# CMMN V2.0

Document : <https://www.omg.org/spec/CMMN/1.1/PDF>

Un « Case » est un procédé qui implique des actions prises sur un sujet dans une situation particulière pour achever un résultat.

Par exemple :

* Un case peut : application de la loi sur un sujet dans une situation particulière.
* Un case peut : soins donnés à un patient sachant son historique médical et ses problèmes médicaux actuels.

Utiliser des données afin de prendre les actions nécessaires pour résoudre le Case.

## CasePlanModel

Chaque « Case » doit être associé avec exactement un « casePlanModel ».

## CaseFileItem

Représente n’importe quel type d’information, qu’elle soit structurée ou non structurée ou sous une forme simple ou complexe.

Peut être n’importe quoi, d’un document ou un répertoire stocké dans un CMIS, une hiérarchie de « CaseFileItems » ou un document XML.

La structure ainsi que le langage utilisée pour définir la structure est définie par « CaseFileItemDefinition ».

Il faut comprendre comment on peut utiliser le protocole CMIS avec CaseFileItem.

## SEntry

Capture les événements devant des PlanItem / CaseItem et évalue une condition.

Possède deux paramètres :

* **OnPart [0…\*]** : Spécifie les événements qui servent de déclencheur. Toutes les propriétés « OnPart » doivent être satisfaites. Il existe deux types de « OnPart « :
  + OnPartPlanItem : événement venant d’un PlanItem
  + OnPartCaseItem : événement devant d’un CaseItem.
* **IfPart [0…1]** : Si la condition est présente, elle doit alors être vérifiée.

## PlanItem

Les « PlanItems » représentent des « Unit Of Work » qui font partie d’un « case ».

Voici les attributs d’un PlanItem

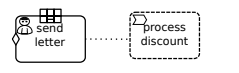
|  |  |
| --- | --- |
| Attribut | Description |
| Nom | Nom de l’objet « PlanItem » |
| itemControl [0…1] | **ManualActivationRule** : L’activation automatique est le comportement par défaut où il n’est pas nécessaire d’activer manuellement la tâche.  Avec l’activation manuelle, l’utilisateur peut décider d’activer ou désactiver la tâche.  Il est possible aussi de demander une activation manuelle d’un utilisateur, lorsqu’une certaine condition est satisfaite par exemple : ${var > 100}.  Lorsque les critères d’entrées sont respectés, l’état va de « AVAILABLE » à « ENABLED » puis de « ENABLED » à « ACTIVE » lorsque la tâche est activée.  Au contraire avec une activation automatique, l’état va directement de « AVAILABLE » à « ACTIVE ».  Voici les propriétés d’un ManualActivationRule :   * Nom * ContextRef * Condition : une expréssion doit être évaluée comme un « boolean ». Voici les attributs d’une expression :   + Language   + Body |
| PlanItemDefinition [1] | Se réfère au « plan item definition » par exemple :   * Task * PlanFragment * EventListener * Milestone |
| entryCriteria [0…\*] | Représente la condition pour laquelle le PlanItem devient disponible.  Lorsqu’un « entryCriteria » est satisfait alors l’état va vers « Enabled ». |
| exitCriteria [0…\*] | Représente la condition pour laquelle le PlanItem est terminée.  Lorsqu’un « exitCriteria » est satisfait alors l’état change vers « TERMINATED ». |

## HumanTask

**Définition** : Tâche qui est exécutée par un « case worker ». Lorsque la tâche « HumanTask » n’est pas bloquante, elle peut être considérée comme une tâche manuelle.

**PlanningTable** : Elle peut avoir un PlanningTable. Les utilisateurs peuvent planifier des tâches durant le « run-time », ils peuvent en effet sélectionner une ou plusieurs « Discretionary  items ».

**Par exemple** : dans un processus de plaintes, il peut exister des situations où le « case worker » qui envoie une lettre au consommateur peut vouloir offrir un discount. On ne souhaite pas forcément faire un discount à chaque fois, mais il est utile d’offrir la possibilité au « case worker » de faire un « discount ».



Dans l’exemple ci-dessus, le « case worker » qui exécute la tâche « send letter » peut ajouter un « process discount discretionary task » au plan.

**Propriétés d’un « HumanTask »**:

|  |  |
| --- | --- |
| Propriété | Description |
| PlanningTable | Un « PlanningTable » associé à un HumanTask. |
| PerformerRef | Le « performer » du « HumanTask » |
| formRef | Référence vers un formulaire |

## ProcessTask

Peut être utilisé dans un « Case » afin d’invoquer un processus business.

Les paramètres sont utilisés afin de passer des informations entre le « ProcessTask » et le processus auquel il se réfère :

* « Inputs » du « ProcessTask » sont « map » aux « inputs » du processus.
* « Outputs » du « ProcessTask » sont « map » aux « outputs » du processus.

**Paramètres « IsBlocking »**: Lorsque le « ProcessTask » est bloquant « IsBlocking est vraie », alors le « ProcessTask » attend que le processus associé au « ProcessTask » soit fini.

Si « IsBlocking est faux » alors le « ProcessTask » n’attend pas la fin du processus.

**Sélection du processus** : La sélection du processus pour un « ProcessTask » peut se faire en sélectionnant a QName valide d’un processus existant lors de la phase de conception ou en spécifiant une expression qui sera évaluée en un « QName » valide. La dernier permet de sélectionner un processus dynamiquement durant le “runtime”.

Voici les paramètres d’un processus task

|  |  |
| --- | --- |
| Paramètre | Description |
| processRef : Process[1] | Référence à un processus.  Si “ProcessRef” n’est pas spécifié alors “ProcessRefExpression” doit l’être.  Seulement un attribut “ProcessRef” ou “ProcessRefExpression” doit être spécifié.  Voici les attributs d’un processus :   * **implementationType** : Type d’implémentation du processus business. * **Inputs** : Un ou plusieurs “inputs”. * **Ouputs** : Un ou plusieurs “outputs”. * **Name** : Nom du processus. * **ProcessRef : QName** : Le processus concret devant être utilisé. |
| Mappings: ParameterMappings[0…\*] | Un ou plusieurs objets “ParameterMapping”. Un “ParameterMapping” d’un “ProcessTask” spécifie comment un input d’un “ProcessTask” est “map” à un input d’un processus et comment un output d’un “ProcessTask” d’un processus est “map” à un “output” d’un “ProcessTask”. |
| processRefExpression : Expression[0…1] | L’expression évalue un “QName” qui est un “QName” d’un processus existant.  Permet de déterminer le processus à invoquer durant le runtime. |

## TimerEventListener

**Définition** : « catch predefined elapses of time ».

**Paramètres** :

|  |  |
| --- | --- |
| Paramètre | Description |
| timerExpression | Une expression optionnelle qui est utilisée afin d’évaluer une date, durée ou intervalle. |
| timerStart | « Starting trigger » |

D’après ISO 8601 il existe plusieurs formes de timer :

* **Timer duration** : Spécifie le temps que la tâche doit attendre avant d’être exécutée.
* **Time cycle** : Permet d’exécuter la tâche de façon périodique.
* **Time date** : Spécifie la date à laquelle la tâche sera déclenchée.

## PlanItemDefinition

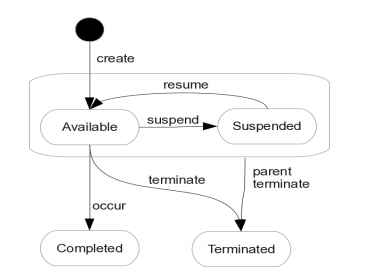
Permet de réutiliser une configuration par exemple si deux « planItems » référencent le même « humanTask » alors les deux vont référencer le même « humanTask ».

PlanItemDefinitions peuvent aussi être instanciés durant l’exécution, ce processus se réfère au concept de « planning ».Le planning permet à des utilisateurs de créer des « plan-items » quand il est nécessaire.

## Milestone

Exprime le fait qu’un objectif intermédiaire a été atteint.

Les conditions qui définissent si un « Milestone » est atteint sont exprimées par des «SEntries ».



Voici les différents états d’un « Milestone » :

|  |  |
| --- | --- |
| Etat | Description |
| Available | Une instance de Milestone qui se trouve dans cet état attend que le SEntry soit satisfait. |
| Suspended | Cet état permet à un « case worker » (human) ou à une instance de « enclosing stage » de suspendre un « milestone » qui n’a pas été atteint. |
| Completed | Indique qu’un critère du « Milestone » est vraie. |
| Terminated | Cet état indique une terminaison par un « case worker » ou une instance de « enclosing stage ».  Indique qu’un « case worker » n’est plus intéressé par le « Milestone ». |

## Rôle

Un « CaseRole » autorise des « case workers » ou une équipe de « case workers » à exécuter un ou plusieurs « HumanTask ».

Par exemple

* Le rôle docteur peut contenir un ou plusieurs participants qui sont autorisés à exécuter des « HumanTasks ».

TODO

* Il faut vérifier que seulement un utilisateur qui a le bon rôle a le droit de confirmer le formulaire.
* Il faut une vue qui permet d’afficher les formulaires.

# BPMN V2.0

Document : <https://www.omg.org/spec/BPMN/2.0/PDF>

## Task

### Définition

Une tâche est une activité atomique qui se trouve dans un processus. De façon générale, elle est exécutée par un utilisateur ou par une application.

### Types de « Task »

#### Service task

Tâche qui utilise un service web, une application ou un autre moyen pour compléter la tâche.

#### Human task

**A REMPLIR**

# Workflow

Problème avec les « locks » distribués. Les 5 événements sont reçus dans cette ordre pour l’aggregat 1 :

* 2, 1, 5, 3, 4

Il faut faire un lock distribué pour l’aggregat et sa version.

Il faut aussi vérifier que ce n’est pas une modification concurrente et ressayer plusieurs fois.

# TODO

1. Supporter SENTRY : OK
2. Supporter Task : OK
3. Supporter ProcessTask et tous les paramètres : OK

# Références

<https://documentation.edorasware.com/CLD/1.5.0.S101/documentation/cmmn.html#case-file-item>

<https://arxiv.org/pdf/1608.05011.pdf>