|  |
| --- |
| Macintosh HD:Users:jh7rnn:Dropbox:Citeck:EcoS:Дизайн 2015:Citeckecos.png  На платформе Alfresco ECM |
| **Citeck EcoS Case Management mapping to CMMN 1.1** |

|  |
| --- |
| Автор: Симонов Павел  [pavel.simonov@citeck.ru](mailto:pavel.simonov@citeck.ru) |
| 2016 |

**Содержание**

[Диаграммы сущностей 3](#_Toc451756775)

[Сiteck EcoS 3](#_Toc451756776)

[Case Management Model and Notation 4](#_Toc451756777)

[Доработки 5](#_Toc451756778)

[Рекомендации по реализации экспорта и импорта 6](#_Toc451756779)

[1. Conditions 7](#_Toc451756780)

[2. Event 9](#_Toc451756781)

[3. Activity 12](#_Toc451756782)

[4. Task 14](#_Toc451756783)

[5. Action 14](#_Toc451756784)

[6. Stage 15](#_Toc451756785)

[7. Role 15](#_Toc451756786)

[8. Case 16](#_Toc451756787)

[Полезные ссылки 17](#_Toc451756788)

****Диаграммы сущностей****

****Сiteck EcoS****

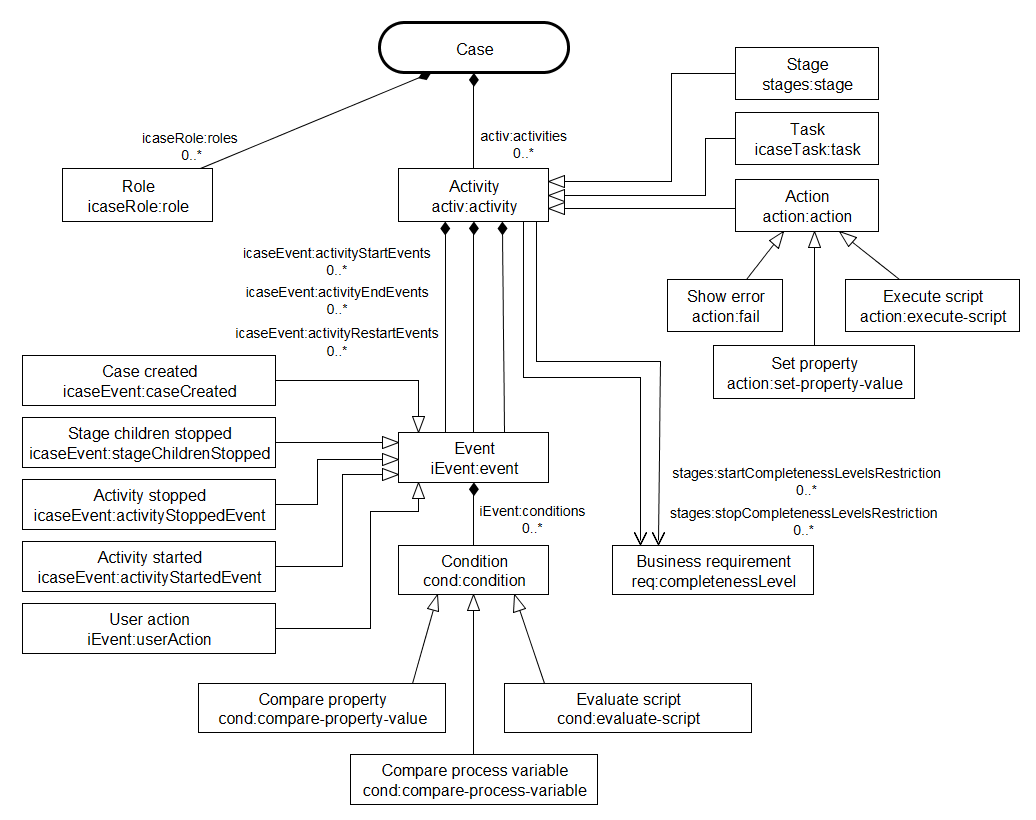


Рисунок 1 – Диаграмма сущностей кейс менеджмента в Citeck EcoS

|  |  |
| --- | --- |
| * Case * Role * Activity * Stage * Task * Action * Show error * Set property * Execute script * Business requirement | * Event * Case created * Stage children stopped * Activity stopped * Activity started * User action * Condition * Compare property * Evaluate script * Compare process variable |

Case Management Model and Notation

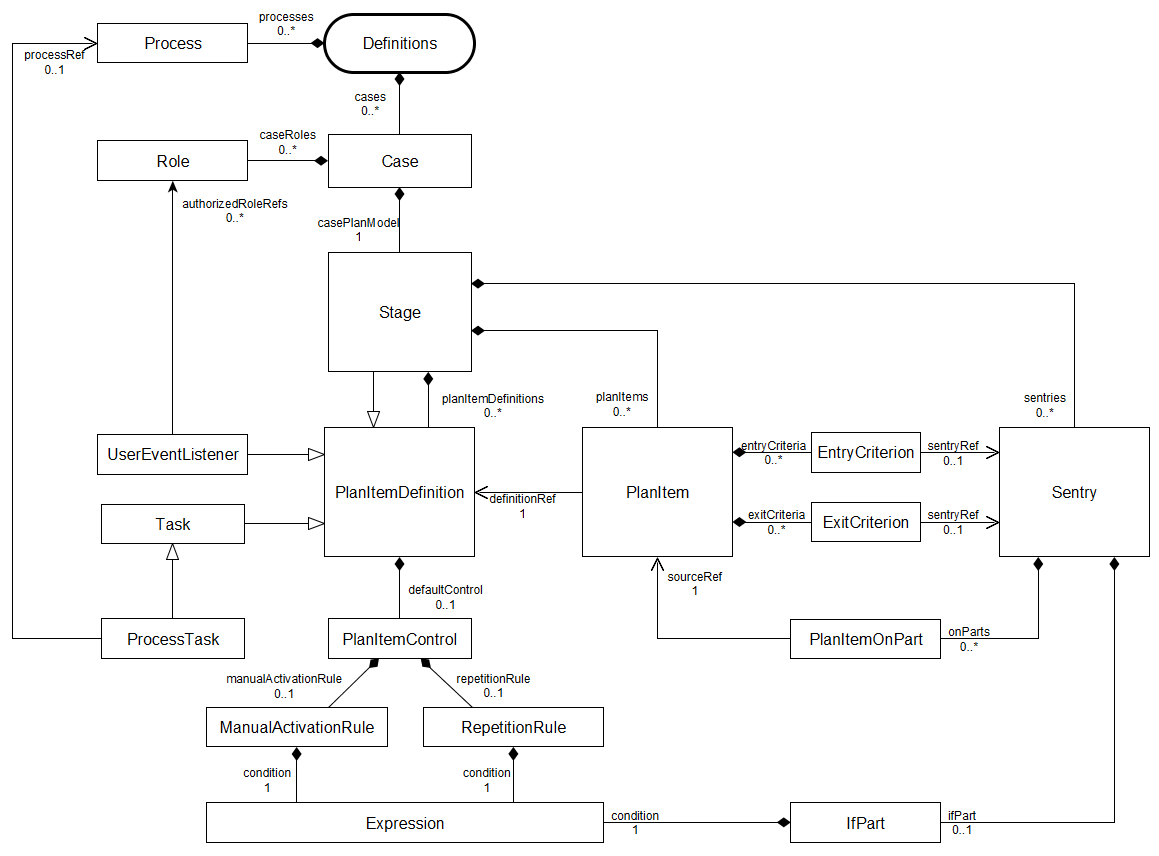


Рисунок 2 – Диаграмма сущностей кейс менеджмента в CMMN 1.1\*

|  |  |
| --- | --- |
| * Definitions * Case * Role * PlanItemDefinition * Stage * Task * ProcessTask * UserEventListener * PlanItem * Process | * Criterion * EntryCriterion * ExitCriterion * Sentry * PlanItemOnPart * IfPart * Expression * PlanItemControl * ManualActivationRule * RepetitionRule |

\* Изображены только сущности, функционал которых реализован в Citeck EcoS

Доработки

1. AuthorizedRoles

а) Необходимо добавить аспект icaseRole:hasAuthorizedRoles в модель ролей (icaseRoleModel.xml) c одной множественной ассоциацией icaseRole:authorizedRoles;

б) Добавить на форму создания iEvent:userAction поле icaseRole:authorizedRoles;

в) Дополнить CaseActionsProvider кодом для проверки наличия текущего пользователя в списке authorizedRoles (непосредственно в одной из ролей или внутри одной из групп выбранных в роли). Если пользователь есть – возвращаем действие, иначе – прячем.

1. Flag ‘repeatable’ in activity

а) Добавить boolean свойство *activ:repeatable* в модель активностей (icaseActivityModel.xml -> activ:activity).

б) Добавить новое свойство на формы создания активностей (stage, task, action).

в) Расширить метод “startActivity” в CaseActivityServiceImpl добавив туда логику для перезапуска активностей:

Когда был вызван метод startActivity: Если активность уже завершена, то проверяем флаг “repeatable” и если он true, то вызываем метод resetActivity и затем выполняем код из startActivity

г) Дописать инвариант на релевантность для icaseEvent:activityRestartEvents. Следует скрыть это поле если оно пустое и показывать только если там что-то есть.

д) Оставить комментарий над аспектом icaseEvent:hasActivityRestartEvents о том, что он устарел и вместо него теперь флаг repeatable.

е) Создать скрипт для переноса всех событий из ассоциации icaseEvent:activityRestartEvents в ассоциацию icaseEvent:activityStartEvents и установку флага repeatable если такие ассоциации были.

1. Stage stages:autoComplete

а) Добавить флаг stages:autoComplete в модель этапов (stagesModel.xml -> stages:stage).

б) Добавит этот флаг на формы создания/редактирования этапа.

в) Убрать форму создания icaseEvent:stageChildrenStopped

г) В CaseActivityEventTrigger метод tryToFireStageChildrenStoppedEvents:

- Поменять название метода на tryToCompleteStages

- Заменить eventService.fireEvent на caseActivityService.stopActivity

- Добавить проверку на наличие флага autoComplete (только если он true вызывать caseActivityService.stopActivity)

Рекомендации по реализации экспорта и импорта

1. Для экспорта информации о типе следует использовать свойство *dd:type*, где *dd* – это *http://www.alfresco.org/model/dictionary/1.0*.
2. Почти у всех сущностей в CMMN есть поле id. Если у сущности есть прямой аналог в Citeck EcoS, то в поле id следует поместить значение свойства sys:node-uuid. Если сущность в CMMN создается новая, то id нужно сгенерировать уникальным в пределах документа.
3. Там где требуется экспортировать «все свойства не сохраненные в других полях» следует поставить ограничение по уровню родства. То есть если у нас есть lact:decisionTransferCreditCommitteeTask, то свойства необходимо взять из этого типа, из типа icaseTask:simpleTask и icaseTask:task (свойства активности должны быть уже сохранены, а свойства описанные в cm:content, cm:cmobject и sys:base следует опустить). Для получения информации о свойствах и ассоциациях следует использовать DictionaryService.
4. При экспорте множественных ассоциаций следует объединять их через «,» (например: icaseTask:confirmers=”1d110a0e-6f01-4b4a-bdcc-ae98714f599e,fd80de5d-608d-4b05-8a22-b22ef8c9e67a,d4b28721-ba38-4e52-89e2-3d0aea01b50e”).
5. При реализации экспорта и импорта условий следует сначала реализовать только один случай когда у нас одно условие типа *cond:evaluate-script*. После этого перейти к реализации следующих сущностей и когда первый прототип будет готов можно вернуться и обработать второй случай с сложными условиями.
6. Conditions

**Type: *cond:condition***

Export

а) Если у нас одно условие, и оно имеет тип *cond:evaluate-script*, то берем содержимое свойства *cond:evaluate-script:script*, оборачиваем его в <![CDATA[${…}]]> и помещаем внутрь condition:



б) Если у нас несколько условий или одно, но его тип не *cond:evaluate-script*, то представляем условие в виде Alfresco Repository View (<https://wiki.alfresco.com/wiki/Export_and_Import_View_Schema>). Вначале мы описываем рутовый объект view:view и перечисляем нэймспейсы. После этого мы можем описать ноды, их свойства и ассоциации.





Для экспорта/импорта в таком формате можно посмотреть на

org.alfresco.tools.Export

org.alfresco.tools.Import

<https://wiki.alfresco.com/wiki/Export_and_Import>

Нужно определить удастся ли их применить для наших целей и насколько они работоспособны в alfresco 5. Если не получится использовать, то придется написать что-то свое.

Посмотреть на примеры такого формата можно по следующему пути:

ecos-community/*MODULE\_NAME*-repo/config/alfresco/templates/bootstrap/ru/citeck/\*

Import

а) Если у нас отсутствует параметр language или он не соответствует *http://www.alfresco.org/view/repository/1.0*, то импортируем условие в виде *cond:evaluate-script*. Берем содержимое объекта condition, удаляем оттуда CDATA и ${…} если они есть и записываем получившийся текст в свойство *cond:evaluate-script:script*.

б) Если параметр language равен *http://www.alfresco.org/view/repository/1.0*, то следует импортировать условия как Alfresco Repository View.

Таблица 1 – Conditions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EcoS | | | CMMN | |
| cond:condition [] | count > 1 || type != *cond:evaluate-script* | *http://www.alfresco.org/view/repository/1.0* | language: URI | Expression |
| *Condition nodes as «Alfresco Repository View»* | body: String |
| else | - | language: URI |
| <![CDATA[${  condition.properties[‘*cond:evaluate-script:script*’]  }]]> | body: String |

1. Event

**Types: *icaseEvent:activityStartedEvent***

***icaseEvent:activityStoppedEvent***

***icaseEvent:caseCreated***

Событие EcoS может быть представлено как *Sentry* в CMMN. В *Sentry* есть две части: *OnPart* и *IfPart*. В нашей системе реализован подтип *OnPart* – *PlanItemOnPart*.

*iEvent:type, iEvent:eventSource*

Эти свойства согласно CMMN сохраняются в объекте *PlanItemOnPart*.

- *iEvent:type* – строка, описывающая тип события. Ей соответствует свойство standartEvent.

Таблица 2 – Соответствие типов событий

|  |  |
| --- | --- |
| EcoS – iEvent:type | CMMN – standartEvent |
| activity-started | start |
| activity-stopped | complete |
| case-created | create |

- *iEvent:eventSource* – ассоциация с источником события. Ей соответствует свойство sourceRef.

Здесь отдельно стоит отметить, что в EcoS у типа icaseEvent:caseCreated источником события является кейс. При экспорте в CMMN источником события должен стать этап casePlanModel в кейсе.

*iEvent:conditions*

Эти ассоциации согласно CMMN сохраняются в объекте IfPart в свойстве «condition» с типом Expression. Для трансформации cond:condition в Expression и обратно см. раздел 1.

Таблица 3 – Event

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EcoS | | CMMN | | |
| icaseEvent:activityStartedEvent  icaseEvent:activityStoppedEvent  icaseEvent:caseCreated | iEvent:type | standartEvent | PlanItemOnPart | Sentry |
| iEvent:eventSource | sourceRef |
| iEvent:conditions | Condition | IfPart |

**Type: *icaseEvent:stageChildrenStopped***

Данное событие следует спрятать из интерфейса и заменить его флагом autoComplete (см. раздел «Доработки»). Экспортировать и импортировать его не нужно.

**Type: *iEvent:userAction***

Для экспортирования userAction необходимо создать помимо Sentry, PlanItemOnPart и IfPart дополнительно UserEventListener.

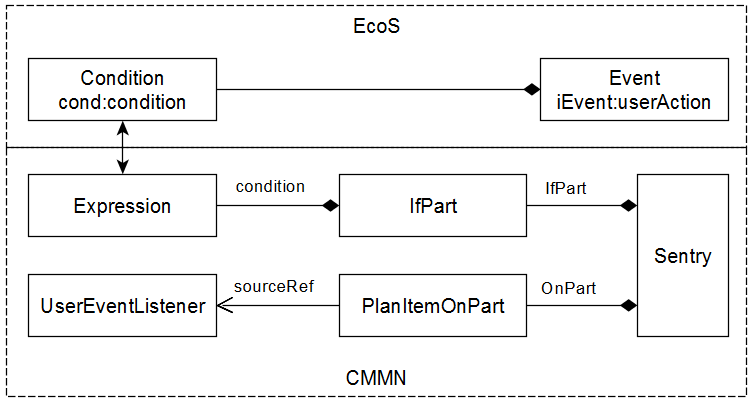


Рисунок 1 – Экпорт *iEvent:userAction*

Примечания:

- Свойство *standartEvent* в *PlanItemOnPart* должно равняться «occure».

- Свойство *iEvent:type* необходимо сохранить в поле *otherAttributes* объекта UserEventListener

Таблица 4 – User action event

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EcoS | | CMMN | | |
| iEvent:userAction | «occure» | standartEvent | PlanItemOnPart | Sentry |
| - | sourceRef  =  UserEventListener PlanItem\* |
| iEvent:conditions | condition | IfPart |
| icaseRole:authorizedRoles | authorizedRoleRefs | UserEventListener | |
| iEvent:type | otherAttributes |

\* UserEventListener – это PlanItemDefinition и у него должен быть инстанс – PlanItem, на который уже указывает sourceRef в PlanItemOnPart.

1. Activity

**Type: *activ:activity***

Все активности при экспорте следует разбивать на три части – PlanItemDefinition, PlanItem и PlanItemControl. PlanItemDefinition – это описание элемента плана, а PlanItem – его инстанс (близкая аналогия – класс и инстанс класса в Java). PlanItemControl определяет 2 параметра: возможность ручного старта и «повторяемость» активности.

В Citeck EcoS не предполагается наличие нескольких инстансов одного definition’а, поэтому при экспорте активности разбиваются строго на 1 PlanItemDefinition, 1 PlanItem и 1 PlanItemControl.

В PlanItem есть возможность переопределения некоторых свойств definition’а, но этого следует избегать, чтобы не вносить лишний хаос в структуру экспортируемого файла без необходимости. Таким образом, надо стараться всю информацию об активности разместить в PlanItemDefinition, а в PlanItem оставить только необходимое.

PlanItemControl

Содержит ManualActivationRule и RepetitionRule:

У нас вместо этих правил есть просто флаги (true/false) в *activity*, а именно – *activ:manualStarted* и *activ:repeatable*. Таким образом, получаем два варианта:



или



Аналогично и для *RepetitionRule*.

PlanItemDefinition

Контейнер для свойств и PlanItemControl’а. Непосредственно с PlanItemDefinition работать не придется. Необходимо будет работать с его наследниками: *Task*, *EventListener* и *Stage.*

PlanItem

PlanItem содержит критерии начала (entryCriteria) и завершения (exitCriteria). В EcoS этому соответствуют ассоциации icaseEvent:activityStartEvents и icaseEvent:activityEndEvents, которые указывают на iEvent:event. EntryCriterion и ExitCriterion наследуются от Criterion и не привносят никаких новых полей, а Criterion имеет ссылку на Sentry. Сответственно задача сводится к преобразованию iEvent:event в Sentry (см. раздел 2).

Таблица 5 – Activity

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EcoS | | CMMN | | |
| activ:activity | cm:name | name | | PlanItemDefinition |
| activ:plannedStartDate | otherAttributes | |
| activ:plannedEndDate |
| activ:actualStartDate |
| activ:actualEndDate |
| activ:manualStopped |
| activ:index |
| cm:title |
| stages:startCompletenessLevelsRestriction |
| stages:stopCompletenessLevelsRestriction |
| activ:manualStarted | manualActivationRule | PlanItemControl |
| activ:repeatable | repetitionRule |
| icaseEvent:activityStartEvents | entryCriteria | | planItem |
| icaseEvent:activityEndEvents | exitCriteria | |

1. Task

***Type: icaseTask:task***

Задаче в Citeck EcoS соответствует ProcessTask в CMMN.

Все свойства и ассоциации с target типом *icaseRole:role* описанные в Task + информацию о типе нужно сохранить в otherAttributes. Исключения – свойства, которые уже записаны в другие поля. Если ассоциация множественная, то следует объединить UUID’ы через запятую.

Process – BPMN процесс, который следует запустить при запуске задачи. В нем описывается implementationType, который в нашей системе всегда равен «http://www.omg.org/spec/CMMN/ProcessType/BPMN20» и name – имя процесса (в нашей системе Id процесса).

Таблица 6 - Task

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EcoS | | CMMN | | |
| icaseTask:task | «http://www.omg.org/spec/CMMN/ProcessType/BPMN20» | implementationType | Process | ProcessTask  planItem |
| icaseTask:workflowDefinitionName | name |
| True | isBlocking | |
| Все не сохраненные свойства | otherAttributes | |
| Ассоциации с target типом *icaseRole:role* |
| Тип объекта |

1. Action

**Type: *action:action***

Специальной сущности для действий в CMMN не предусмотрено и Task с параметром isBlocking = false лучше всего подходит для этой роли.

Таблица 6 - Action

|  |  |  |  |
| --- | --- | --- | --- |
| EcoS | | CMMN | |
| action:action | false | isBlocking | Task  planItem |
| Все не сохраненные свойства | otherAttributes |
| Тип объекта |

1. Stage

**Type: *stages:stage***

У этапа есть дочерние активности activ:activities. Необходимо их разбить на PlanItemDefinition, PlanItem и сохранить внутри того же этапа в полях planItems и planItemDefinitions. Все события в пределах этапа нужно преобразовать в Sentry и сохранить в поле sentries.

Таблица 7 – Stage

|  |  |  |  |
| --- | --- | --- | --- |
| EcoS | | CMMN | |
| stages:stage | activ:activities | planItems | Stage  planItem |
| sentries |
| planItemDefinitions |
| stages:autoComplete | autoComplete |
| stages:documentStatus | otherAttributes |
| cm:title |

1. Role

**Type: *icaseRole:role***

У ролей в CMMN есть ссылка на кейс, в котором они находятся. В EcoS можно получить эту ссылку просто взяв parent у ноды роли.

Ассоциации icaseRole:assignees следует экспортировать в виде Authority Names разделенных через запятую (см. RepoUtils.java getAuthorityName(…)). При импорте следует проверить наличие импортируемых Authority Name в системе и если их нет, то такие имена следует игнорировать.

Таблица 9 – Role

|  |  |  |  |
| --- | --- | --- | --- |
| EcoS | | CMMN | |
| icaseRole:role | cm:name | name | Case  planItem |
| role.parent | case |
| cm:title | otherAttributes |
| icaseRole:assignees |

1. Case

**Aspect: *icase:case***

В CMMN у кейса есть casePlanModel, который является этапом, внутри которого хранятся все элементы плана. В Citeck EcoS для этих целей используется сам кейс, поэтому при экспорте нужно создать новый этап, поместить его в свойство casePlanModel и в нем разместить все активности из кейса. При импорте соответственно содержимое этого этапа должно оказаться в кейсе.

Таблица 8 – Case

|  |  |  |  |
| --- | --- | --- | --- |
| EcoS | | CMMN | |
| icase:case | cm:name | name | Case  planItem |
| icaseRole:roles | caseRoles |
| новый stages:stage | casePlanModel |

Полезные ссылки

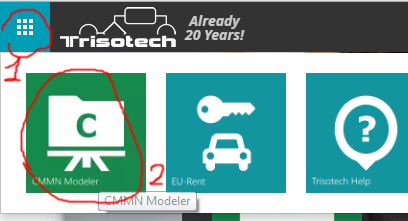
1. Пояснения, как в камунде реализован CMMN (один из живых примеров реализации):

<https://docs.camunda.org/manual/7.4/reference/cmmn11/>

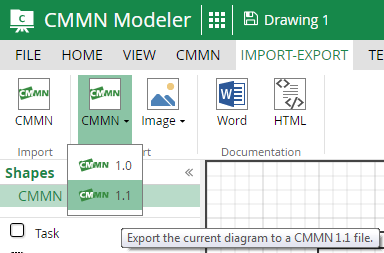
1. Веб-редактор для CMMN:

<http://www.cmmnwebmodeler.com/>

Для доступа необходима регистрация, после которой выдадут доступ к редактору на 30 дней. Валидации реальности данных похоже нету (основное - email). Редактор работает в браузере и чтобы в него зайти надо в левом верхнем углу найти кнопку меню:



В редакторе есть возможность экспорта в файл:



Мы работаем с CMMN 1.1 и при экспорте следует выбирать эту версию.

1. Ссылка на стандарт CMMN 1.1:

<http://www.omg.org/spec/CMMN/1.1/Beta/>