que le délai se calcule à partir de la fin de l'activité jusqu'au début de la suivante.





Pour les évènements intermédiaires de types **message**, on distingue l'envoi ou la réception par le remplissage de l'enveloppe.

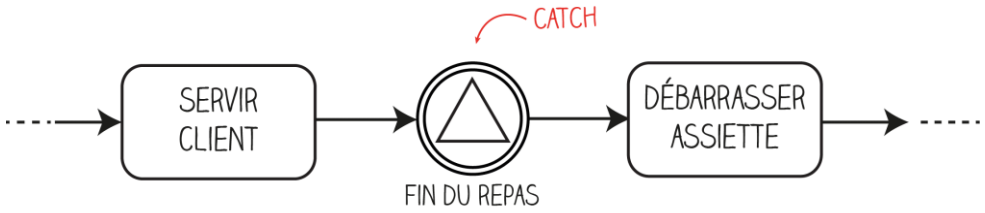
Exemple : le client peut demander la note et notre serveur la lui remettre une fois éditée. Attention, il faut toujours se placer au niveau du processus étudié. La demande de note est effectivement envoyée par le client. Mais le processus représenté ici, est celui du serveur, qui lui reçoit le message. C'est donc un type catch.

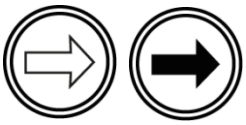


Il en est de même pour les évènements intermédiaires de type **signal**. Nous précisons que pour qu'un signal émis face référence à un signal reçu, ils doivent être libellés de la même façon. On les utilise pour deux raisons : pour broadcaster un message à plusieurs destinataires, définis ou

non mais surtout pour s'affranchir de la contrainte qu'un message ne peux pas être envoyé dans un même processus. En effet, la norme BPMN n'autorise pas l'envoi-réception de message au sein d'un même processus.

Dans notre exemple, une fois son plat terminé, le client peut émettre un signal au serveur qui viendra le débarrasser.





L'évènement de type **Lien** est plus un élément d'aide graphique qu'un véritable évènement. En effet, la modélisation complète de certain processus peut vite devenir imposante et donc perdre en lisibilité. Le lien permet de faire un renvoi entre deux parties de processus. La séquence de flux est découpée pour une meilleure visibilité. Les évènements lien marchent par paires et il est obligatoire de donner le même nom aux deux liens se faisant référence.

Par exemple, supposons que notre processus devienne vraiment trop grand et doivent être découpé. Je termine la première partie par un évènement lien de type throw et je récupère le flux par un évènement lien de type catch. C'est juste une question de représentation graphique. Cela n'a aucune incidence sur le processus.
Nous venons de voir les concepts de base afin de réaliser nos premiers modèles de processus.