IBM Cúram Social Program Management
Version 6.0.5

Guide de référence des flux de travaux Cúram

Important
Avant d'utiliser le présent document et le produit associé, prenez connaissance des informations contenues dans la section «Remarques», à la page 139

Dernière révision : Mars 2014

Cette édition s'applique à IBM Cúram Social Program Management version 6.0.5 et à toutes les versions ultérieures, sauf indication contraire dans les nouvelles éditions.

Compagnie IBM France
Direction Qualité
17, avenue de l'Europe
92275 Bois-Colombes Cedex

Table des matières

Figures

Tableaux

Avis aux lecteurs canadiens

Le présent document a été traduit en France. Voici les principales différences et particularités dont vous devez tenir compte.

Illustrations

Les illustrations sont fournies à titre d'exemple. Certaines peuvent contenir des données propres à la France.

Terminologie

La terminologie des titres IBM peut différer d'un pays à l'autre. Reportez-vous au tableau ci-dessous, au besoin.

IBM France	IBM Canada
ingénieur commercial	représentant
agence commerciale	succursale
ingénieur technico-commercial	informaticien
inspecteur	technicien du matériel

Claviers

Les lettres sont disposées différemment : le clavier français est de type AZERTY, et le clavier français-canadien de type QWERTY.

OS/2 et Windows - Paramètres canadiens

Au Canada, on utilise :

- les pages de codes 850 (multilingue) et 863 (français-canadien),
- le code pays 002,
- le code clavier CF.

Nomenclature

Les touches présentées dans le tableau d'équivalence suivant sont libellées différemment selon qu'il s'agit du clavier de la France, du clavier du Canada ou du clavier des États-Unis. Reportez-vous à ce tableau pour faire correspondre les touches françaises figurant dans le présent document aux touches de votre clavier.

France	Canada	Etats-Unis
(Pos1)		Home
Fin	Fin	End
(PgAr)		PgUp
(PgAv)		PgDn
Inser	Inser	Ins
Suppr	Suppr	Del
Echap	Echap	Esc
Attn	Intrp	Break
Impr écran	ImpEc	PrtSc
Verr num	Num	Num Lock
Arrêt défil	Défil	Scroll Lock
(Verr maj)	FixMaj	Caps Lock
AltGr	AltCar	Alt (à droite)

Brevets

Il est possible qu'IBM détienne des brevets ou qu'elle ait déposé des demandes de brevets portant sur certains sujets abordés dans ce document. Le fait qu'IBM vous fournisse le présent document ne signifie pas qu'elle vous accorde un permis d'utilisation de ces brevets. Vous pouvez envoyer, par écrit, vos demandes de renseignements relatives aux permis d'utilisation au directeur général des relations commerciales d'IBM, 3600 Steeles Avenue East, Markham, Ontario, L3R 9Z7.

Assistance téléphonique

Si vous avez besoin d'assistance ou si vous voulez commander du matériel, des logiciels et des publications IBM, contactez IBM direct au 1 800 465-1234.

Référence des flux de travaux Cúram

Le système Cúram Workflow Management est utilisé pour définir des processus afin d'atteindre certains objectifs métier. La définition de processus est le composant central qui décrit le processus métier. Les métadonnées de définition de processus constituent le concept de niveau supérieur d'une définition de processus. Elles contiennent des informations permettant d'identifier et de décrire la définition de processus. Des descriptions détaillées des métadonnées de flux de travaux peuvent être entrées. Les effets de ces métadonnées lors de l'exécution peuvent être définis.

Introduction

Présentation

Il s'agit du Guide de référence du flux de travaux conçu pour fournir des explications détaillées fournies sur les concepts de Cúram Workflow Management System (WMS). Il a pour objectif de décrire comment définir un processus pour atteindre certains objectifs en donnant une description approfondie des métadonnées de flux de travaux ainsi que des effets de ces métadonnées lors de l'exécution. Ce document n'est pas conçu comme un document tutoriel mais une brève description de toutes les fonctions disponibles dans le flux de travaux Cúram.

Conditions préalables

Ce document suppose une certaine familiarité avec les concepts de flux et la façon dont ils sont réalisés dans Cúram WMS. En particulier, elle suppose que vous avez au moins lu les *Guides d'analyste métier : Guide de présentation de flux de travaux Cúram*.

Comment utiliser ce document

Etant donné que ce document est un manuel de référence, les chapitres sont indépendants les uns des autres autant que possible. L'objectif est que le lecteur découvre un concept sur lequel il souhaite obtenir plus de détails, trouve le chapitre approprié et n'ait besoin de lire que ce chapitre. Puisque qu'il n'est pas prévu que ce document soit lu intégralement, il a été structuré de telle manière que la lecture soit possible et productive.

Certaines parties de Cúram WMS s'inspirent fortement les unes des autres et la documentation reflète cet état de fait. Ces références externes sont présentées en deux versions : les conditions préalables qui sont des pointeurs vers des informations qui sont indispensables pour comprendre la section abordée et les liens généraux qui soulignent les informations associées mais non requises.

Structure de ce document

Ce document peut également être consulté dans un certain nombre de sections distinctes dont chacune correspond à un domaine de Cúram WMS et la manière dont elles interagissent les unes avec les autres. Les sections suivantes incluent un récapitulatif du contenu de ces sections logiques, les chapitres inclus dans ces sections logiques et les domaines de Cúram WMS qui sont abordés dans ces chapitres associés.

Processus de flux de travaux

La section *Processus de flux de travaux* du document décrit les métadonnées associées à une définition de processus de flux de travaux. Le cycle de vie d'une définition de processus est également décrit.

«Création d'un processus de flux de travaux», à la page 5 décrit comment créer et visualiser un processus de flux de travaux en utilisant le système de flux de travaux Cúram. La publication d'un processus est également décrite et l'effet qu'elle a sur les versions associées à des définitions de processus est également détaillé. L'importation et l'exportation des définitions de processus sont discutées et la localisation du texte contenu dans un processus est traitée. L'exécution d'un processus de flux de travaux utilisant le moteur de flux de travaux Cúram est décrite en détail. Une description de la bibliothèque de méthodes et de la bibliothèque de modèles d'objet de données de flux de travaux (voir «Objets de données de flux de travaux», à la page 17) est également fournie.

«Métadonnées d'une définition de processus», à la page 13 décrit les caractéristiques associées à une définition du processus de flux de travaux. Chaque zone de métadonnées est soulignée et l'ensemble des validations et des objets de données de flux de travaux du contexte est détaillé.

Flux de données

La section *Flux de données* du document décrit comment les données sont stockées et manipulées dans une instance de processus. En particulier, les problèmes sur la manière dont les données sont transmises de l'extérieur (lors du lancement des processus) et entre les activités et les transitions dans le processus sont décrits.

«Objets de données de flux de travaux», à la page 17 décrit les objets utilisés pour gérer et transmettre les données dans le moteur de flux de travaux. Les métadonnées qui constituent les objets de données de flux de travaux et leurs attributs sont décrites en détail. Les validations qui se rapportent à la création et à la modification des objets de données de flux de travaux sont abordées. Enfin, les objets de données de flux de contexte qui sont mis à disposition par l'outil de définition de processus et le moteur de flux de travaux sont également décrits dans ce chapitre.

«Adoption d'un processus», à la page 26 décrit le démarrage d'une instance de processus (par exemple, l'exécution du travail défini dans la définition de processus). L'API de service de lancement est décrit et les mappages d'adoption des métadonnées associées à l'adoption d'un processus sont abordés. Les exemples de validation et de code associés sont également fournis. Il est possible de démarrer un processus en réponse à un événement émis et cela est également décrit dans ce chapitre. Les données de configuration pour effectuer cette action est décrite en détail. Toutes les validations exécutées lors de la création des mappages entre des événements et des processus de flux de travaux sont décrits.

Activités

Les activités sont centrales dans un processus de flux de travaux parce qu'elles constituent les étapes dans lesquelles le traitement métier du flux de travaux a lieu. Il existe différents types d'activités pris en charge par Cúram WMS et elles sont toutes décrites dans la section *Activités* du document. Lorsque les notifications sont également pertinentes à chaque type d'activité, elles sont également décrites dans cette section du document.

«Activité de base», à la page 32 décrit les détails des métadonnées communs à tous les types d'activités pris en charge dans le système de flux de travaux Cúram. Les

validations exécutées lors de la création ou de la modification d'une activité sont également présentées. Enfin, certains types d'activité plus simple sont décrits, y compris l'activité de routage et les activités de processus de début et de fin.

«Automatique», à la page 34 décrit les caractéristiques des métadonnées associées à une activité automatique. Les mappages d'entrée et de sortie spécifiés pour la méthode associée à l'activité automatique sont décrits en détail. Les validations exécutées lors de la création ou la modification des métadonnées pour une activité automatique sont décrits. Enfin, les objets de données de flux de travaux de `Context_Result` et `Context_Error` qui sont disponibles pour utilisation dans les transitions à partir des activités automatiques sont également décrits dans ce chapitre.

«Attente d'événement», à la page 49 décrit les caractéristiques des métadonnées associées à une activité automatique. Ceci inclut la liste des événements, les détails de l'échéance (y compris les rappels d'échéance) associée à une attente d'un événement et également les mappages de sortie qui peuvent être spécifiés. Les validations exécutées lors de la création ou de la modification des métadonnées d'attente d'événement sont également décrites. Les informations d'exécution qui sont associées à l'exécution des activités d'attente d'événement par le moteur de flux de travaux sont également décrites en détail. Enfin, les objets de données de flux de travaux de `Context_Event` et `Context_Deadline` qui sont disponibles pour utilisation dans les transitions à partir des activités d'attente d'événement sont également décrits dans ce chapitre.

«Manuel», à la page 61 décrit les caractéristiques des métadonnées associées à une activité manuelle. Cela inclut les détails de la tâche manuelle, la stratégie d'allocation, les associations d'objet métier et l'attente d'événement associée à l'activité manuelle. Les validations exécutées lors de la création ou la modification des métadonnées pour une activité manuelle sont également décrites. Les informations d'exécution qui sont associées à l'exécution des activités manuelle par le moteur de flux de travaux sont également décrites en détail. Enfin, une description de l'objet des données de flux de travaux de `Context_Task` qui est disponible pour une utilisation dans les différents mappages associés à une activité manuelle est également fournie dans ce chapitre.

«Décision», à la page 80 décrit les caractéristiques des métadonnées associées à une activité de décision. Ces métadonnées incluent les détails de la tâche de décision (qui est similaire à celle des détails de la tâche d'activité manuelle) et les détails de la question à choix multiple et des questions de texte libre. Les diverses validations exécutées lors de la création ou de la modification des détails de la tâche ou question associée à une activité de décision sont décrites. Ce chapitre comprend également une description des informations d'exécution qui est présente lorsque le moteur de flux de travaux exécutait une activité de décision. Une description de l'objet de données de flux de travaux `Context_Decision` est également fournie dans ce chapitre.

«Flux secondaire», à la page 89 décrit les caractéristiques des métadonnées associées à une activité de sous-flux. Cela inclut les détails du processus de flux secondaire associé à l'activité de flux secondaire et les mappages d'entrée requis pour appliquer ce processus de sous-flux. Il existe plusieurs validations exécutées lors de la création ou de modification de ces métadonnées et une description de ces éléments est également fournie dans ce chapitre.

«Début et fin de boucle», à la page 94 décrit les caractéristiques des métadonnées associées à une activité de boucle de début et de boucle de fin. Le type de boucle,

la condition de boucle et la référence de l'activité de fin de boucle sont décrits. Ce chapitre comprend également une description des informations d'exécution qui sont présentes lorsque le moteur de flux de travaux exécutait une boucle dans une définition de processus de flux de travaux. Une description de l'objet de données de flux de travaux `Context_Loop` est également fournie dans ce chapitre.

«Parallel», à la page 97 décrit les caractéristiques des métadonnées associées à une activité parallèle. Les activités parallèles incluent des types d'activité y compris les activités «Manuel», à la page 61 et les activités «Décision», à la page 80. Puisque les métadonnées associées à ces types d'activité restent les mêmes, cela ne sera pas décrit à nouveau dans ce chapitre. Les validations exécutées lors de la création ou la modification des métadonnées d'activité sont également décrites. Les informations d'exécution qui sont associées à l'exécution des activités parallèles par le moteur de flux de travaux sont également décrites en détail. Enfin, une description de l'objet des données de flux de travaux de `Context_Parallel` qui est disponible pour une utilisation dans les différents mappages associés à une activité manuelle est également fournie dans ce chapitre.

«Notifications d'activité», à la page 102 décrit les caractéristiques des métadonnées associées à une notification d'activité. Ces descripteurs comprennent le mécanisme de distribution, l'objet, le corps, la stratégie d'allocation et les actions associées à la notification. Il existe plusieurs validations exécutées lors de la création ou de modification de ces métadonnées et une description de ces éléments est également fournie dans ce chapitre. Une description des informations d'exécution lorsque le moteur de flux de travaux crée une notification est également fournie. Enfin, il y a un certain nombre de détails d'implémentation qui sont requis lors de l'application Cúram pour permettre de livrer correctement les notifications. Ils sont également décrits dans ce chapitre.

Contrôle des flux

Un processus de flux de travaux modélise le flux d'informations à travers une organisation, en passant par un agent humain ou un logiciel d'ordinateur pour atteindre un objectif métier. La section *Flow Control* du document fournit des informations sur la manière dont le flux d'informations (entre les activités) est spécifié et géré par Cúram WMS.

«Transitions», à la page 109 décrit les liens entre les activités. Les métadonnées associées aux transitions sont décrites en détail. Les validations qui se rapportent à la création et à la modification des transitions sont également abordées. Les informations d'exécution qui sont associées à l'exécution des transitions par le moteur de flux de travaux sont également décrites.

«Conditions», à la page 112 décrit la construction de métadonnées de définition de processus qui représente une condition. Les validations qui se rapportent à la création et à la modification des conditions sont également abordées.

«Fractionnement/Jointure», à la page 116 décrit les métadonnées associées aux fractionnements et jointures de l'activité, le moment où elles doivent être utilisées, ainsi que les différents types disponibles.

«Structure d'un flux de travaux», à la page 118 décrit la structure d'un processus de flux de travaux telle que déterminée par les activités dans le processus et les transitions entre elles. Les contraintes existantes lors de la construction d'une définition de processus afin de garantir qu'il s'agit d'une structure de bloc valide sont décrites. Les validations exécutées dans le cadre de ces contraintes sont également abordées.

Développement et exécution

La section *Développement et exécution* du document décrit les caractéristiques de l'environnement de développement et d'exécution pour les flux de travaux Cúram. Plus précisément, il détaille comment exécuter, configurer et déboguer les flux de travaux.

«Services Web de flux de travaux», à la page 124 décrit les étapes nécessaires pour permettre le lancement des processus via les services Web en exposant le processus de flux de travaux Cúram en tant que services Web.

«Emplacements des fichiers», à la page 126 fournit des détails sur l'emplacement où les différentes sorties des utilitaires tels que l'outil de définition de processus et les autres interfaces utilisateur d'administration sont exportés et soumis au contrôle de version. Ces produits incluent les fichiers de métadonnées de définition de processus ainsi que les fichiers source associés à des événements.

«Configuration», à la page 128 décrit le flux de travaux associé aux propriétés d'application, leurs noms, leurs paramètres par défaut et ce pour quoi ils sont utilisés dans le système de flux de travaux Cúram.

«JMSLite», à la page 130 fournit des détails sur le serveur Cúram léger qui peut s'exécuter en même temps que l'environnement de test RMI dans un environnement de développement intégré pris en charge. Les étapes requises pour démarrer le serveur JMSLite sont exposées et une description détaillée de la façon de déboguer les flux de travaux à l'aide de JMSLite est également abordé.

Configuration et personnalisation de la boîte de réception

La section *Configuration et personnalisation de la boîte de réception* section du document décrit les options de configuration et de personnalisation qui sont disponibles dans la section Gestion de la boîte de réception et des tâches du Cúram WMS. Plus précisément, il détaille la manière de configurer le nombre de tâches qui sont affichées sur les différentes listes affichées dans la Boîte de réception et également la manière de personnaliser les actions de Gestion de la boîte de réception et des tâches qui sont disponibles sur le système.

«Boîte de réception et gestion des tâches», à la page 132 décrit les options de configuration disponibles pour être utilisées dans la Boîte de réception. Il décrit également comment personnaliser les fonctions Gestion de la boîte de réception et des tâches à travers l'utilisation de l'infrastructure Google Guice.

Création d'un processus de flux de travaux

Cycle de vie d'une définition de processus

La définition de processus est le concept central de n'importe quel système de flux. De ce fait, la manière dont elle est créée et utilisée est d'une importance capitale. Ce chapitre décrit les fonctions fournies par le système de flux de travaux Cúram pour créer et administrer des définitions de processus.

Création d'un processus

Le système de flux de travaux Cúram fournit un *outil de définition de processus* (PDT) pour créer et gérer des définitions de processus, qui sont ensuite interprétées par le moteur de flux de travaux. La création d'une définition de processus implique l'utilisation de l'outil de définition de processus pour décrire le comportement de processus souhaité en termes d'activités et de transitions.

Un certain nombre d'utilitaires sont fournis en tant que partie de l'outil de définition de processus qui peut aider à la création du processus. L'outil PDT permet de visualiser une définition de processus au cours de la conception. Les processus peuvent également être copiés, importés et exportés à l'aide de PDT.

Visualisation d'un processus

Un utilitaire graphique en lecture seule est fourni en tant que composant de l'outil de définition de processus qui permet aux administrateurs de processus de visualiser les processus lorsqu'ils sont en cours de création ou de modification. Cet outil permet aux administrateurs d'afficher toutes les activités et les transitions dans une définition de processus et fournit une vue de haut niveau de tous les chemins possibles via le processus de flux de travaux lors de l'exécution. L'exemple d'une représentation graphique d'une définition de processus de flux de travaux est affiché ci-après.

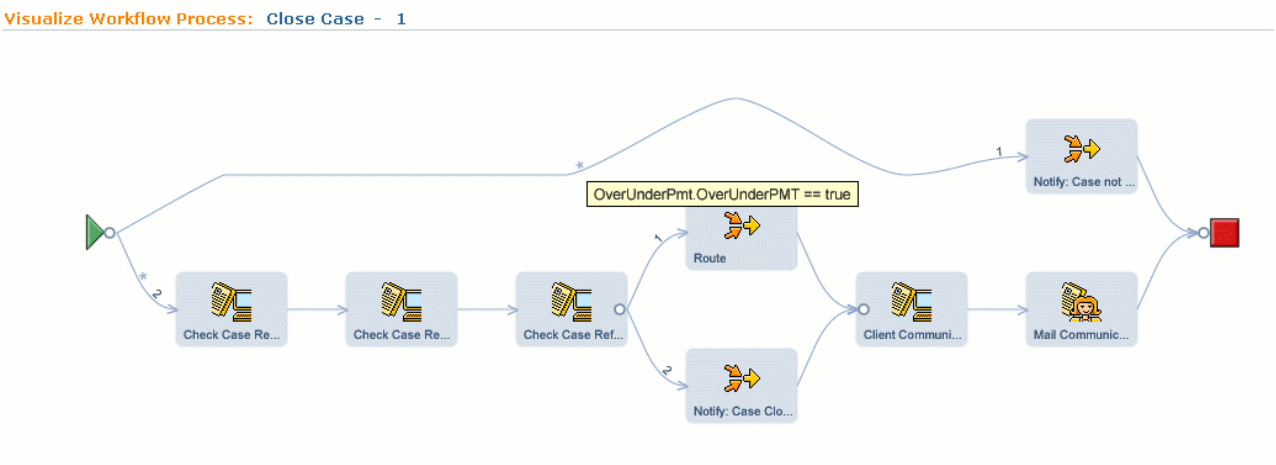

Figure 1. Visualisation d'une définition de processus de flux de travaux de clôture de dossier

Le processus visualisé comprend un certain nombre de noeuds sur un graphe représentant les activités dans le processus. Les noeuds sont reliés par des arêtes de graphe et ils reflètent les transitions définies dans la définition de processus. Cliquez sur une activité dans le graphe affiche les détails de l'activité dans le PDT. De similaire, en cliquant sur une transition entre les activités sur le graphe, vous affichez les détails de la transition dans le PDT.

L'outil graphique affiche les informations suivantes pour chaque processus visualisé :

- Le type et le nom de chaque activité. Chaque type d'activité est identifié par une icône spécifique.
- Les notifications définies pour chaque activité (Voir «Notifications d'activité», à la page 102). Si une activité est associée à une notification, elle sera représentée sous la forme d'une enveloppe sur laquelle il est possible de cliquer via la page de notification d'activité associée.
- Le type fractionnement/jointure (Voir «Fractionnement/Jointure», à la page 116) pour chaque activité. Un type de fractionnement ou de jonction "choix" sur une activité est représenté par un cercle, alors qu'un type de fractionnement ou de jonction "parallèle" est représenté par un carré.
- Les transitions entre les activités. Lorsqu'une transition entre les activités possède une condition de transition associée (Voir «Conditions», à la page 112),

elle est représentée par un astérisque. Les détails relatifs à la condition sont affichés lorsque la souris est placée sur cet astérisque.
- L'ordre de chaque fractionnement de type choix (Voir «Fractionnement/Jointure» , à la page 116) à partir d'une activité. Etant donné que l'ordre d'un fractionnement de type choix à partir d'une activité est important, la première transition éligible sur la liste sera suivie), l'ordre de chaque transition à partir de l'activité est affiché en tant que numéro de cette transition.

Publication d'un processus

Une fois qu'une définition de processus a été créé et est prête à être utilisée, elle doit être publiée avant de pouvoir être exécutée par le moteur de flux de travaux (, voir «Exécution d'un processus», à la page 9). En même temps qu'un processus est publié à l'aide du PDT, il est examiné pour s'assurer que toutes les informations dont le moteur a besoin pour exécuter le processus sont présentes et cohérentes en interne. Les validations requises pour publier un processus sont décrites dans les différentes sections de métadonnées de ce document.

Seuls les processus qui ont adopté toutes les validations requises peuvent être publiés et mis à la disposition du moteur de flux de travaux. Une fois publiée, la définition de processus passe en lecture seule et ne peut plus être modifiée par l'outil de définition de processus sans générer une nouvelle version.

Versions de processus (édition d'un processus)

Des modifications peuvent être nécessaires pour un processus publié au fil du temps. Toutefois, puisqu'un processus publié est en lecture seule, une nouvelle version est requise avant que des modifications ne puissent être appliquées. La tentative d'éditer un processus publié dans l'outil PDT créera automatiquement une nouvelle version non publiée de ce processus.

Il ne peut y avoir qu'une seul version non publiée d'un processus à tout moment. Si l'administrateur souhaite éditer un processus validé, les versions non existantes doivent d'abord être publiées ou supprimées.

Importation, exportation et copie d'un processus

La fonctionnalité d'importation et d'exportation permet aux développeurs de proposer des définitions de processus en fonction des besoins. Par exemple. Une définition de processus peut être développée sur un système de développement et déplacée seulement vers un système de production après que des tests aient été effectués.

L'exportation d'un processus envoie les métadonnées de processus vers un système de fichiers. Ces métadonnées peuvent alors être importées à l'aide de l'option du processus dans le PDT. Le numéro de version le plus élevé sera affecté à un processus importé de cette manière. Ledit processus ne sera pas publié quel que soit son état de publication lors de l'importation. Cela permet de garantir que les définitions de processus importées sont soumises aux mêmes validations de publication au même titre que d'autres définitions au niveau local. Une option d'écrasement est disponible lors de l'importation afin de garantir que toute version non publiée du processus est remplacée par la version importée.

Il peut exister des situations où une définition de processus diffère très légèrement d'une autre dans le système de flux de travaux. Une option de processus `copy` est disponible, ce qui permet de copier un processus existant vers un nouveau processus lorsque nécessaire. Le nouveau processus sera toujours non publié lorsqu'il est copié avec une version définie à 1, quel que soit l'état du processus original.

Validations :

- Une définition de processus ne peut pas être importée si une version non publiée d'un processus existe déjà avec le même nom, et que l'option d'écrasement n'a pas été sélectionnée.
- Une définition de processus ne peut pas être importée si aucun nom n'a été spécifié pour ce processus.
- Une définition de processus ne peut être importée si un processus existe déjà avec le même nom et un identificateur de processus différent. Cette validation permet de vérifier qu'une définition importée ne peut écraser par inadvertance une définition de processus existante, sauf si les identificateurs de processus correspondent.
- Lors de la copie d'un processus existant, le nom du nouveau processus doit être unique dans le système de flux de travaux.
- La longueur du nom de la définition de processus de flux de travaux en cours d'importation ne doit pas dépasser la longueur maximale autorisée pour un nom, par exemple. Le nom doit inclure au maximum 254 caractères.
- La longueur des noms de définition de processus de flux de travaux en cours d'importation ne doit pas dépasser la longueur maximale autorisée pour un nom, par exemple. Le nom doit inclure au maximum 75 caractères.
- La longueur des noms de définition de processus de flux de travaux en cours d'importation ne doit pas dépasser la longueur maximale autorisée pour un nom, par exemple. Le nom doit inclure au maximum 75 caractères.
- Toutes les valeurs de la table de code qui sont contenues dans la définition de processus du flux de travaux importée doivent être valides (par exemple, la table de code doit exister et le code spécifié doit exister dans cette table des codes).
- Pour chaque texte localisable dans la définition de processus en cours d'importation, il doit exister une entrée en anglais (c'est-à-dire pour l'environnement local "en"). Des entrées dans d'autres langues peuvent également exister (par exemple pour les différents environnements locaux d'utilisateur pris en charge par l'application), mais chaque traduction doit être accompagnée d'une entrée pour l'environnement local anglais.
- Les identificateurs d'activités, de transitions, d'expressions de conditions de transition, d'expressions de conditions de fin de boucle, d'événements et de mémentos doivent être uniques dans la définition de processus de flux de travaux en cours d'importation.

Localisation

Les définitions de processus de flux de travaux contiennent du texte de métadonnées qui doit être affiché dans différentes langues par différents utilisateurs. Par exemple, lorsqu'une activité manuelle est exécutée, elle crée une tâche qui possède un objet associé. L'outil de définition de processus permet au développeur de processus de localiser cette chaîne d'objet pour chacun des environnements locaux pris en charge par l'application.

Les chaînes localisables peuvent être identifiées dans une définition de processus par les métadonnées spécifiées dans «Texte localisé», à la page 33. Toutes les chaînes de texte localisables qui ont été spécifiées dans une définition de processus doivent avoir au moins une entrée correspondante pour l'environnement local anglais (c'est-à-dire "en"). Lors de l'ajout d'une chaîne localisée à une définition de processus, l'outil PDT par défaut ajoute la chaîne à l'environnement local anglais et à l'environnement local de l'utilisateur. Toute modification suivante du texte localisé (ajout, suppression ou modification) peut être apportée dans l'écran de localisation de l'outil PDT.

Voici une liste des chaînes de texte localisables qui peuvent être indiquées dans une définition de processus.

- Nom d'affichage du processus
- Description de processus
- Nom d'affichage d'objet de données de flux de travaux
- Description d'objet de données de flux de travaux
- Nom d'affichage d'attribut d'objet de données de flux de travaux
- Nom de l'activité
- Description de l'activité
- Message de tâche d'activité manuelle
- Message d'action de tâche d'activité manuelle
- Message de tâche d'activité manuelle parallèle
- Message d'action de tâche d'activité manuelle parallèle
- Message d'action d'activité de décision
- Message de question d'activité de décision
- Message d'action secondaire d'activité de décision
- Valeur d'affichage de réponse d'activité de décision
- Message d'action d'activité de décision parallèle
- Message de question d'activité de décision parallèle
- Message d'action secondaire d'activité de décision parallèle
- Valeur d'affichage de réponse d'activité de décision parallèle
- Message d'objet du message de notification d'activité
- Message du corps de notification d'activité
- Message d'action de notification d'activité
- Message d'objet de notification de rappel
- Message du corps de notification de rappel
- Message d'action de notification de rappel

L'API `LocalizableStringResolver` fournit des routines qui résolvent et renvoient les différentes chaînes localisables pour des tâches et des notifications qui existent dans une définition de processus de flux de travaux pour les paramètres locaux de l'utilisateur en cours. Lorsqu'une chaîne de texte n'a pas été localisée pour l'environnement local de l'utilisateur courant, le texte en anglais (de l'environnement local "en") est renvoyé à la place.

Exécution d'un processus

Une définition de processus de flux de travaux décrit les tâches et flux d'un processus métier en des termes pouvant être interprétés par le système de gestion de flux de travaux Cúram. Pour effectuer le travail décrit dans la définition de processus spécifié, une instance de cette définition doit être créée et exécutée par le moteur de flux de travaux. Le mécanisme de réalisation de cette tâche est décrit dans cette section. Une instance de processus peut être considérée comme les données d'exécution pour une définition de processus de flux de travaux lancé.

Comportement du moteur de base

Le système de gestion des flux de travaux Cúram comprend un moteur de flux de travaux qui fournit l'environnement d'exécution nécessaire à une instance de processus. Plusieurs mécanismes sont disponibles pour un processus de flux et ils sont abordés dans «Adoption d'un processus», à la page 124. Lorsqu'un processus

est lancé, le moteur de flux de travaux examine la table de base de données appropriée et utilise la *dernière version publiée* de la définition de processus spécifiée pour créer l'instance de processus à exécuter.

Puisque chaque activité est exécutée, un enregistrement d'instance d'activité associé est créé et géré par le moteur de flux de travaux. Cet enregistrement contient les données d'exécution d'une instance d'activité du flux de travaux lancé. Au fur et à mesure que progresse le flux de travaux, le moteur évalue les transitions (voir «Transitions», à la page 109) pour les différentes activités afin de déterminer le chemin via le processus à suivre. Cela implique de déterminer les types de fractionnements et de jointures (voir «Fractionnement/Jointure», à la page 116) que l'activité possède et l'exécution de n'importe quelle condition (voir «Conditions», à la page 112) que les diverses transitions possèdent dans le processus. Les enregistrements d'instance de transition (qui contiennent les données d'exécution d'une transition de flux de travaux) pour chaque transition suivie dans le processus de flux de travaux sont également créés et gérés par le moteur.

Exécution de plusieurs versions

La modification et la publication d'une nouvelle version d'un processus n'affecteront aucune instance actuellement exécutée de ce processus. Un processus sera exécuté jusqu'à son terme dans le moteur de flux de travaux avec la version avec laquelle il a été démarré, quelles que soient les versions ultérieures qui ont été publiées.

Administration d'une instance de processus

Un administrateur a la possibilité d'influencer l'exécution d'une instance de processus en cours d'exécution via l'interface d'administration de flux de travaux Cúram. Les fonctions suivantes sont disponibles aux fins suivantes :

Suspendre une instance de processus
N'importe quelle instance de processus peut être suspendue. Lorsque cela se produit, le moteur de flux de travaux permettra aux instances d'activité *en cours* dans cette instance de processus de terminer. Cependant, l'ensemble des activités suivant qui doivent être exécutées pour cette instance de processus sont démarrées par le moteur de flux de travaux et immédiatement suspendues. N'importe quel processus de sous-flux synchrone (voir «Flux secondaire», à la page 89) associé à l'instance de processus en cours de suspension sera également suspendu par le moteur de flux de travaux.

Reprendre une instance de processus
Toute instance de ce processus de flux de travaux qui a été suspendue peut être reprise. Lorsque cela se produit, les instances d'activité qui ont été précédemment suspendues pour cette instance de processus sont redémarrées par le moteur de flux de travaux. N'importe quel processus de sous-flux synchrone (voir «Flux secondaire», à la page 89) associé à l'instance de processus en cours de suspension sera également suspendu par le moteur de flux de travaux.

Abandon d'une instance de processus
N'importe quelle instance de processus en cours d'exécution ou suspendue peut être abandonnée. Toutes les activités qui sont *en cours* dans l'instance de processus abandonnée sont complétées. Si le processus contient des activités manuelles ou de décision en cours, les tâches associées seront fermées par le moteur de flux de travaux lorsque l'instance de processus est abandonnée. Aucune nouvelle activité associée à une instance de processus interrompue ne sera lancée par le moteur de flux de travaux. N'importe quel processus de sous-flux (voir «Flux secondaire», à la page 89

89) associé à l'instance de processus sera également abandonné. Une instance de processus interrompue ne peut pas être reprise.

Bibliothèque de référence des méthodes

Il existe plusieurs situations dans le système de gestion des flux de travaux Cúram pour lesquelles il est nécessaire d'interagir avec l'application Cúram en appelant un processus métier ou des méthodes d'entité (voir «Les méthodes métier de Cúram», à la page 35 pour un exemple de cette interaction). Tout objet de processus métier (BPO) ou méthode d'entité dans l'application peut être appelé par le moteur de flux de travaux. Toutefois, il existe d'autres méthodes similaires beaucoup plus nombreuses à présenter à un concepteur de processus afin qu'il les utilise dans ses définitions de processus d'une manière acceptable. Le but de cette bibliothèque est de permettre à un administrateur d'attribuer des méthodes qui sont susceptibles d'être utilisées dans les définitions de processus sur une liste facilement gérable pour une utilisation dans la conception de processus. Evidemment, il n'est pas nécessaire de préremplir la bibliothèque avec toutes les méthodes qui pourraient être utilisées à l'avenir. De nouvelles méthodes peuvent être ajoutées à la bibliothèque selon les besoins.

Référencement des méthodes Cúram

Un objet de processus métier (BPO) ou des méthodes d'entité doivent être ajoutés à la bibliothèque de référence des méthodes avant qu'ils ne puissent être référencés dans une définition de processus. Le type de méthode défini lors de l'ajout à la bibliothèque indique l'emplacement où cette méthode sera disponible pour une utilisation dans la définition d'un processus.

Il faut noter que la suppression d'une référence de méthode de la bibliothèque de référence de méthode ne la supprime pas de toutes les définitions de processus qui y font référence. Tant que la méthode reste une méthode d'application Cúram valide, toutes les définitions de processus qui y font référence resteront valides.

Types de méthode

Un objet de processus métier (BPO) Cúram ou une méthode d'entité doit être ajoutée à la bibliothèque de référence de méthode avec l'un des trois types de méthodes définis. Une méthode peut être associée à plusieurs types de méthode, mais elles devront être ajoutées à plusieurs reprises avec le type de méthode différent à chaque fois. Les différents types de méthode sont présentés ci-après, dans la bibliothèque de référence de méthode, ainsi que les restrictions concernant leur utilisation.

Dispositions générales

Les méthodes comportant un type *General* sont disponibles uniquement en tant que méthodes d'application à appeler à partir d'activités automatiques (voir «Les méthodes métier de Cúram», à la page 35). L'outil de définition de processus limite l'accès à ces méthodes uniquement lors de la sélection d'une méthode à appeler à partir d'une activité automatique.

Allocation

Les méthodes dans la bibliothèque avec un type *Allocation* sont uniquement disponibles pour une utilisation en tant que fonctions de stratégie d'allocation associées à des activités manuelles, des activités de décision, des activités parallèles et des notifications d'activité. (Voir «Stratégie d'allocation», à la page 68). Toutes les méthodes spécifiées avec un type de méthode d'allocation doivent avoir un type de retour de `curam.util.workflow.struct.AllocationTargetList`.

Echéance

Les méthodes de type *Deadline* dans la bibliothèque de méthodes peuvent seulement être référencées en tant que méthodes de gestionnaire avec des activités d'attente d'événement, de décision et des activités parallèles. (Voir «Echéance», à la page 53).

Modèles d'objet de données de flux de travaux

Les données sont gérées et transmises dans le moteur de flux de travaux en tant qu'objets de données de flux de travaux (voir: «Objets de données de flux de travaux», à la page 17) Les objets de données de flux de travaux qu'un processus peut utiliser sont définis dans la définition du processus lui-même. Toutefois, il est concevable que certains objets de données de flux de travaux soient utiles dans plusieurs définitions de processus. Par conséquent, l'idéal serait qu'ils puissent être importés à partir d'un pool au lieu de devoir être recréés dans chaque processus individuel. C'est pour cela que cette bibliothèque a été créée.

Métadonnées

```
<wdo is-list-wdo="false" initialize-attributes="false">
    <wdo-name>TaskCreateDetails</wdo-name>
    <display-name>
        <localized-text>
            <locale language="fr">TaskCreateDetailsName</locale>
        </localized-text>
    </display-name>
    <description>
        <localized-text>
            <locale language="fr">The Task Create Details WDO
              Template</locale>
        </localized-text>
    </description>
    <attributes>
        <attribute>
            <attribute-name>subject</attribute-name>
            <display-name>
                <localized-text>
                    <locale language="fr">Task Subject</locale>
                </localized-text>
            </display-name>
            <type>STRING</type>
            <required-at-enactment>false</required-at-enactment>
            <process-output>false</process-output>
            <constant-value/>
        </attribute>
        <attribute>
            <attribute-name>dueDate</attribute-name>
            <display-name>
                <localized-text>
                  <locale language="fr">Task Due Date</locale>
              </localized-text>
          </display-name>
          <type>DATE</type>
            <required-at-enactment>false</required-at-enactment>
            <process-output>false</process-output>
            <constant-value/>
        </attribute>
    </attributes>
  </wdo>
```

Les métadonnées définies pour les modèles d'objet de données de flux de travaux sont exactement les mêmes que celles définies pour les objets de données de flux de travaux. Pour une description complète de ces métadonnées, voir «Objets de

données de flux de travaux», à la page 17. La bibliothèque de modèles d'objet de données de flux est stockée dans la table de base de données WDOTemplateLibrary.

Il faut noter que l'élément `initialize-attributes` d'un objet de données de flux de travaux et les éléments `required-at-enactment`, `process-output` et `constant-value` d'un attribut d'objet de données de flux de travaux ne sont pas disponibles pour édition dans les modèles d'objet de données de flux de travaux et sont automatiquement initialisés dans les métadonnées associées.

Importation et synchronisation

Les modèles définis dans la bibliothèque de modèles d'objet de données de flux de travaux sont disponibles pour être utilisés lors de la création de définitions de processus. L'importation d'un modèle d'objet de données de flux de travaux à partir de la bibliothèque ajoute l'objet de données de flux de travaux et tous ses attributs à la définition de processus actuelle.

Une fois qu'un modèle d'objet de données de flux de travaux a été importé dans une définition de processus, il peut être synchronisé avec son entrée correspondante dans la bibliothèque de modèles d'objet de données de flux à tout moment. La synchronisation du modèle d'une définition de processus force la mise à jour du nom et du nom d'affichage de l'objet de données de flux de travaux à partir de la bibliothèque de modèles. Parallèlement, toutes les nouvelles entrées d'attribut qui existent dans l'entrée de la bibliothèque de modèle seront automatiquement ajoutées à l'objet de données du flux de travaux dans la définition du processus. L'utilisateur peut éventuellement décider de remplacer les attributs existants dans l'objet de données de flux de travaux par ceux de la bibliothèque de modèles lors de la synchronisation. Il convient de noter que le remplacement des attributs existants peut invalider la définition de processus et nécessiter des mises à jour dans lesquelles les anciennes valeurs d'attributs ont été utilisées.

Validations

- Il est impossible d'importer un objet de données de flux de travaux à partir d'un modèle s'il en existe déjà dans la définition de processus de flux de travaux associé du même nom.

Métadonnées d'une définition de processus

Présentation

Le processus est le concept de haut niveau dans une définition de processus. Il contient essentiellement des informations permettant d'identifier et de décrire la définition du processus. Ces informations incluent l'identificateur et la version de la définition de processus, il s'agit du nom et d'une brève description. Elles comprennent également une description de la stratégie d'allocation d'échec pouvant être spécifiée pour un processus. Les sections suivantes décrivent ces informations de haut niveau.

Métadonnées

```
<de flux de travaux de processus id="100" processus-version="2"
                    language-version="1.0"
                    released="false" category="PC5"
                    createdBy="testuser"
                    creationDate="20050812T135800">
    <name>ApprovePlannedItem</name>
    <process-display-name>
      <localized-text>
        <locale language="en">Approuver l'élément planifié</locale>
      </localized-text>
    </process-display-name>
    <description>
      <localized-text>
        <locale language="fr">Ce processus de flux de travaux peut être
        adopté pour approuver un élément planifié.</locale>
      </localized-text>
    </description>
    <documentation>Reportez-vous à la documentation validée et planifiée des éléments.
                   documentation sur les éléments.
    </documentation>
    <web-service expose="true">
      <callback-service>wsconnector.ApprovePlannedItem
      </callback-service>
    </web-service>
    <failure-allocation-strategy>
      <allocation-strategy type="target"
       identifier="FAILUREALLOCATIONSTRATEGY" />
    </failure-allocation-strategy>

  ...

  </workflow-process>
```

workflow-process
Il s'agit de la balise parente de toutes les métadonnées de définition de processus.

id Il s'agit d'un identificateur à 64-bit fourni par le serveur de clés Cúram lorsqu'un processus est créé dans l'outil de définition de processus. L'identificateur de processus est requis et doit être unique dans le système de flux de travaux Cúram. La raison en est que l'identificateur de processus en conjonction avec le numéro de version de processus est de savoir comment le moteur du flux de travaux distingue un dossier de définition de processus d'un autre aux fins de lecture de bases de données.

Version de processus
Ce nombre représente la version d'une définition de processus de flux de travaux. Un enregistrement de définition de processus de flux de travaux est identifiée de manière unique par son numéro identificateur et version. Une définition de processus peut avoir plusieurs versions publiées et une version qui est en cours d'édition. Une fois qu'une définition de processus a été publiée, une nouvelle version est créée et ne peut plus être mise à jour. Toutes les mises à jour ultérieures nécessiteront la création d'une nouvelle version et cette dernière ne sera activée qu'après avoir été publiée. Lorsqu'un processus est lancé, le numéro de version publiée la plus élevée est utilisée. Les instances de processus qui commencent par un numéro de version donné reste liées à cette version jusqu'à la fin.

La version de langue
Les métadonnées de définition de processus est le langage de flux de travaux Cúram. Cette langue peut changer au fur et à mesure que de nouvelles fonctionnalités et améliorations sont ajoutées. Ce numéro de version va permettre soit au moteur de flux de travaux d'exécuter des anciennes versions de langue différentes des plus récentes ou plus probablement aux outils de mise à niveau de convertir les anciennes définitions de processus de nouvelles versions de langue.

publié Il s'agit d'un indicateur booléen indiquant si la définition de processus a été publiée ou non. Seules les définitions de processus ayant été publiées peuvent être adoptées ou sélectionnées en tant que sous-processus d'une activité d'un flux secondaire (voir: «Flux secondaire», à la page 89).

category
Une définition de processus doit être placée dans une catégorie. La catégorie doit être sélectionnée dans l'outil de définition de processus et est extraite de la table de codes `ProcessCategory`. Cet attribut est destiné à être utilisé pour la fonctionnalité de recherche de la définition du processus et n'a aucun effet fonctionnel sur le processus dans le moteur de flux de travaux.

createdBy
Cela représente le nom de l'utilisateur qui a créé la définition de processus de flux de travaux. Cet attribut est destiné à être utilisé pour la fonctionnalité de recherche de la définition du processus et n'a aucun effet fonctionnel sur le processus dans le moteur de flux de travaux.

creationDate
Ceci représente la date et l'heure auxquelles la définition de processus de flux de travaux a été créée. Cet attribut est destiné à être utilisé pour la fonctionnalité de recherche de la définition du processus et n'a aucun effet fonctionnel sur le processus dans le moteur de flux de travaux.

process-display-name
Il s'agit du nom d'affichage de la définition de processus, c'est-à-dire le nom du processus que l'utilisateur voit dans l'outil PDT. Il est présenté dans l'environnement local de l'utilisateur. Le nom d'affichage du processus peut être localisé et édité dans l'écran de localisation.

name Il s'agit de l'identificateur technique de la définition de processus. Il permet d'identifier le processus en vue de son adoption. Le service d'adoption (de l'API utilisée pour lancer un processus dans le code) identifie le processus à adopter par son nom. En tant que tel, ce nom est requis pour être unique dans le système de flux de travaux et ne peut pas être modifié une fois le processus créé. Etant donné que le nom du processus est effectivement une constante, il n'est pas localisable comme pourrait l' être un nom d'activité.

description
Un processus peut également avoir une description facultative qui indique brièvement ce qu'il (le processus) fait en faveur de ceux qui éditent la définition de processus à l'avenir. Il s'agit de la zone de texte localisable dans le même format que toutes les zones localisables dans une définition de processus (voir: «Texte localisé», à la page 33).

documentation

Un processus peut également comporter un lien vers des documents qui peuvent l'expliquer d'une manière plus descriptive. Il s'agit d'une zone de texte à format libre dans laquelle le développeur peut entrer le nom du document concerné pour le processus de flux de travaux ou même un lien vers ledit document.

service Web

Cet élément facultatif décrit les détails du service Web d'un processus de flux de travaux. Un processus peut être marqué comme étant un service Web définissant cette valeur de métadonnées qui indique que le processus doit être exposé en tant que service Web. Cela permet au processus de participer à un processus orchestré par BPEL (Business Process Execution Language) et signifie qu'il est possible d'appeler le processus à partir d'un processus BPEL. De plus amples informations sur cette fonctionnalité sont disponibles sur «Services Web de flux de travaux», à la page 124.

exposer

Cet attribut représente un indicateur booléen indiquant si la définition de processus doit être exposée en tant que service Web ou non. Une définition de processus de flux de travaux n'est pas exposée en tant que service Web par défaut.

service de rappel

Il s'agit d'un élément facultatif car ce ne sont pas tous les appels provenant d'un processus BPEL qui nécessitent un rappel. La valeur est le nom qualifié complet d'une classe qui agrandit la classe `org.apache.axis.client.Service` (qui fait partie du Service (Axis API) du projet Apache Axis). La classe `org.apache.axis.client.Service` est générée par la fonction du connecteur de services Web Cúram pour les services Web sortants.

Echec d'allocation-stratégie

Un processus peut éventuellement disposer d'une stratégie de non-attribution spécifiée pour lui. Lors de l'allocation d'une tâche (associée à une activité «Manuel», à la page 61 ou «Décision», à la page 80), le moteur de flux de travaux sollicite la stratégie d'allocation associée pour l'extraction de la liste des cibles d'allocation. Si aucune cible d'allocation n'est retournée suite à cet appel, le moteur de flux de travaux vérifiera alors si une stratégie d'allocation d'échec est disponible et utilisera cette stratégie pour tenter d'allouer la tâche. Etant donné que la stratégie d'allocation de type *TARGET* spécifie directement une cible d'allocation, il n'est jamais nécessaire de revenir à la stratégie d'allocation d'échec. La stratégie d'allocation d'échec est une stratégie à l'échelle de processus. Si cette stratégie est spécifiée, elle sera utilisée pour tous les manuels et les activités de décision dans le processus lorsque nécessaire.

allocation-strategy

Cette section décrit la stratégie d'allocation d'échec en cours d'utilisation pour le processus. La stratégie d'allocation d'échec doit être de type *TARGET*. Si le programme de résolution de travail n'est pas en mesure d' allouer la tâche à un utilisateur, un objet organisationnel (ex. : unité organisationnelle, position ou travail) ou une file d'attente de travaux à l'aide de la cible d'allocation spécifiée, la tâche sera affectée à la file d'attente de travaux par défaut. L'attribut identificateur représente l'identificateur de la cible d'allocation utilisé comme stratégie d'allocation d'échec.

Validations

- Un processus de flux de travaux doit avoir un nom de processus unique. Cela signifie qu'un processus ne peut être créé si le nom du processus est vide ou si un processus portant le même nom existe déjà.
- Un processus de flux de travaux doit avoir un nom d'affichage de processus en anglais (c'est-à-dire dans l'environnement local "en"). Un nom d'affichage dans l'environnement local de l'utilisateur peut aussi être spécifié, en option.
- Un processus de flux de travaux est requis pour spécifier une catégorie.
- Une version publiée du processus de flux de travaux ne peut être supprimée une fois adoptée. Cette opération est nécessaire car même si une version plus récente d'un processus existe, l'exécution en cours des instances de processus lorsque la nouvelle version devient disponible se poursuit à terme avec la version avec laquelle ces instances ont été lancées. Les définitions de processus sont également un enregistrement historique nécessaire qui sert à créer des informations d'audit.
- Une version publiée du processus de flux de travaux ne peut être supprimée si elle est référencée par une activité de flux secondaire dans une version publiée d'un autre processus dans le cas où cette version publiée est la toute dernière version publiée.
- Si une stratégie d'allocation d'échec a été spécifiée pour le processus de flux de travaux, elle doit être de type *TARGET*.
- Le nom de la classe de service de rappel ne peut pas être spécifié si le processus de flux de travaux n'a pas été exposé en tant que service Web.
- Le nom de la classe de service de rappel doit représenter une classe disponible dans le chemin d'accès aux classes d'applications.
- Le nom de la classe de service de rappel doit représenter une classe qui agrandit la classe`org.apache.axis.client.Service`.

Description des WDO (objet de données de flux de travaux) de contexte

Certaines informations génériques d'exécution du système relatives au moteur de flux de travaux doivent être mises à la disposition des activités et des transitions au cours de la durée de vie d'une instance de processus. Les détails sur l'objet de données de flux de travaux `Context_RuntimeInformation` qui permettent d'avoir ces informations peuvent être consultés aux emplacements suivants: «Liste des objets de données de flux de travaux contextuels», à la page 23.

Objets de données de flux de travaux

Présentation

Les données sont gérées et transmises dans le moteur de flux de travaux, sous forme d'objets de données de flux de travaux et d'objets de données de flux de travaux de liste. Il s'agit d'objets logiques définis dans la définition de processus, qui portent un nom et sont associés à une liste d'attributs de différents types auxquels les données peuvent être affectées. Ces objets sont dans leur concept, similaires aux objets des langages de programmation bien que leur apparition dans le système de flux de travaux soit complètement différente. Des valeurs d'objet de données de flux de travaux peuvent être écrites au moment du lancement du processus ou à partir de la sortie de différents types d'activités.

Les instances d'objet de données de flux de travaux et les instances d'objet de données de flux de travaux de type liste interviennent dès que le processus est adopté et tant que le processus n'est pas terminé. Elles peuvent être utilisées comme telles dans les activités et les transitions tout au long de la durée de vie de cette instance de processus. Il est donc de la responsabilité du concepteur de processus de s'assurer que les attributs des objets de données de flux de travaux sont remplis avant de pouvoir être utilisés. Les tentatives d'utilisation des attributs d'objet de données de flux de travaux avant qu'ils ne soient remplis provoquent des incidents au cours de l'exécution.

Métadonnées

```
<workflow-process id="32456" ..... >
    <name>CreateManualTask</name>
    .....
    </description>
    <enactment-mappings>
      ......
    </enactment-mappings>
    <wdos>
      <wdo is-list-wdo="false" initialize-attributes="true">
        <wdo-name>TaskCreateDetails</wdo-name>
        <display-name>
          <localized-text>
            <locale language="fr">Détails de création de la tâche</locale>
          </localized-text>
        </display-name>
        <description>
          <localized-text>
            <locale language="fr">Cet objet de données de flux de travaux
            contient les attributs nécessaires à la
            création manuelle d'une tâche.</locale>
          </localized-text>
        </description>
        <attributes>
          <attribute>
            <attribute-name>subject</attribute-name>
            <display-name>
              <localized-text>
                <locale language="fr">Objet de la tâche</locale>
              </localized-text>
            </display-name>
            <type>STRING</type>
            <required-at-enactment>true</required-at-enactment>
            <process-output>true</process-output>
          </attribute>
          <attribute>
            <attribute-name>participantRoleID</attribute-name>
            <display-name>
              <localized-text>
                <locale language="fr">ID rôle du participant</locale>
              </localized-text>
            </display-name>
            <type>INT64</type>
            <required-at-enactment>true</required-at-enactment>
            <process-output>true</process-output>
          </attribute>
          <attribute>
          <attribute-name>deadlineDateTime</attribute-name>
          <display-name>
            <localized-text>
              <locale language="fr">Date d'échéance</locale>
            </localized-text>
          </display-name>
          <type>DATETIME</type>
          <required-at-enactment>true</required-at-enactment>
          <process-output>false</process-output>
        </attribute>
        <attribute>
          <attribute-name>deadlineDuration</attribute-name>
          <display-name>
            <localized-text>
              <locale language="fr">Durée de l'échéance</locale>
            </localized-text>
          </display-name>
          <type>INT32</type>
          <required-at-enactment>false</required-at-enactment>
          <process-output>false</process-output>
          <initial-value>300</initial-value>
        </attribute>
        <attribute>
          <attribute-name>priority</attribute-name>
```

wdos Facultatif, car une définition de processus de flux de travaux ne doit pas contenir d'objet de données de flux de travaux. Contient les détails de tous les objets de données de flux de travaux définis pour la définition de processus de flux de travaux.

wdo Contient les détails de l'un des objets de données de flux de travaux défini pour la définition de processus de flux de travaux. Sont inclus les détails génériques de l'objet de données de flux de travaux, ainsi que les détails de chacun de ses attributs. Les métadonnées qui décrivent un objet de données de flux de travaux et ses attributs sont décrites ci-après.

is-list-wdo
Contient une valeur booléenne qui indique si l'objet de données de flux de travaux spécifié est de type liste ou non. Lorsque cette option est définie sur `true`, l'objet de données de flux de travaux spécifié joue le rôle de liste et peut ainsi être utilisé pour générer des listes de données disponibles tout au long du flux de travaux.

initialize-attributes
Contient une valeur booléenne qui indique si les attributs associés à l'objet de données de flux de travaux devraient être initialisés lors de la première utilisation de l'objet de données de flux de travaux. Les valeurs par défaut utilisées sont identiques à celles qui seraient définies dans un struct Cúram.

wdo-name
Contient le nom de l'objet de données de flux de travaux.

display-name
Contient le nom d'affichage de l'objet de données de flux de travaux. Ce nom représente une brève description de l'objet de données de flux de travaux et s'affiche dans l'outil de définition de processus. Il s'agit d'une chaîne localisable qui ne contient aucun paramètre. Pour plus de détails sur le texte localisé et les métadonnées associées, voir «Texte localisé», à la page 33.

description
Contient une description plus détaillée de l'objet de données de flux de travaux. Il s'agit également d'une chaîne localisable sans aucun paramètre. Pour plus de détails sur le texte localisé et les métadonnées associées, voir «Texte localisé», à la page 33.

attributes
Contient les détails de tous les attributs associés à l'objet de données de flux de travaux.

attribute
Contient les détails de l'un des attributs associés à l'objet de données de flux de travaux. Les métadonnées décrites ci-après constituent un attribut d'objet de données de flux de travaux :

nom-attribut
Contient le nom de l'attribut d'objet de données de flux de travaux.

display-name
Contient le nom d'affichage de l'attribut d'objet de données de flux de travaux. Ce nom représente une brève description de l'attribut d'objet de données de flux de travaux. Il s'agit d'une chaîne localisable qui ne contient aucun paramètre. Pour plus de détails sur le texte localisé et les métadonnées associées, voir «Texte localisé», à la page 33.

type Chaque attribut d'objet de données de flux de travaux défini doit spécifier un type qui doit être un domaine de base Cúram valide. Lors de la création d'un attribut d'objet de données de flux de travaux dans l'outil de définition de processus, ce type est sélectionné dans la table de codes `DomainType`. Il est conseillé de consulter cette table de codes afin d'obtenir la liste complète des types d'attribut d'objet de données de flux de travaux disponibles. Le type d'un attribut d'objet de données de flux de travaux est utilisé pour garantir que les mappages de données contenus dans un processus de flux de travaux sont compatibles et qu'ils ne provoqueront pas d'incidents au cours de l'exécution. Par exemple, si une zone de paramètre de méthode d'objet de processus métier est de type STRING, l'attribut d'objet de données de flux de travaux utilisé pour mapper les données dans cette zone doit également être de type STRING.

required-at-enactment
Les mappages d'adoption représentent le volume minimal de données requis par le flux de travaux pour être adopté. Ils doivent contenir une entrée pour chaque attribut d'objet de données de flux de travaux dont l'indicateur d'adoption est défini sur `true`. A l'inverse, si cet indicateur est défini sur `false` (par défaut), cet attribut d'objet de données de flux de travaux n'est pas requis pour l'adoption du processus associé. L'outil de définition de processus permet de créer ces mappages d'adoption en examinant chaque attribut d'objet de données de flux de travaux qui a été défini et en créant un mappage pour chaque attribut dont l'indicateur d'adoption est défini sur `true`. Lorsqu'une définition de processus de flux de travaux publiée a été sélectionnée en tant que processus de flux secondaire dans une activité de flux secondaire (voir «Flux secondaire», à la page 89), tous les objets de données de flux de travaux qui ont été marqués comme requis pour l'adoption dans le processus de flux secondaire doivent être mappés pour que la définition de processus parente puisse être publiée.

process-output
Un processus de flux de travaux peut être marqué comme service Web en définissant une valeur de métadonnées qui indique que le processus devrait être exposé en tant que service Web. Ceci permet aux processus de participer à un processus orchestré BPEL (Business Process Execution Language) et implique que le processus peut être appelé à partir d'un processus BPEL de manière synchrone ou asynchrone. Il peut également être nécessaire de mapper les données provenant d'un processus de flux de travaux dans le processus BPEL qui l'a appelé. Lorsqu'il est défini sur `true`, cet élément facultatif indique que les données provenant de cet attribut d'objet de données de flux de travaux devraient être retransmises au processus BPEL appelant une fois le processus de flux de travaux Cúram terminé. La valeur par défaut pour cet élément est `false`.

constant-value
Cet élément facultatif indique si l'attribut d'objet de données de flux de travaux représente ou non une valeur constante. A différents endroits d'une définition de processus de flux de travaux, les attributs d'objet de données de flux de travaux sont utilisés dans les mappages d'entrée (c'est-à-dire dans les mappages

de fonction d'allocation, les mappages de fonction d'échéance, etc.). Dans certains cas, une autorisation d'utilisation des constantes dans certains de ces mappages est obligatoire. En fournissant une valeur constante, les attributs d'objet de données de flux de travaux de ce type peuvent être utilisés dans ce but. L'indicateur "requis pour l'adoption" d'un attribut d'objet de données de flux de travaux ne peut pas être défini sur `true` et contenir une valeur constante. Les données transmises en tant que données d'adoption sont considérée comme des données dynamiques susceptibles d'être modifiées. Les données spécifiées dans un attribut d'objet de données de flux de travaux de type constante ne conviennent pas dans ce cas car leur valeur est déjà connue.

initial-value
Cet élément indique si l'attribut d'objet de données de flux de travaux possède ou non une valeur initiale. Cette fonction peut s'avérer utile dans les situations où un attribut d'objet de données de flux de travaux est utilisé dans le flux de travaux avant d'être rempli par une activité automatique ou autre (c'est-à-dire pour éviter d'avoir à utiliser une activité automatique pour remplir les attributs d'objet de données de flux de travaux uniquement pour vérifier que ces attributs ne sont pas nuls lorsqu'ils sont utilisés dans le cadre des conditions de transition plus tard dans le flux de travaux). Lorsque cet élément a été renseigné, l'attribut d'objet de données de flux de travaux est initialisé sur la valeur spécifiée lors de sa première utilisation. La valeur initiale d'un attribut d'objet de données de flux de travaux peut être remplacée ultérieurement par les différents mappages de sortie qui existent dans un processus de flux de travaux. Un attribut d'objet de données de flux de travaux ne peut pas posséder à la fois une valeur constante et une valeur initiale spécifiées.

Validations

- Un processus de flux de travaux doit contenir un seul objet de données de flux de travaux `Context_RuntimeInformation`.
- Un nom d'objet de données de flux de travaux doit être unique au sein de la définition de processus de flux de travaux qui le contient.
- Le nom de l'objet de données de flux de travaux doit être un identificateur Java™ valide.
- Un nom d'objet de données de flux de travaux défini par l'utilisateur ne peut pas contenir le préfixe `Context_` car il s'agit d'un préfixe réservé dans le système de flux de travaux Cúram.
- Chaque objet de données de flux de travaux spécifié dans la définition de processus de flux de travaux doit contenir au moins un attribut associé.
- Le nom de l'attribut d'objet de données de flux de travaux doit être un identificateur Java valide.
- Vous ne pouvez pas créer un attribut d'objet de données de flux de travaux portant le nom `"value"`. Il s'agit d'un nom d'attribut réservé dans le système de flux de travaux Cúram.
- Le type d'un attribut d'objet de données de flux de travaux doit être un domaine de base Cúram valide et doit se trouver dans la table de codes `DomainType`.
- Un attribut d'objet de données de flux de travaux ne peut pas être marqué à la fois comme "requis pour l'adoption" et comme une valeur constante.

- Un attribut d'objet de données de flux de travaux ne peut pas posséder à la fois une valeur constante et une valeur initiale spécifiées.
- Si un attribut d'objet de données de flux de travaux a été marqué comme une constante, une valeur constante doit être fournie. A l'inverse, si l'attribut n'a pas été marqué comme une constante, aucune valeur constante ne doit être spécifiée.
- Si l'attribut d'objet de données de flux de travaux a été marqué comme une constante, seule une valeur vide peut être spécifiée pour cet attribut si le type de l'attribut est STRING.
- Si l'attribut d'objet de données de flux de travaux a été spécifié avec une valeur initiale, seule une valeur initiale vide peut être spécifiée pour cet attribut si le type de l'attribut est STRING.
- Si l'attribut d'objet de données de flux de travaux a été marqué comme une constante, la valeur spécifiée pour cette constante doit être compatible avec le type de l'attribut associé.
- Si l'attribut d'objet de données de flux de travaux a été spécifié avec une valeur initiale, la valeur spécifiée pour cette valeur initiale doit être compatible avec le type de l'attribut associé.
- L'indicateur de sortie de processus peut uniquement être défini sur true pour un attribut d'objet de données de flux de travaux spécifié si le processus de flux de travaux associé a été exposé en tant que service Web.

Liste des objets de données de flux de travaux contextuels

Les objets de données de flux de travaux contextuels sont des objets qui ne sont pas définis de manière explicite dans les métadonnées de la définition de processus de flux de travaux, mais qui sont mis à disposition par l'outil de définition de processus et le moteur de flux de travaux à différents emplacements au cours de l'exécution d'un processus. Les informations ci-après décrivent brièvement ces objets de données de flux de travaux contextuels et fournissent des liens vers des informations supplémentaires sur ces objets.

Objet de données de flux de travaux Context_RuntimeInformation

L'objet de données de flux de travaux Context_RuntimeInformation est un objet de données de flux de travaux mis à disposition et conservé par le moteur de flux de travaux. Il contient des informations qui s'avèrent pertinentes tout au long du cycle de vie d'une instance de processus de flux de travaux ; les attributs disponibles reflètent ces informations. Ces attributs sont les suivants :

- `processInstanceID` : identificateur généré par le système de l'instance de processus (extrait du serveur de clés Cúram à l'aide du jeu de clés de flux de travaux).
- `enactingUser` : nom de l'utilisateur dont les actions dans l'application ont entraîné l'adoption du processus de flux de travaux.
- `enactmentTime` : date et heure auxquelles le processus a été adopté.

Objet de données de flux Context_Result

Une transition provenant d'une activité automatique devrait pouvoir utiliser la valeur de retour de la méthode appelée directement dans sa condition sans avoir besoin des mappages aux attributs d'objet de données de flux de travaux. Toutefois, en raison du modèle transactionnel du moteur de flux de travaux, ces données doivent être conservées hors de la transaction de l'appel de la méthode d'objet de processus métier. Pour ce faire, une définition d'objet de données de flux de travaux est créée au moment de l'exécution si la valeur de retour est utilisée dans les conditions de la transition sortante. Ces définitions de valeur de retour ne doivent

jamais être conservées car elles induites chaque fois qu'elles sont nécessaires dans le moteur de flux de travaux. Les données de l'objet de données de flux de travaux réelles sont conservées jusqu'à ce que les transitions provenant des instances d'activité en question soient évaluées. Elles sont ensuite supprimées. Pour plus de détails sur l'objet de données de flux de travaux Context_Result, voir «Description des WDO (objet de données de flux de travaux) de contexte», à la page 47

Objet de données de flux de travaux Context_Event
L'objet de données de flux de travaux Context_Event peut être utilisé dans les conditions d'une fonction ou d'un élément de données (voir «Conditions», à la page 112) pour une transition provenant d'une activité contenant une attente d'événement. Il rend certaines informations disponibles (par exemple, la classe et le type de l'événement émis, l'heure à laquelle l'événement a été émis, etc.) contenus dans l'événement émis pour terminer cette instance d'activité. Ces informations peuvent ensuite être utilisées pour modéliser le chemin partant de cette activité spécifiée. Pour plus de détails sur l'objet de données de flux de travaux Context_Event, voir «Description des WDO (objet de données de flux de travaux) de contexte», à la page 58.

Objet de données de flux de travaux Context_Decision
L'objet de données de flux de travaux Context_Decision peut être utilisé dans la condition d'une fonction ou d'un élément de données (voir «Conditions», à la page 112) pour une transition provenant d'une activité de décision. Les attributs disponibles dépendent du format de réponse défini pour l'activité de décision. Pour plus de détails sur l'objet de données de flux de travaux Context_Decision, voir «Description des WDO (objet de données de flux de travaux) de contexte», à la page 88

Objet de données de flux de travaux Context_Task
L'objet de données de flux de travaux Context_Task peut être utilisé dans différents mappages associés à une tâche d'activité manuelle (par exemple, les mappages de saisie de la fonction d'allocation (Allocation Function Input), les mappages de saisie de la fonction d'échéance (Deadline Function Input) et les paramètres de liaison des actions d'activité manuelle (Manual Activity Action Link)). Cet objet de données de flux de travaux contextuel rend disponible l'identificateur de la tâche créée suite à l'exécution de l'activité la contenant. Pour plus de détails sur l'objet de données de flux de travaux Context_Task, voir «Description des WDO (objet de données de flux de travaux) de contexte», à la page 68.

Objet de données de flux de travaux Context_Loop
L'objet de données de flux de travaux Context_Loop peut être utilisé lors de la création de la condition de fin de boucle associée à une activité de début de boucle. Il est également disponible pour créer des conditions de transition sortante pour n'importe quelle activité au sein d'une boucle, et pour spécifier des mappages d'entrée, des paramètres de texte et des paramètres de liaison d'actions pour certaines activités et fonctions contenues dans une boucle. Cet objet de données de flux de travaux contextuel rend disponible le nombre d'itérations d'une boucle pour de tels mappages. Pour plus de détails sur l'objet de données de flux de travaux Context_Loop, voir «Description des WDO (objet de données de flux de travaux) de contexte», à la page 96.

Objet de données de flux de travaux Context_Deadline
L'objet de données de flux de travaux Context_Deadline peut être utilisé pour créer une condition de fonction ou d'élément de données (voir

«Conditions», à la page 112) pour une transition provenant d'une activité qui possède une attente d'événement avec une date d'échéance spécifiée. Il est disponible pour permettre à un développeur de modéliser différents chemins d'exécution à partir d'une activité contenant une date d'échéance, en fonction du fait que cette date soit arrivée à expiration ou non. Pour plus de détails sur l'objet de données de flux de travaux Context_Deadline, voir «Description des WDO (objet de données de flux de travaux) de contexte», à la page 57.

Objet de données de flux de travaux Context_Parallel
L'objet de données de flux de travaux Context_Parallel peut être utilisé dans les différents mappages associés à une activité manuelle parallèle (par exemple, les paramètres de texte d'une opération de tâche et d'un objet de tâche, les mappages d'une stratégie d'allocation, etc.) et à une activité de décision parallèle (par exemple, les paramètres de texte d'une opération de décision, les paramètres de texte d'une opération secondaire, les paramètres de texte qu'une question, etc.). Il rend disponible l'index de l'élément à partir de l'objet de données de flux de travaux Liste des activités parallèles (Parallel Activity List), utilisé pour créer l'instance spécifiée de l'activité encapsulée. Pour plus de détails sur l'objet de données de flux de travaux Context_Parallel, voir «Description des WDO (objet de données de flux de travaux) de contexte», à la page 101.

Objet de données de flux Context_Error
L'objet de données de flux de travaux Context_Error peut être utilisé dans la condition d'une fonction ou d'un élément de données (voir «Conditions» , à la page 112) pour une transition provenant d'une activité automatique. Il permet à un développeur de processus de modéliser un chemin d'exception en dehors d'une activité automatique, c'est-à-dire une transition qui est suivie si l'activité automatique échoue en raison d'une exception non gérée. Pour plus de détails sur l'objet de données de flux de travaux Context_Error, voir «Description des WDO (objet de données de flux de travaux) de contexte», à la page 47

Informations d'exécution

Les instances d'objets de données de flux de travaux et d'objets de données de flux de travaux de type liste interviennent dès qu'un processus de flux de travaux est adopté et tant que le processus n'est pas terminé. Ces instances d'objets de données de flux de travaux peuvent ainsi être utilisées dans les activités (transmission de données à une méthode BPO par exemple) et dans les transitions (mise à disposition de données lors de l'évaluation des conditions de transition par exemple) tout au long de la durée de vie de cette instance de processus.

L'attribut `enactingUser` de l'objet de données de flux de travaux `Context_RuntimeInformation` est défini sur le nom de l'utilisateur dont les actions dans l'application ont provoqué l'adoption du processus de flux de travaux. La même valeur affectée à la transaction lorsqu'une méthode BPO est appelée par la suite dans l'instance du processus de flux de travaux n'est *pas* générée. Ceci est dû à la démarcation de la transaction dans le moteur de flux de travaux lorsque les activités automatiques (à savoir les méthodes BPO) sont appelées dans le serveur d'applications. En raison de la nature asynchrone de cet appel et de l'obligation de vérifier que l'appel du code d'application se trouve dans sa propre transaction, la méthode BPO est appelée par le moteur de flux de travaux (utilisateur SYSTEM) plutôt que par l'utilisateur qui a adopté le processus de flux de travaux en premier lieu. D'ailleurs, dans un contexte métier, la personne qui a adopté le flux de travaux peut même ne pas savoir que cette méthode BPO a été appelée.

De même, notez que l'utilisateur à l'origine de l'adoption d'une instance de processus de flux de travaux n'est transmis à aucune instance de processus de flux secondaire pouvant être appelée par le processus parent. Si l'utilisateur à l'origine de l'adoption de l'instance de processus parent est requis dans l'une des instances de processus de flux secondaire, il doit être transmis de manière explicite à l'aide d'un attribut d'objet de données de flux de travaux dans les mappages d'entrée de ce processus de flux secondaire.

Soyez attentif lors de la mise à jour des données de l'instance de l'attribut d'objet de données de flux de travaux lorsque vous exécutez des activités automatiques parallèles dans une instance de processus de flux de travaux. Si de telles activités automatiques appellent la même méthode BPO et que cette méthode tente de mettre à jour les données pour le même attribut d'objet de données de flux de travaux, une situation d'interblocage des enregistrements de base de données est susceptible de se produire. Le concepteur de processus de flux de travaux doit corriger de telles situations qui se produisent lors de la conception d'une définition de processus de flux de travaux afin de s'assurer que les activités automatiques exécutées en parallèle ne mettent pas à jour le même attribut d'objet de données de flux de travaux.

Adoption d'un processus

Présentation

Une définition de processus définit la structure d'un processus métier et pour lancer l'exécution du travail défini dans cette définition de processus d'une instance du processus doit être créé. Le démarrage d'une instance de processus est appelé *processus de lancement*. La plupart des définitions de processus requièrent un ensemble minimal de données initiales qui sont utilisées principalement pour identifier les objets métier spécifiques sur lesquels l'instance de processus va s'exécuter. Tous les mécanismes de lancement doivent disposer de données d'entrées pour déclencher un processus donné. Ces données d'entrée sont connues en tant que *enactment data* d'un processus.

Actuellement, il existe quatre mécanismes pris en charge par le flux de travaux Cúram :

- Lancement à partir d'un code
- Lancement à partir d'un événement
- Lancement en tant que flux secondaire
- Lancement via un service Web

Les deux premiers mécanismes sont décrits dans ce chapitre. Le mécanisme de lancement du flux secondaire est décrit dans «Flux secondaire», à la page 89. Le mécanisme de lancement de service web est décrit dans «Services Web de flux de travaux», à la page 124.

Code enactment (enactment service API)

La manière la plus directe de lancer un processus consiste à identifier un emplacement dans l'application, à partir duquel une instance de processus sera déclenchée. Le code doit alors être inséré pour appeler l'API de service de lancement. Cette API permet au développeur d'indiquer le nom du processus à démarrer et de fournir les données de lancement requises par le processus.

Si ce mode de lancement de processus est à la fois simple et intuitif, il présente cependant l'inconvénient d'être codé en dur dans la logique de l'application. Ainsi, des modifications comme la suppression du lancement, la modification du processus à démarrer ou même des changements mineurs apportés aux données de lancement exigeront de modifier le code et de redéployer l'application.

Métadonnées

```
<enactment-mappings>
    <mapping>
      <source-attribute
          struct-name="curam.core.sl.struct.TaskCreateDetails"
          name="subject" />
      <target-attribute
          wdo-name="TaskCreateDetails"
          name="subject" />
    </mapping>
    <mapping>
      <source-attribute
          struct-name="curam.core.sl.struct.GroupMemberDetails"
          name="dtls.memberName" />
      <target-attribute
          wdo-name="MemberCreateDetails"
          name="memberName" />
    </mapping>
    <mapping>
      <source-attribute
          struct-name="curam.core.sl.struct.ChildDetailsList"
          name="dtls.identifier" />
      <target-attribute
          wdo-name="ChildDetails"
          name="identifier" />
    </mapping>

    ...

  </enactment-mappings>
```

enactment-mappings
Contient une liste des mappages qui peuvent être utilisés en tant que données initiales en adoptant l'instance de processus associée. Une définition de processus n'est pas nécessaire pour lancer des mappages d'adoption définis.

mapping
Un `mappage` représente un élément de données fournies à partir d'un attribut struct Cúram lors de l'adoption de l'instance de processus associée.

source-attribute
Cela représente un attribut struct Cúram à utiliser dans le remplissage des données d'adoption du processus et est obligatoire dans un mappage d'adoption.

struct-name
Le nom d'une structure Cúram qui contient un attribut requis pour appliquer le processus de flux de travaux. Les structures agrégées et de liste peuvent également être utilisées pour transmettre des données d'adoption dans un processus de flux de travaux, comme indiqué dans le fragment de métadonnées ci-dessus.

name Le nom d'un attribut d'une structure Cúram qui contient un attribut requis pour appliquer le processus de flux de travaux associé. Si la zone d'une structure agrégée ou si une structure de

liste est utilisée, ce nom représente le nom complet de cette zone. Dans un tel cas, le nom est constitué du nom de rôle de l'association entre la structure parent et enfant en plus du nom de zone réel. Cela est illustré dans le fragment de métadonnées ci-dessus.

target-attribute
Cela représente un attribut de l'objet de données de flux de travaux qui doit être rempli avec les données d'adoption du processus et est obligatoire dans un mappage d'adoption.

wdo-name
Le nom d'un objet de données de flux de travaux Cúram contenant l'attribut cible à mapper. (Voir «Objets de données de flux de travaux», à la page 17).

name Le nom d'un attribut d'objet de données de flux de travaux Cúram qui est marqué comme étant requis pour l'adoption. La valeur de l'attribut source de la structure Cúram correspondante sera mappée sur cet attribut lorsque le processus est lancé.

Validations

- L'attribut struct Cúram utilisé comme un attribut source dans un mappage d'adoption doit être valide et être du type correct pour l'attribut d'objet de données de flux de travaux cible associé.
- L'attribut d'objet de données de flux de travaux cible dans un mappage d'adoption doit être valide et doit être marqué comme étant requis pour l'adoption.
- Si l'attribut cible du mappage d'adoption provient d'un objet de données de flux de travaux de liste, alors l'attribut source doit être une zone d'une struct de liste.

Code

```
// Créer la liste que nous allons transmettre au service d'adoption.
    final List enactmentStructs = new ArrayList();

    final TaskCreateDetails taskCreateDetails =
      new TaskCreateDetails();

    taskCreateDetails.subject = "The subject of a Task";
    taskCreateDetails.reservedBy = "someUser";

    enactmentStructs.add(taskCreateDetailsStruct);

    // Une structure agrégée.
    GroupMemberDetails groupMemberDetails
      = GroupMemberDetails nouveau ();

    groupMemberDetails.dtls.memberName = "Test User";

    enactmentStructs.add(groupMemberDetails);

    // Une structure de liste.
    ChildDetailsList childDetailsList
      = ChildDetailsList nouveau ();

    = ChildDetails nouveau ();
    recordOne.identifier = 1;
    childDetailsList.dtls.add(recordOne);

    = ChildDetails nouveau ();
    recordTwo.identifier = 2;
    childDetailsList.dtls.add(recordTwo);

    enactmentStructs.add(childDetailsList);


    EnactmentService.startProcess(
      "TASKCREATEWORKFLOW", enactmentStructs);
```

- L'API `EnactmentService` est fourni pour permettre l'adoption de processus de flux de travaux à partir du code de l'application. La liste des structures Cúram fournie à la méthode `startProcess()` doit suffire pour complètement exécuter les mappages d'adoption du processus associé. Notez qu' adopter une procédure de cette manière est asynchrone et le processus sera déclenché une fois la transaction d'application en cours terminée.
- La méthode `startProcessInV3CompatibilityMode` est fournie uniquement pour l'utilisation de la tâche d'application de l'API. L'utilisation directe de cette méthode dans le code personnalisé n'est pas pris en charge et peut entraver les mises à niveau ultérieures.

Lancement d'événements

Il est possible de démarrer un processus en réponse à un événement émis. Ceci nécessite de paramétrer certaines données de configuration (via une interface d'administration ou en tant qu'entrées préconfigurées de base de données). La configuration indique le(s) processus à démarrer en réponse à un événement spécifique émis. Cette méthode permet aussi de configurer des mappages de données d'événement sur les données de lancement requises par le processus.

La configuration de l'événement d'adoption du processus est stockée dans la base de données et une interface utilisateur est fournie pour permettre la manipulation de ces données. Un lancement de processus créé de cette façon peut donc être

activé, désactivé, modifié et même supprimé lors de l'exécution. Le principal inconvénient de cette approche est qu'elle ne peut être utilisée que pour les définitions de processus qui nécessitent une petite quantité de données de lancement, car les événements n'ont qu'une quantité limitée d'informations.

Un gestionnaire d'événements de lancement de processus est fourni avec Cúram et est automatiquement enregistré pour écouter les événements émis dans l'application. Lorsqu'un processus a été configuré pour être lancé à partir d'un événement, les données de l'événement sont mappées sur les données de lancement du processus et le processus est déclenché.

Données de configuration

L'activation d'un événement pour adopter un processus nécessite la configuration d'une association événement-processus. Chaque événement émis dans l'application vérifie si des processus ont été associés et s'ils doivent être adoptés. La dernière version publiée d'un processus sera toujours adoptée pour un événement associé.

L'enregistrement d'un événement pour déclencher un processus est stocké sous la forme d'un dossier sur la table ProcEnactmentEvt. Le gestionnaire d'événements du processus recherche une représentation en cache de cette table de correspondance des entrées lorsqu'un événement est émis dans l'application et arrête tous les processus correspondants. Le tableau suivant décrit les données requises pour remplir la tableau ProcEnactmentEvt.

Tableau 1. Description du tableau ProcEnactmentEvt

Nom de zone d'entité	Description de la zone
procStartEventID	Identificateur unique de l'association d'événements de processus.
eventClass	La classe d'événement de l'événement qui a été spécifiée pour appliquer le processus de flux de travaux.
eventType	Le type d'événement de l'événement qui a été spécifié pour appliquer le processus de flux de travaux.
processToStart	Si un événement contenant la classe d'événement indiquée et le type décrit ci-dessus est soulevé, la dernière version publiée du processus de flux de travaux spécifié par ce nom sera adopté.
enabled	Cet indicateur booléen qui indique si l'association d'événements-processus est activée. Cela permet à l'application d'un processus de flux de travaux par un événement spécifié d'être activée/désactivée lors de l'exécution.

La table ProcEnactEvtData table stocke les données devant être mappées à partir d'un événement métier dans le flux de trav aux étant adopté lors de l'émission de l'événement spécifié. Le tableau suivant décrit les données requises pour remplir la table ProcEnactEvtData.

Tableau 2. Description of the ProcEnactEvtData Table

Nom de zone d'entité	Description de la zone
procEventMappingID	Identificateur unique du mappage de données d'événement de processus.
procStartEventID	Identificateur unique de l'association d'événements de processus. Cette zone est la clé unique sur la table ProcEnactmentEvt associée et est utilisée pour associer toutes les données requises pour appliquer le processus de flux de travaux lorsqu'un événement spécifié est émis.
eventField	Ceci indique laquelle des trois zones d'un événement sera utilisée pour remplir l'attribut de l'objet de données de flux de travaux. Les valeurs de cette zone sont extraites de la table `EventField` et sont décrites en détail ci-après.
wdoAttribute	Le nom complet d'un attribut d'objet de données de flux à remplir avec les données de la zone d'événement donnée lorsqu'un processus est lancé. Cette table contiendra une entrée pour chaque attribut d'objet de données de flux de travaux qui a été marqué comme Obligatoire pour l'adoption dans le processus en cours adopté par l'événement émis.

Il existe trois champs d'événement pouvant être utilisés en tant que mappages aux fins d'adoption. Ils sont ainsi énumérés dans la table de codes `EventField` et décrits ci-après.

événement de données principal
: Un identificateur unique associé à la classe d'événement à partir de laquelle l'événement est émis. Par exemple, lorsque le type d'objet métier spécifié pour un événement est égal à 'cas d'utilisation', les données d'événement peuvent être l'identificateur d'observations.

Les données d'événements secondaires
: Il peut s'agir de n'importe quelle valeur numérique et est destiné aux événements qui doivent représenter une association entre deux entités.

par l'utilisateur
: Le nom d'utilisateur Cúram de l'utilisateur ayant créé l'événement.

Validations

- Les données disponibles à partir d'un événement doivent être suffisantes pour remplir complètement les données d'adoption de la définition de processus associée.
- Lorsqu'un processus est configuré pour une adoption basée sur les événements, les modifications ultérieures aux données de processus doivent se conformer aux mappages de données d'événement existantes.

- Lorsqu'un processus est configuré pour être lancé à partir d'un événement, il ne peut disposer de sa dernière version mise à jour supprimée si la prochaine version mise à jour la plus récente ne peut entièrement remplir ses données d'adoption à partir de l'événement.

Activité de base

Présentation

Tous les types d'activité pris en charge par le flux de travaux Cúram partagent certaines informations de base. Ces informations leur permet d'être identifié de manière unique par le moteur de flux de travaux et s'affiche à la fois textuellement ou graphiquement et dans l'outil de définition de processus. Chaque activité à un nom et une description facultative qui sont tous les deux sont localisables. Cela permet aux différentes interfaces utilisateur d'administration d'afficher les informations dans l'environnement local approprié.

Ce niveau d'uniformité de base permet d'identifier et d'exécuter les activités par le moteur de flux de travaux sans connaître le type d'activité spécifique. Chaque type d'activité connaît ses propres métadonnées et leur comportement lorsqu'elles sont exécutées. Cet arrangement permettra l'ajout de nouveaux types d'activité, si nécessaire, sans affecter le comportement de base du moteur de flux de travaux.

Métadonnées

```
<automatic-activity id="1" category="AC1">
    <name>
      <localized-text>
        <locale language="fr">ApproveCase</locale>
      </localized-text>
    </name>
    <description>
      <localized-text>
        <locale language="fr">This automatic activity
        sera exécuté pour approuver un cas.</locale>
      </localized-text>
    </description>

    ...

  </automatic-activity>
```

id Il s'agit d'un identificateur à 64 bits fourni par le serveur de clés Cúram lorsque les activités sont créées dans l'outil de définition de processus. L'ID activité doit être unique au sein d'une définition de processus, mais son unicité globale dans toutes les définitions de processus sur le système n'est pas obligatoire.

category
Vous pouvez placer une activité facultativement dans une catégorie. La catégorie doit être sélectionnée dans l'outil de définition de processus. Par ailleurs, il est extrait de la table de codes `ActivityCategory`. Cet attribut est destiné à être utilisé pour rechercher une fonctionnalité basée sur les activités et n'a aucun effet fonctionnel sur l'activité.

name Le nom de l'activité est le moyen par lequel l'activité est identifiée dans le but de l'afficher. Ceci est différent de l'identificateur d'activité qui est utilisé pour identifier l'activité pour exécution par le moteur de flux de travaux.

description
Une activité peut également avoir une description facultative qui indique brièvement ce que l'activité fait pour le bénéfice de ceux qui seront chargés d'éditer la définition de processus dans le futur.

Texte localisé

Comme indiqué dans le fragment XML ci-dessus, le nom de l'activité et la description sont non seulement des zones de texte, mais sont définis comme un élément `localized-text`. Il s'agit de l'élément à usage général utilisé tout au long du processus de définition des métadonnées partout où le texte est requis pour être localisé.

Un élément `localized-text` valide doit contenir au moins un élément enfant `locale`. Sauf dans l'écran de localisation, tout texte localisable entré dans l'outil de définition de processus est sauvegardé dans l'environnement local de l'utilisateur et dans l'environnement local anglais (c'est-à-dire "en").

```
<localized-text>
    <locale language="fr">ApproveCase</locale>
    <locale language="fr" country="CA">ApprouverAffaire</locale>
    <locale language="fr">ApprouverAffaire</locale>
    <locale language="fr" country="CA">ApprouverAffaire</locale>
  </localized-text>
```

locale Ce fichier contient le texte de l'environnement local indiqué par les attributs `langue` et `pays`. Remarque : un environnement local est identifié de manière unique à la fois par son langage et le pays ce qui signifie que *en*, *en_US* et *en_GB* représentent des environnements locaux différents.

langue
Elle est obligatoire et correspond au code de langue ISO à deux lettres.

pays La mention est optionnelle et correspond au code de langue ISO à deux lettres.

Validations

- Le nom de l'activité est obligatoire et doit être unique au sein d'une définition de processus de flux de travaux spécifiés. Toutefois, le nom de l'activité est également une chaîne localisable. Cette validation garantit également qu'un nom d'activité spécifiée est également unique pour chaque environnement local spécifié.
- Une activité doit être l'un des types d'activité autorisée. Dans la pratique, cette règle est satisfaisante puisqu'il n'y a aucun moyen de créer des activités sans sélectionner un type approprié dans l'outil de définition de processus. Même lorsque vous construisez manuellement des définitions de processus dans un éditeur de texte, le type d'activité correspond aux mêmes noms d'éléments de métadonnées permettant de créer un marquage valide qui représente un type d'activité inexistant.

Type d'activité de base

Certains types d'activité : activités de routage, démarrage de processus et fin de processus n'ont d'autres métadonnées que celles communes à tous les types d'activité. Leur comportement est également assez intuitif pour être décrit ici. Tous les autres types d'activité ont des chapitres dédiés.

Activité d'acheminement

Une activité d'acheminement est une activité qui n'effectue aucune fonctionnalité métier. Cela peut être considéré comme une activité nulle puisque son exécution n'affecte ni les données ni l'application ni le processus métier en aucune manière.

Son rôle principal est d'aider à contrôler le flux. Les activités d'acheminement sont souvent utilisées comme points de séparation (fractionnement) et de synchronisation (jonction). Ils sont également utiles lorsque les activités sont requises par le processus métier et ne forment pas naturellement une structure de bloc valide que le moteur de flux de travaux peut exécuter.

Etant donné que tous les types d'activité peuvent avoir des notifications qui leur sont associées (voir : «Notifications d'activité», à la page 102), les activités d'acheminement peuvent être utilisés pour fournir l'effet d'une notification qui n'est pas connectée à une autre fonctionnalité.

Activité du processus début/fin

Les activités de début et de fin de processus marquent le début et la fin d'un processus. Il s'agit de points d'ancrage auxquels d'autres activités peuvent être connectés à l'aide de transitions, créant ainsi une série d'étapes entre le début et la fin de la procédure. Une définition de processus valide qui traverse toutes les transitions entre les activités le démarrage de l'activité de début de processus doit mener vers l'activité de fin de processus (notez qu'il n'est pas nécessaire de traverser tous les chemins dans une instance en cours d'exécution, par exemple, si vous rencontrez un fractionnement (voir «Fractionnement/Jointure», à la page 116) seuls certains des chemins peuvent effectivement être suivis en fonction de l'évaluation des conditions de transition). En tant que tel, la plus simple (et accessoirement la plus inutile) définition de processus est celle qui ne contient que ces deux activités et une transition du début à la fin de l'activité du processus.

Chaque définition de processus doit avoir exactement un début de processus et exactement une activité de fin de processus. Lors de la définition d'un processus à l'aide de l'outil de définition de processus, ces deux activités sont créées automatiquement sur la création de processus et n'ont pas besoin d'être (en fait, ne peuvent pas être) créées explicitement par l'utilisateur.

Les activités de début de processus et de fin de processus forment le bloc extrême d'un bloc de définition de processus valide structuré comme requis par le flux de travaux Cúram.

Automatique

Conditions préalables

- Les détails de base communs à tous les types d'activité pris en charge par le flux de travaux Cúram sont décrits dans «Activité de base», à la page 32 et sont applicables à l'activité automatique ici décrite.

Présentation

Une activité automatique est une étape totalement automatisée d'un processus de flux de travaux. Normalement, aucune intervention humaine n'est nécessaire pour la réalisation de cette activité. Une étape d'activité automatique appelle une méthode de l'application pour effectuer un traitement requis dans le cadre du processus métier global. Les utilisations générales des activités automatiques

incluent : le fait d'effectuer des calculs, la mise à jour des entités et l'extraction des données dans le moteur de flux de travaux.

Les méthodes métier de Cúram

Une partie importante du traitement pour une activité automatique est effectuée dans le code d'application qui est appelé. Les activités automatiques effectuent leur travail en appelant les méthodes métier Cúram (les deux BPO (objet de processus métier) et les méthodes d'entité sont pris en charge). Techniquement, il s'agit de méthodes publiques sur les objets de processus métier et les entités. Une partie essentielle de la définition d'activité automatique est la méthode à appeler et les paramètres à transmettre. Les sections suivantes décrivent ces derniers.

Métadonnées

```
<automatic-activity id="1" category="AC1">
    ...

    <bpo-mapping
      interface-name="curam.sample.facade.intf.SampleBenefit"
      method-name="createAssociatedProductDeliveryForPlannedItem">
      <formal-parameters>

         ...

      </formal-parameters>
    </bpo-mapping>
  </automatic-activity>
```

bpo-mapping
Contient les détails de la méthode métier Cúram qui sera appelée lorsque l'activité automatique associée est exécutée. Ces détails comprennent le nom de l'interface et la méthode associées et également toutes les entrées et des mappages de retour associés à la méthode appelée. Les mappages d'entrée et de sortie sont décrits dans les sections suivantes. Les attributs obligatoires d'un mappage d'attribut d'un BPO (objet de processus métier) sont décrits ci-après.

interface-name
Cela représente le nom complet de l'interface Cúram qui contient la méthode associée à l'activité automatique.

method-name
Représente la méthode qui se trouve sur l'interface Cúram spécifiée qui sera appelée lorsque l'activité est exécutée.

Validations

- Aussi bien l'interface que les noms de méthode doivent être spécifiés pour le mappage de la méthode d'objet de processus métier de l'activité automatique.
- Le nom de l'interface spécifiée doit être une classe valide et cette classe doit exister sur le chemin d'accès aux classes de l'application Cùram.
- Le nom de la méthode doit être un nom de méthode valide et doit exister sur l'interface indiquée.

Code

Comme indiqué précédemment, toute méthode métier Cúram publique valide (BPO ou entité) peut être associée à une activité automatique d'un processus de flux de travaux et peut donc être appelée lorsque cette activité est exécutée. En règle générale, l'échec d'une telle méthode lorsque une activité automatique est

exécutée entraîne l'appel d'une stratégie de gestion des erreurs de flux de travaux. Ainsi, l'activité associée à la méthode en échec peut, par exemple, être réessayée plusieurs fois. De ce fait, les méthodes associées à des activités automatiques ne doivent généralement pas générer d'exceptions. Si la fonction exceptions modélisées est utilisée, lorsqu'une méthode BPO émet une exception et qu'elle a été relancée le nombre de fois requis, toutes les transitions de l'activité automatique contenant le flux de données de l'objet `Context_Error` sont évaluées. Si aucune de ces transactions n'est évaluée à True, leurs chemins sont suivis et de ce fait, le processus de réparation peut survenir automatiquement une fois que la méthode BPO échoue.

Mappages des entrées

Il doit y avoir un moyen de fournir les paramètres requis par une méthode pour l'appeler dans le moteur de flux de travaux. Le moteur de flux de travaux dispose d'un pool de données sous forme d'objets de flux de données (voir «Objets de données de flux de travaux», à la page 17). Les mappages d'entrée sont utilisés pour déclarer les attributs d'objets de données de flux de travaux qui seront utilisés pour renseigner les valeurs des paramètres de méthode spécifique lorsque la méthode est appelée. Les mappages d'entrée sont facultatives là où les champs de structure ont été spécifiés en tant que paramètres de méthode. Cependant, les paramètres de type primitif doivent être mappés.

Métadonnées

Les métadonnées suivantes sont communes à tous les trois types de mappages de paramètre d'entrée (type de base, structure et structures regroupées) et ne seront donc pas décrites.

formal-parameters
Contient la liste des paramètres tels que définis dans la signature de méthode métier de l'activité automatique.

formal-parameter
Contient les détails d'un mappage d'entrée de paramètres formels tel que défini dans la signature de méthode métier associée. Dans cet exemple, une entrée de mappage de paramètre formel existera pour chaque paramètre défini dans la méthode métier associée.

index Représente la position du paramètre formel dans la liste des paramètres formels définis pour la méthode spécifiée. Il s'agit d'un index à base zéro.

Mappages d'entrée des paramètres type de base : Les paramètres de type de base offrent le plus simple type de mappage d'entrée. Dans cet exemple, les mappages d'entrée sont créés pour chaque paramètre formel de type de base contenu dans la méthode métier associée à l'activité automatique. Un paramètre de type de base dans une méthode métier Cúram représente une définition de domaine (voir le *Guide de référence de modélisation Cúram* pour plus de détails sur les définitions de domaine).

```
<automatic-activity id="1" category="AC1">
    ...

    <bpo-mapping
      interface-name="curam.sample.facade.intf.SampleBenefit"
      method-name="createDelivery">
      <formal-parameters>
        <formal-parameter index="0">
          <base-type type="STRING">
            <wdo-attribute wdo-name="SPProductDeliveryPI"
                           name="description"/>
          </base-type>
        </formal-parameter>
        <formal-parameter index="1">
          <base-type type="INT64">
            <wdo-attribute wdo-name="SPProductDeliveryPI"
                           name="plannedItemID"/>
          </base-type>
        </formal-parameter>
      </formal-parameters>
    </bpo-mapping>
  </automatic-activity>
```

base-type

Contient les détails d'un mappage d'entrée de type de base. Un mappage de type de base indique que la zone à laquelle il est mappé est primitive (contrairement à la structure et les mappages de structures imbriqués, comme indiqué ci-après). Un mappage d'entrée de type de base contient l'attribut obligatoire suivant :

type Cette section décrit le type de la zone primitive auquel il est mappé. Pour un mappage d'entrée de type de base, il s'agit de la définition du type de domaine spécifié comme paramètre formel dans la méthode.

attribut wdo

Contient les détails de l'attribut de l'objet de données de flux de travaux (voir «Objets de données de flux de travaux», à la page 17) contenant les données qui seront utilisées pour renseigner le paramètre de type de base associé lorsque la méthode métier est appelée. Les attributs obligatoires sont décrits ci-après.

wdo-name

Il décrit le nom de l'objet de données de flux de travaux utilisé dans le mappage d'entrée.

name Il décrit le nom de l'attribut de l'objet de données de flux de travaux indiqué dans le mappage d'entrée.

Les mappages d'entrée pour les paramètres de structure : Les structures peuvent être spécifiées comme des paramètres aux méthodes d'objets de données de processus métier. Cette section décrit les métadonnées des mappages d'entrée associés à ces paramètres.

```
<automatic-activity id="1" category="AC1">
    ...

    <bpo-mapping
      interface-name="curam.sample.facade.intf.SampleBenefit"
      method-name="createAssociatedProductDeliveryForPlannedItem">
      <formal-parameters>
        <formal-parameter index="0">
          <struct
          type="curam.struct.SampleBenefitPlanItemDetails">
            <field name="description">
              <base-type type="STRING">
                <wdo-attribute wdo-name="SPProductDeliveryPI"
                               name="description"/>
              </base-type>
            </field>
            <field name="plannedItemIDKey">
              <base-type type="INT64">
                <wdo-attribute wdo-name="SPProductDeliveryPI"
                               name="plannedItemID"/>
              </base-type>
            </field>
            <field name="plannedItemName">
        <base-type type="STRING" />
            </field>
          </struct>
        </formal-parameter>
      </formal-parameters>
    </bpo-mapping>
  </automatic-activity>
```

structure
Elle contient les détails d'un mappage d'entrée de structure, y compris le type de structure et les mappages pour chaque zone définie dans cette structure. Un mappage d'entrée type contient l'attribut obligatoire suivant :

type Cette section décrit le type de la structure qui a été spécifié comme paramètre formel dans la méthode. Ceci est représenté comme le nom complet de la structure spécifiée comme paramètre formel.

Zone Elle contient les détails du mappage d'entrée pour l'une des zones définies dans le paramètre de structure. Une zone contient les détails du mappage d'entrée pour le type de base primitif associé à cette zone ainsi que l'attribut obligatoire suivant :

name Cette section décrit le nom de la zone tel qu'il est défini dans la structure spécifiée en tant que paramètre formel.

base-type
Contient les détails d'un mappage d'entrée de type de base pour la zone spécifiée. Un mappage d'entrée de type de base contient l'attribut obligatoire suivant :

type Cette section décrit le type de la zone primitive auquel il est mappé.

attribut wdo
Contient les détails de l'attribut de l'objet de données de flux de travaux (voir «Objets de données de flux de travaux», à la page 17) contenant les données qui seront utilisées pour renseigner la zone de type de base associée lorsque la méthode est appelée. Il ne sera pas présent si l'utilisateur n'a pas spécifié un mappage d'entrée pour le paramètre de cette méthode. Cet élément, lorsqu'il est spécifié, contient les attributs obligatoires suivants :

wdo-name
Il décrit le nom de l'objet de données de flux de travaux utilisé dans le mappage d'entrée.

name Il décrit le nom de l'attribut de l'objet de données de flux de travaux indiqué dans le mappage d'entrée.

Les mappages d'entrée pour les paramètres de structure agrégée : Les structures agrégées (voir le *Guide de référence de la modélisation Cúram* pour en savoir plus sur l'agrégat de structure) peuvent être spécifiées en tant que paramètre aux méthodes métier. Dans ce cas, les métadonnées sont similaires à celles décrites ci-dessus pour les paramètres de structure formelle (voir «Les mappages d'entrée pour les paramètres de structure», à la page 37). Toutefois, il existe une légère différence du fait qu'une zone dans le paramètre de structure peut se résoudre vers une autre structure et non vers un type primitif comme indiqué dans l'exemple des mappages de structure. Dans ce scénario, le nom de zone n'est pas le même que celui de la zone mappée qui est associée au paramètre de structure mais c'est le nom du rôle dans l'association entre la structure spécifiée et la structure qu'il regroupe. Le fragment de métadonnées fournit un exemple d'un tel exemple de mappage d'entrée. Les éléments de métadonnées ont été précédemment décrits plus haut, dans la section des mappages d'entrée.

```
<automatic-activity id="1" category="AC1">
    ...

    <bpo-mapping
      interface-name="curam.sample.facade.intf.SampleBenefit"
      method-name="createBenefit">
      <formal-parameters>
        <formal-parameter index="0">
          <struct type="curam.struct.PlannedItemDetails">
            <field name="description">
              <base-type type="STRING">
                <wdo-attribute wdo-name="SPProductDeliveryPI"
                               name="description"/>
              </base-type>
            </field>
            <field name="plannedItemID">
              <base-type type="INT64">
                <wdo-attribute wdo-name="SPProductDeliveryPI"
                               name="plannedItemID"/>
              </base-type>
            </field>
            <field name="dtls">
              <struct type="curam.struct.PlannedItemKey">
                <field name="subject">
                  <base-type type="STRING">
                    <wdo-attribute wdo-name="SPProductDeliveryPI"
                                   name="subject"/>
                  </base-type>
                </field>
                <field name="concernRoleID">
                  <base-type type="INT64">
                    <wdo-attribute wdo-name="SPProductDeliveryPI"
                                   name="concernRoleID"/>
                  </base-type>
                </field>
              </struct>
            </field>
          </struct>
        </formal-parameter>
      </formal-parameters>
    </bpo-mapping>
  </automatic-activity>
```

Mappages d'entrée des paramètres struct de liste : Les mappages d'entrée des paramètres de la structure de liste peuvent désormais être également spécifiés. Dans cet exemple, les métadonnées sont similaires à celles décrites ci-dessus pour les paramètres formels regroupés (voir «Les mappages d'entrée pour les paramètres de structure agrégée», à la page 39). Le type de la structure spécifiée dans les métadonnées d'un paramètre de structure de liste est le nom de la structure de liste. Le nom de la première zone indique le nom du rôle contenu dans l'association entre la structure de liste spécifiée et la structure enfant qu'il regroupe. En général, cette zone se dissipe dans une autre structure (la structure enfant contenue dans la structure de liste). L'objet de données de flux de travaux indiqué dans un tel mappage représente un objet de données de flux de travaux de liste. Le fragment de métadonnées fournit un exemple d'un tel exemple de mappage d'entrée. Les éléments de métadonnées ont été précédemment décrits plus haut, dans la section des mappages d'entrée.

```
<automatic-activity id="1" category="AC1">
    ...

    <bpo-mapping
      interface-name="curam.sample.facade.intf.SampleBenefit"
      method-name="processClaimantDependents">
      <formal-parameters>
        <formal-parameter index="0">
          <struct type="curam.sample.struct.
                        ClaimantDependentDetailsList">
            <field name="dtls">
              <struct type="curam.sample.struct.
                            ClaimantDependentDetails">
                <field name="identifier">
                  <base-type type="INT64">
                    <wdo-attribute wdo-name="ClaimantDependent"
                    name="identifier"/>
                  </base-type>
                </field>
                <field name="firstName">
                  <base-type type="STRING">
                    <wdo-attribute wdo-name="ClaimantDependent"
                    name="firstName"/>
                  </base-type>
                </field>
                <field name="surname">
                  <base-type type="STRING">
                    <wdo-attribute wdo-name="ClaimantDependent"
                    name="surname"/>
                  </base-type>
                </field>
              </struct>
            </field>
          </struct>
        </formal-parameter>
      </formal-parameters>
    </bpo-mapping>
  </automatic-activity>
```

Mappages d'entrée et éléments indexés à partir d'objets de données de flux de travaux de liste : Pour les activités contenues dans les boucles, un élément provenant d'un objet de données de flux de travaux de liste peut être utilisé dans un mappage d'entrée pour renseigner une zone de paramètre formel. Lorsque ce type d'entrée de mappage est utilisé, à chaque fois que la boucle contenant l'activité y est réitérée, la zone de paramètre formel sera renseignée par la valeur suivante provenant de cet objet de données de flux de travaux de liste. Cela est mis en évidence ici, comme la syntaxe de métadonnées d'un tel mappage est légèrement différente de celle des autres types de mappage d'entrée. Le fragment de métadonnées fournit un exemple de tels mappages d'entrée. Le nom de la liste d'objets de flux de données utilisé pour renseigner la zone du paramètre formel est qualifiée avec la syntaxe `[Context_Loop.loopCount]`. Il permet d'utiliser le moteur de flux de travaux lors de l'exécution pour déterminer l'itération de la boucle qui est en cours d'exécution et l'emplacement de l'élément de l'objet de données de flux de travaux de liste pour récupérer les données permettant de renseigner la zone du paramètre formel.

```
<automatic-activity id="1" category="AC1">
    ...

    <bpo-mapping
      interface-name="curam.sample.facade.intf.SampleBenefit"
      method-name="retrieveClaimantDependentDetails">
      <formal-parameters>
        <formal-parameter index="0">
          <struct type="curam.sample.struct.
                        ClaimantDependentDetails">
            <field name="identifier">
              <base-type type="INT64">
                <wdo-attribute name="identifier"
                wdo-name=
                  "ClaimantDependent[Context_Loop.loopCount]"/>
              </base-type>
            </field>
            <field name="fullName">
              <base-type type="STRING">
                <wdo-attribute name="fullName"
                wdo-name=
                  "ClaimantDependent[Context_Loop.loopCount]"/>
              </base-type>
            </field>
          </struct>
        </formal-parameter>
      </formal-parameters>
    </bpo-mapping>
  </automatic-activity>
```

Validations

- Les attributs de l'objet de données de flux de travaux spécifiés dans les mappages d'entrée doivent être valides. Les critères définissant un attribut d'objet de données de flux de travaux valide dans peuvent être affichés dans «Validations», à la page 22
- Le type de paramètre formel dont le mappage est effectué et le type d'attribut d'objet de données de flux de travaux utilisé dans ce mappage d'entrée doivent être compatibles. Par exemple, si le mappage d'entrée en cours de création est un champ de structure qui a un type STRING, alors l'attribut d'objet de données de flux de travaux utilisé pour ce mappage doit également être de type STRING.
- L'objet de données de flux de travaux `Context_Task` ne peut être utilisé dans un mappage d'entrée si l'activité associée n'est pas une activité manuelle ou de décision.
- L'objet de données de flux de travaux `Context_Loop` ne peut être utilisé dans un mappage d'entrée si l'activité associée n'est pas contenue dans une boucle.
- Un avertissement de validation sera affiché si tous les paramètres de structure définis dans la méthode d'objet de processus métier ne contiennent pas un mappage d'entrée associé.
- Tous les paramètres formels de type de base primitif définis dans la méthode d'objet de processus métier qui doivent contenir un mappage d'entrée associé.
- Si le champ de paramètre formel mappé est un paramètre de type de base, alors un attribut d'objet de données de flux de travaux de liste ne peut pas être utilisé.
- Si le champ de paramètre formel mappé provient d'une structure de liste, alors il doit être mappé à un attribut d'un objet de données de flux de travaux de liste.
- Si l'élément indexé dans un objet de données de flux de travaux de liste (c'est-à-dire `ClaimantDependent[Context_Loop.loopCount]`) est utilisé dans un mappage d'entrée, alors l'objet de données de flux de travaux associé doit être

un objet de données de flux de travaux de liste et l'activité contenant les mappages d'entrée doit se trouver dans une boucle.

Informations d'exécution

Les valeurs des attributs d'objet de données de flux de travaux définies dans les mappages de paramètres d'entrée sont fournies comme données d'entrée à la méthode spécifiée avant qu'elle ne soit appelée lorsque l'activité automatique associée est exécutée.

Mappages de sortie

Les objets de flux de données (voir«Objets de données de flux de travaux», à la page 17) sont des stockages de données du moteur de flux de travaux. Certains des attributs sur les objets de données de flux de travaux spécifiés sont renseignés lorsque le processus est lancé. Il est utile, toutefois, de mettre à jour ou de définir les valeurs des attributs d'objet de données de flux de travaux puisque le processus de flux de travaux est exécuté. Pour prendre en charge cela, certains types d'activité peuvent mapper les données au moteur de flux de travaux. Cela est particulièrement utile pour les activités automatiques, puisque les méthodes métier qu'elles appellent pourraient théoriquement accéder aux données stockées sur n'importe quelle entité dans l'application et les renvoyer pour une utilisation dans des activités supplémentaires dans le processus de flux de travaux. Ces mappages de retour d'une méthode d'objet de processus métier associés à une activité automatique sont facultatifs.

Métadonnées

De la même manière pour les mappages d'entrée (voir «Mappages des entrées», à la page 36), les mappages de sortie sont pris en charge pour les types de retour primitifs, les types de retour de structures imbriquées (regroupées) et les types de retour de structure de liste. Si le type de retour est un type primitif, un retour de mappage d'entrée peut être spécifié. Si le type de retour est une structure, une structure imbriquée ou une structure de liste, des mappages de retour pour une ou plusieurs zones dans la structure spécifiée peuvent être créés. Les fragments de métadonnées suivants fournissent des exemples de ces mappages :

Type de retour primitif :

```
<automatic-activity id="1" category="AC1">
    ...

    <bpo-mapping
      interface-name="curam.sample.facade.intf.SampleBenefit"
      method-name="createAssociatedProductDeliveryForPlannedItem">
      <formal-parameters>
        <formal-parameter index="0">
          ...
        </formal-parameter>
      </formal-parameters>
      <return>
        <base-type>
          <wdo-attribute wdo-name="SPProductDeliveryPI"
                         name="plannedItemID"/>
        </base-type>
      </return>
    </bpo-mapping>
  </automatic-activity>
```

Type de retour de structure :

```
<automatic-activity id="1" category="AC1">
    ...

    <bpo-mapping
      interface-name="curam.sample.facade.intf.SampleBenefit"
      method-name="createAssociatedProductDeliveryForPlannedItem">
      <formal-parameters>
        <formal-parameter index="0">
          ...
        </formal-parameter>
      </formal-parameters>
      <return>
        <struct>
          <field name="description">
            <base-type>
              <wdo-attribute wdo-name="SPProductDeliveryPI"
                             name="description"/>
            </base-type>
          </field>
          <field name="subject">
            <base-type>
              <wdo-attribute wdo-name="SPProductDeliveryPI"
                             name="subject"/>
            </base-type>
          </field>
        </struct>
      </return>
    </bpo-mapping>
  </automatic-activity>
```

Type de retour struct consolidé :

```
<automatic-activity id="1" category="AC1">
    ...
    <bpo-mapping
      interface-name="curam.sample.facade.intf.SampleBenefit"
      method-name="createAssociatedProductDeliveryForPlannedItem">
      <formal-parameters>
        <formal-parameter index="0">
          ...
        </formal-parameter>
      </formal-parameters>
      <return>
        <struct>
          <field name="description">
            <base-type>
              <wdo-attribute wdo-name="SPProductDeliveryPI"
                             name="description"/>
            </base-type>
          </field>
          <field name="subject">
            <base-type>
              <wdo-attribute wdo-name="SPProductDeliveryPI"
                             name="subject"/>
            </base-type>
          </field>
          <field name="dtls">
            <struct>
              <field name="concernRoleID">
                <base-type>
                  <wdo-attribute wdo-name="SPProductDeliveryPI"
                                 name="concernRoleID"/>
                </base-type>
              </field>
              <field name="participantID">
                <base-type>
                  <wdo-attribute wdo-name="SPProductDeliveryPI"
                                 name="participantID"/>
                </base-type>
              </field>
            </struct>
          </field>
        </struct>
      </return>
    </bpo-mapping>
  </automatic-activity>
```

Type de retour de structure de liste :

```
<automatic-activity id="1" category="AC1">
    ...
    <bpo-mapping
      interface-name="curam.sample.facade.intf.SampleBenefit"
      method-name="readClaimantDependentDetails">
      <formal-parameters>
        <formal-parameter index="0">
          ...
        </formal-parameter>
      </formal-parameters>
      <return>
        <struct>
          <field name="dtls">
            <struct>
              <field name="identifier">
                <base-type>
                  <wdo-attribute wdo-name="ClaimantDependent"
                  name="identifier"/>
                </base-type>
              </field>
              <field name="firstName">
                <base-type>
                  <wdo-attribute wdo-name="ClaimantDependent"
                  name="firstName"/>
                </base-type>
              </field>
              <field name="surname">
                <base-type>
                  <wdo-attribute wdo-name="ClaimantDependent"
                  name="surname"/>
                </base-type>
              </field>
            </struct>
          </field>
        </struct>
      </return>
    </bpo-mapping>
  </automatic-activity>
```

return Contient les détails des mappages de sortie spécifiés pour la méthode métier associée à l'activité automatique. Pour un type de retour primitif, une entrée des métadonnées de type de base sera présente, comme indiqué dans l'exemple ci-dessus (voir «Type de retour primitif», à la page 43). Pour une structure, une structure regroupée et des types de retour de structure de liste, la balise des métadonnées de structure est spécifiée et contient des zones dont les types de base sont mappés à l'aide d'attributs d'objets de flux de données.

structure
Contient les détails de mappage de sortie de la structure. Un mappage de sortie de structure contient l'attribut obligatoire suivant.

Zone Il contient les détails du mappage de sortie pour l'une des zones définies dans le type de retour de structure. Une zone contient les détails du mappage de sortie pour le type de base primitif associé à cette zone ainsi que l'attribut obligatoire suivant :

name Il représente le nom de la zone tel qu'il est défini dans la structure spécifiée en tant que type de retour. Pour les types de retour struct non-agrégés, cela ne représente le nom de la zone sur la structure de retour spécifiée qui est en cours de mappage. Pour la structure regroupée et les types de retour de structure de liste, le nom de la

zone représente le nom du rôle contenu dans l'association entre la structure spécifiée et la structure qu'il regroupe.

base-type
Contient les détails d'un mappage de sortie de type de base pour la zone spécifiée ou le type de retour primitif.

attribut wdo
Contient les détails de l'attribut d'objet de données de flux de travaux (voir «Objets de données de flux de travaux», à la page 17) que les données présentes dans la zone de type de retour associé vont mapper et conserver. Les attributs obligatoires sont décrits ci-après.

wdo-name
Il représente le nom de l'objet de données de flux de travaux utilisé dans le mappage de sortie.

name Ceci représente le nom de l'attribut de l'objet de données de flux dans le mappage de sortie.

Validations

- Aucun mappage de paramètre de sortie en double n'est autorisé. En d'autres termes, un attribut d'objet de données de flux de travaux ne peut être spécifié qu'une seule fois dans une liste de mappages de retour de sortie.
- Tous les attributs d'objet de données de flux de travaux indiqués dans les mappages de sortie doivent être des attributs d'objet de données de flux de travaux valides dans le cadre de la définition du processus de flux de travaux qui fait figure de contenant.
- Le type de champ de retour mappé depuis et le type d'attribut d'objet de données de flux de travaux mappé à doivent être compatibles.
- Les mappages de sortie ne peuvent pas être créés pour les attributs d'objet de données de flux de travaux qui ont été marqués comme des attributs d'objet de données de flux de travaux constants. Les attributs d'objet de données de flux de travaux constants représentent les données qui doivent rester constants tout au long du processus (voir «Métadonnées», à la page 19). Si ces attributs ont pu être utilisés dans les mappages de sortie, ces données seraient remplacées par celles qui sont spécifiées dans les mappages de sortie.
- Si la structure de retour est une structure de retour de liste, alors l'objet de données de flux de travaux utilisé dans le mappage de retour doit être un objet de données de flux de travaux de liste.

Informations d'exécution

Les valeurs des champs de type de retour définies dans les mappages de paramètre de sortie sont conservées à l'aide des attributs d'objet de données de flux de travaux spécifiés après que l'activité automatique associée a été exécutée.

Description des WDO (objet de données de flux de travaux) de contexte

Il existe deux objets de données de flux de travaux du contexte qui sont disponibles lorsque vous créez l'élément de données et les conditions de fonction pour les transitions provenant d'une activité automatique. Celles-ci sont décrites ci-après.

Objet de données de flux Context_Result
L'objet de données de flux Context_Result est disponible pour une utilisation dans un élément de données ou des conditions de fonction (voir «Conditions», à la page 112) pour une transition provenant d'une activité

automatique. Cela permet d'utiliser la version de cette valeur de retour de la méthode appelée dans les conditions à déclarer. Les conventions pour les attributs disponibles pour l'objet de données de flux de Context_Result sont les suivantes :

- Si le type de retour est un type de base, l'attribut disponible est appelé `value` (par exemple `Context_Result.value`).
- Si la valeur de retour est une structure, alors les valeurs d'attribut Context_Result sont disponibles dans toutes les zones présentes sur le retour de classe structure (par exemple, `Context_Result.description` etc.).
- Si la valeur de retour est une structure imbriquée (structure globale) alors les valeurs de l'attribut Context_Result disponibles seront les zones présentes dans la structure qui le contient (exemple, `Context_Result.description` etc.) et également les noms qualifiés complets de ces zones dans la structure imbriquée (par exemple, `Context_Result.dtls:concernRoleID` etc.). Quelle que soit la profondeur de l'imbrication de la structure de la valeur de retour, il n'y a qu'un seul objet de données de flux Context_Result disponible avec les noms des structures imbriquées faisant partie du nom d'attribut. Le séparateur entre une structure imbriquée et ses zones est représenté par un signe deux-points comme dans l'exemple ci-dessus.
- Si le type de retour est une structure listée, l'objet de données de flux de travaux `Context_Result` n'est pas disponible.

Objet de données de flux Context_Error

Une méthode BPO appelée par une activité automatique peut parfois échouer (par exemple, émettre une exception qui entraîne l'activité de transaction pour annulation). Lorsque cela se produit, il peut être utile de pouvoir modéliser des actions de suivi après l'échec. L'objet de données de flux de travaux `Context_Error` active ce type de "chemin d'erreur" de modélisation. Il est disponible pour une utilisation dans un élément de données ou des conditions de fonction (voir «Conditions», à la page 112) pour une transition depuis une activité automatique. L'objet de données de flux de travaux `Context_Error` a un attribut `exceptionOccurred` qui est décrit ci-après.

- L'attribut `exceptionOccurred` est une valeur booléenne indiquant si la méthode BPO associée à une activité automatique a échoué. Il prend par défaut la valeur false et est défini sur true si la méthode BPO associée échoue.

Lors de l'exécution, si la méthode BPO appelée dans une activité automatique échoue (est relancée le nombre de fois requis et échoue toujours), le moteur de flux définira l'attribut `exceptionOccurred` de `Context_Error` sur true. Toutes les transitions effectuées à l'aide de l'objet de données de flux `Context_Error` sont alors évaluées et suivies si elles se résolvent sur true. Ceci permet à une instance de processus de flux de travaux de poursuivre le chemin d'erreur défini, même si l'activité automatique a échoué.

Si la méthode BPO appelée échoue et qu'*aucune* transition n'est effectuée à l'aide de l'objet de données de flux `Context_Error`, alors l'activité est arrêtée et une entrée est créée dans la console Failed Messages Admin.

Remarque : L'objet de données de flux `Context_Error` ne tient pas compte de la *cause* de l'échec, uniquement s'il en existe une ou non.

Attente d'événement

Conditions préalables

- Les détails de base communs à tous les types d'activité pris en charge par le flux de travaux Cúram sont décrits dans «Activité de base», à la page 32 et sont applicables à l'activité d'attente d'événement décrite ici.

Présentation

L'application Cúram a la possibilité d'émettre des événements à différents points tout en informant les écouteurs enregistrés de ce qui s'est passé. Un certain nombre de programmes d'écoute d'événement différents peuvent être enregistrés pour écouter un événement spécifié. Ces programmes d'écoute d'événement sont des fonctions d'application qui implémentent l'interface `curam.util.events.impl.EventHandler`. Lorsqu'un événement spécifié est émis, le moteur de flux de travaux appelle la fonction de gestionnaire d'événements associée (voir le *Guide du développeur de serveur Cúram* pour plus de détails sur les événements et les gestionnaires d'événements).

Le flux de travaux utilise cette fonction d'une manière légèrement différente via les activités d'attente d'événement. Une activité d'attente d'événement met en pause l'exécution d'une branche particulière d'une instance de processus jusqu'à ce qu'un événement particulier se produise.

Liste des événements

Il n'est pas tout à fait exact de dire qu'une activité d'attente d'événement met en pause un processus de flux de travaux jusqu'à ce qu'un événement particulier soit émis. Une attente d'événement peut en effet définir n'importe quel nombre d'événements à attendre. S'il a été indiqué de ne pas attendre que tous ces événements soient émis pour terminer l'activité, le premier événement qui correspond à l'une des attentes d'événement indiquées va terminer l'activité et faire progresser le flux de travaux. Ce scénario permet de déterminer si oui ou non le reste des événements jamais émis n'a aucun effet sur le processus. Il est également possible d'indiquer que toutes les attentes d'événements doivent correspondre, en fonction des événements associés émis avant de terminer l'activité et de poursuivre le processus de flux de travaux.

Métadonnées

```
<event-wait-activity id="1" category="AC1">

    ...

    <event-wait wait-on-all-events="true">
      <events>
        <event event-class="Task" event-type="Close"
               identifier="1">
          <event-match-attribute name="taskID"
                                 wdo-name="Context_Task"/>
        </event>
        <event event-class="Parent" event-type="Approve"
               identifier="1">
          <event-match-attribute name="identifier"
            wdo-name="ParentList[Context_Loop.loopCount]"/>
        </event>
        <event event-class="Child" event-type="Approve"
               identifier="2">
          <event-match-attribute name="identifier"
                                 wdo-name="ChildDetails"/>
          <multiple-occurring-event>
            <list-wdo-name>ChildDetails</list-wdo-name>
          </multiple-occurring-event>
        </event>
      </events>
    </event-wait>

    ...

  </event-wait-activity>
```

event-wait
Cela contient les détails de l'attente de l'événement associée à l'activité spécifique. Cela inclut les détails de tous les événements de l'attente d'événement.

wait-on-all-events
La valeur de cet indicateur indique au moteur de flux de travaux s'il doit attendre que des événements soient émis pour l'ensemble des attentes d'événement indiquées avant de terminer l'activité associée. Si la valeur est false, le premier événement qui correspond à l'une des attentes d'événement indiquées entraîne la fin de l'activité associée et du flux de travaux progresse. Lorsque la valeur est définie sur true, un événement doit être émis pour chacune des attentes d'événements spécifiées pour l'activité avant que cette dernière ne soit terminée avant que le flux de travaux ne progresse.

events Cela contient les détails de tous les événements que l'activité spécifiée attend.

event Cela contient les détails d'un événement spécifique que cette activité attend. Les détails d'événement comprennent les attributs obligatoires suivant :

event-class
Cela représente la classe d'événement métier que ce processus attend.

event-type
Cela représente le type d'événement métier que ce processus attend. La combinaison event-class et event-type indique l'événement métier requis.

identifier
Cela représente l'identificateur unique de cet événement. L'identificateur ne doit être unique que dans la liste des événements de cette activité.

event-match-attribute
Cela représente l'attribut d'objet de données de flux de travaux (voir «Objets de données de flux de travaux», à la page 17) utilisé pour établir une correspondance avec l'instance requise de l'événement spécifique. Par exemple, dans le premier événement spécifié des métadonnées ci-dessus, l'attribut d'objet de données de flux de travaux fait référence à l'identificateur de tâche associé à la fermeture d'une tâche spécifique. Lorsque cet événement est émis, le moteur de flux de travaux utilise les données de l'attribut de mise en correspondance d'événement pour identifier manière unique la tâche à fermer.

multiple-occurring-event
Cela signifie que cet événement représente un événement se produisant plusieurs fois. Cela signifie que si ces métadonnées sont spécifiées pour un événement, le moteur de flux de travaux permet de créer un enregistrement d'attente d'événement pour chaque élément des objets de données de flux de travaux spécifiés de la liste comme les événements se produisant plusieurs fois lorsque cette activité est exécutée. Cela permet au moteur de flux de travaux d'attendre plusieurs occurrences d'un même événement.

Notez que lorsqu'un événement se produisant plusieurs fois est indiqué pour un événement donné, un attribut de l'objet de données de flux de travaux de liste associé doit être utilisé comme données de mise en correspondance de l'événement. Cela permet de s'assurer que chaque événement généré par le moteur de flux de travaux de l'événement se produisant plusieurs est unique.

list-wdo-name
Cela représente le nom de l'objet de données de flux de travaux de liste à utiliser comme événement se produisant plusieurs fois.

Validations

- Vous devez définir au moins un événement doit être défini pour les informations d'attente d'événement associées à une activité d'attente d'événement.
- La classe et le type d'événement indiqués de chaque événement métier doivent être des entrées valides sur les tables de base de données d'événement pertinentes.
- Un événement et un attribut de mise en correspondance d'événement associé ne peuvent être définis qu'une seule fois dans une activité d'attente d'événement. Autrement dit, la même classe d'événement, le même type d'événement et le même attribut de mise en correspondance d'événement ne peuvent être utilisés qu'une seule fois, comme un événement spécifique en attente d'une activité d'attente d'événement.
- L'attribut d'objet de données de flux de travaux mappé à l'attribut de mise en correspondance d'événement d'un événement doit être valide, et comme il est utilisé comme un identificateur unique dans le mécanisme de mise en

correspondance d'événement, il doit être de type LONG pour répercuter les identificateurs de 64 bits utilisés dans Cúram.

- L'objet de données de flux de travaux Context_Task ne peut être utilisé qu'en tant qu'attribut d'objet de données de flux de travaux de données de mise en correspondance d'événement si l'activité est manuelle ou manuelle parallèle et que l'événement ne se produit pas plusieurs fois.
- Si un élément indexé à partir d'un objet de données de flux de travaux de liste (par exemple : ParentList[Context_Loop.loopCount]) est utilisé en tant que données de mise en correspondance d'événement, l'objet de données du flux de travaux doit être un objet de données de flux de travaux de liste et l'activité contenant les mappages d'événement doit être mise en boucle.
- Si un élément indexé à partir de l'objet de données de flux de travaux de la liste parallèle est utilisé en tant que données de mise en correspondance d'événement, l'activité contenant le mappage doit être une activité parallèle (par exemple, ParallelListWDO[Context_Parallel.occurrenceCount]). L'objet de données de flux de travaux en cours d'indexation par le Context_Parallel de flux de données de l'objet doit être l'objet de données de flux de la liste d'activités parallèle.
- Si l'objet de données de flux de travaux de la liste des événements se produisant plusieurs fois n'a pas été spécifié pour l'événement et que l'activité contenant le mappage d'événement n'est pas une activité parallèle, alors un attribut de l'objet de données du flux de travaux de la liste ne peut être utilisé comme les données de mise en correspondance d'événement dudit événement.
- Si l'objet de données du flux de travaux de la liste des événements se produisant plusieurs fois a été spécifié pour l'événement, alors un attribut de l'objet de données du flux de travaux de la liste doit être utilisé comme les données de mise en correspondance d'événement dudit événement.
- L'attribut de l'objet de données de flux de travaux mappé à des événements répétitifs doit être valide. Il doit également être un objet de données de flux de travaux de liste.

Code

Un gestionnaire d'événement de flux de travaux est fourni avec Cúram et est automatiquement enregistré pour écouter les événements lancés dans l'application. Plusieurs attentes d'événements peuvent être enregistrées pour une instance d'activité particulière dans un processus de flux de travaux. Si l'indicateur *waitOnAllEvents* est défini sur false pour les données d'attente d'événement spécifiées, seule une de ces attentes d'événements doit correspondre pour compléter cette instance d'activité. Le gestionnaire d'événements de flux de travaux traitera cet événement en complétant l'instance d'activité indiquée et en faisant avancer le processus en commençant par le prochain ensemble d'activités dans le processus. Tous les autres enregistrements d'attente d'événement non sauvegardés pour l'instance d'activité complète sont supprimés. Si les mappages de sortie (voir «Mappages de sortie», à la page 57) ont été indiqués pour l'attente d'événement, ils seront conservés par le moteur de flux de travaux et peuvent être utilisés dans des activités et des transitions sous-jacentes du processus.

Lorsque l'option *waitOnAllEvents* est définie sur true, toutes les attentes d'événements indiquées de l'instance d'activité doivent correspondre en fonction des événements émis pour compléter l'activité et faire progresser le flux de travaux. Pour chaque événement émis qui correspond à une attente d'événement associée pour l'instance d'activité, le gestionnaire d'événements de flux de travaux va traiter l'événement en supprimant l'enregistrement d'attente d'événement associé et en conservant les mappages de sortie (voir «Mappages de sortie», à la page 57) spécifiées pour l'attente d'événement. Le traitement se poursuit jusqu'à ce

que l'ensemble des attentes d'événement associées correspondent en fonction des événements émis. C'est seulement à ce moment que le gestionnaire d'événements de flux de travaux va terminer l'instance d'activité indiquée et faire avancer le processus avancer en faisant avancer le processus en commençant par le prochain ensemble d'activités dans le processus.

Informations d'exécution

Un événement émis dans l'application peut uniquement entraîner la poursuite d'une instance de processus si l'événement correspond à celui qui est en attente et que l'attribut de mise en correspondance d'événement indiqué pour l'attente d'événement correspond aux données d'événement principales de l'événement.

Echéance

Une attente d'événement met en pause un processus de flux de travaux à la place d'un événement émis. Toutefois, dans de nombreux cas, il n'est pas souhaitable qu'un processus attende indéfiniment. Une chaîne d'événements peut se produire. Cela signifie que l'événement que le processus attend pourrait ne jamais être lancé. Par exemple, par chance l'événement peut être émis avant que le processus n'atteigne l'activité d'attente d'événement. Pour diminuer ce risque, il est possible d'indiquer éventuellement un échéance pour qu'un événement soit émis, après quoi un gestionnaire d'échéance est appelé.

Conditions préalables

- Les méthodes de gestionnaire d'échéance spécifiées d'une échéance d'attente d'événement sont des méthodes d'objet de processus métier Cúram. Les mappages d'entrée des paramètres formels de ces méthodes et leurs métadonnées associées sont décrites dans «Automatique», à la page 34. Il est recommandé de consulter ce chapitre pour obtenir une description de ces mappages.

Métadonnées

```
<event-wait-activity id="1" category="AC1">

    ...

    <deadline complete-activity="true">
      <duration>
        <mapped-duration>
          <wdo-attribute wdo-name="TaskCreateDetails"
            name="deadlineDuration" />
        </mapped-duration>
      </duration>
      <deadline-handler interface-name=
        "curam.core.sl.intf.WorkflowDeadlineFunction"
        method-name="defaultDeadlineHandler">
        <formal-parameters>
          <formal-parameter index="0">
            <struct type="curam.core.struct.TaskKey">
              <field name="taskID">
                <base-type type="INT64">
                  <wdo-attribute wdo-name="Context_Task"
                    name="taskID" />
                </base-type>
              </field>
            </struct>
          </formal-parameter>
          <formal-parameter index="1">
            <struct type="curam.core.struct.ChildKey">
              <field name="identifier">
                <base-type type="INT64">
                  <wdo-attribute wdo-name=
                    "ClaimantDependents[Context_Loop.loopCount]"
                    name="identifier" />
                </base-type>
              </field>
            </struct>
          </formal-parameter>
        </formal-parameters>
      </deadline-handler>
      <deadline-output-mappings>
        <duration-expired wdo-name="TaskDeadlineDetails"
          name="booleanValue" />
        <deadline-expiry-time wdo-name="TaskDeadlineDetails"
          name="dateTimeValue" />
      </deadline-output-mappings>
    </deadline>

    ...


  </event-wait-activity>
```

complete-activity

Il s'agit d'un indicateur booléen qui indique si l'activité doit terminer lorsque la durée de l'échéance expire. La valeur par défaut de l'indicateur est false.

duration

Cela représente la durée de temps qui peut s'écouler avant que la méthode du gestionnaire d'échéance ne soit appelée. La durée peut être représentée dans l'un des formats ci-après qui sera ensuite utilisé pour calculer la date et l'heure d'échéance de l'attente d'événement :

seconds
Cela représente le nombre de secondes qui peuvent s'écouler avant que le gestionnaire d'échéance ne soit appelé.

mapped-duration
Il s'agit de l'attribut d'un objet de données de flux de travaux qui peut être mappé, représentant ainsi le nombre de secondes qui peuvent s'écouler avant que le gestionnaire d'échéance ne soit appelé.

deadline-handler
Il s'agit de la méthode à appeler une fois que l'échéance a expiré. Les métadonnées suivantes doit être spécifiées pour un gestionnaire d'échéance :

interface-name
Cela représente le nom qualifié complet du nom de classe de l'interface du gestionnaire d'échéance.

method-name
Cela représente la méthode requise dans l'interface du gestionnaire d'échéance qui doit être appelée lorsque l'échéance expire.

formal-parameters
Cela contient la liste des paramètres de méthode du gestionnaire d'échéance et les attributs d'objet de données de flux de travaux associés qui sont mappés à ces paramètres lorsque le gestionnaire d'échéance est appelé. Pour plus d'informations sur les mappages du paramètre de la méthode, voir «Mappages des entrées», à la page 36.

deadline-output-mappings
Cela contient les données de sortie d'échéance qui peuvent être éventuellement mappées aux attributs d'objet de données de flux de travaux. Ces données indiquent si l'échéance a expiré ou non ainsi que la date et l'heure d'expiration de l'échéance.

Validations

- Si un gestionnaire d'échéance est spécifié, il doit faire référence à une méthode métier Cúram valide qui existe sur le chemin d'accès aux classes de l'application.
- Les attributs de l'objet de données de flux de travaux spécifiés dans les mappages d'entrée doivent être valides. Les critères définissant un attribut d'objet de données de flux de travaux valide peuvent être affichés dans «Validations», à la page 22
- Le type du paramètre formel en cours de mappage et le type d'attribut d'objet de données de flux de travaux en cours d'utilisation dans ce mappage d'entrée ne sont pas compatibles. Par exemple, si le mappage en cours de création est une zone de structure ayant un type STRING, l'attribut d'objet de données du flux de travaux en cours d'utilisation dans ce mappage doit également être un type STRING.
- Si l'élément indexé à partir d'un objet de données de flux de travaux de liste (par exemple : `ClaimantDependent[Context_Loop.loopCount]`) est en cours d'utilisation dans un mappage d'entrée, l'objet de données du flux de travaux associé doit être un objet de données de flux de travaux de liste et l'activité contenant les mappages d'entrée doit être mise en boucle.
- Si l'objet de données de flux de travaux `Context_Parallel` est en cours d'utilisation dans un mappage d'entrée, l'activité contentant les mappages d'entrée doit être une activité `Parallèle`.

- Si un élément indexé à partir de l'objet de données de flux de travaux de la liste parallèle est en cours d'utilisation dans un mappage d'entrée, l'activité contenant le mappage doit être une activité parallèle (par exemple, ParallelListWDO[Context_Parallel.occurrenceCount]). L'objet de données de flux de travaux en cours d'indexation par le Context_Parallel de flux de données de l'objet doit être l'objet de données de flux de la liste d'activités parallèle.
- La durée de l'échéance peut être indiquée en secondes pouvez ou à l'aide d'un mappage d'attribut d'objet de données de flux de travaux, mais pas les deux.
- Si la durée d'échéance a été spécifiée à l'aide d'un attribut d'objet de données de flux de travaux, l'attribut doit être valide et doit être de type INTEGER.
- Si une échéance a été spécifiée pour une activité, une fonction de gestionnaire d'échéance doit être spécifiée et/ou l'indicateur d'activité complète doit être définie sur true. Si tel n'est pas le cas, le flux de travaux n'aurait rien faire lorsque le délai est atteint.
- Si la valeur expirée de la durée des mappages de sortie d'échéance a été mappée à un attribut d'objet de données de flux de travaux, l'attribut doit être valide et de type BOOLEAN.
- Si la valeur temporelle expirée des mappages de sortie d'échéance a été mappée à un attribut d'objet de données de flux de travaux, l'attribut doit être valide et de type DATETIME.
- L'indicateur d'activité complète ne peut pas être défini sur `true` si l'activité contenant l'échéance est une activité parallèle. Cela est dû au fait que les activités parallèles ne prennent pas en charge les échéances de modèle.

Code

- Tous les paramètres de retour associés à la méthode du gestionnaire d'échéance ne sont pas utilisés dans le moteur de flux de travaux et sont donc sans intérêt.
- La fonction API du scanner d'échéance du flux de travaux `DeadlineScanner`. `scanDeadlines()` est fourni pour permettre l'analyse des échéances d'attente d'événement qui ont dépassé leur durée spécifiée. Toutes les attentes d'événement seront traitées et leur fonction de gestionnaire associée sera appelée ou l'activité associée sera terminée.

Informations d'exécution

Lorsque le moteur de flux de travaux exécute une activité qui contient des métadonnées d'échéance, il crée la date et l'heure d'échéance comme suit :

- Si la durée est spécifiée en secondes, le calcul se présente comme suit : date et heure en cours + secondes définies dans les métadonnées = date et heure d'échéance.
- Si la durée est spécifiée en tant qu'attribut de l'objet de données de flux de travaux, le calcul se présente comme suit : date et heure en cours + la valeur telle qu'elle est définie dans l'attribut d'objet de données de flux de travaux = date et heure d'échéance

Les échéances qui ont expiré sont traitées en appelant le travail par lots `ScanTaskDeadlines`. Ce travail par lots appelle à son tour l'interface de programme d'application du scanner d'échéance de flux de travaux décrite ci-dessus qui extrait une liste de toutes les échéances qui ont expiré et les traite. Si une méthode de gestionnaire d'échéance a été spécifiée pour l'échéance, les valeurs des attributs d'objets de données de flux de travaux définies dans les mappages de paramètres sont fournies comme des paramètres d'entrée à la méthode de gestionnaire d'échéance et celle-ci est appelée. Si l'indicateur d'activité complète a été défini sur true, alors l'activité associée est terminée. Tous les mappages de sortie d'échéance (durée expirée et heure d'expiration de l'échéance) qui ont peut-être été spécifiés

sont conservés ici. Les attributs de l'objet de données de flux de travaux `Context_Deadline` sont également conservés pendant ce traitement pour qu'ils puissent être utilisés dans les transitions émanant de l'activité contenant l'échéance.

Description des WDO (objet de données de flux de travaux) de contexte

L'objet de données de flux de travaux Context_Deadline est disponible pour une utilisation dans un élément de données ou des conditions de fonction (voir «Conditions», à la page 112) pour une transition à partir d'une activité avec une attente d'événement ayant une échéance. Les attributs de l'objet de données de flux de travaux Context_Deadline disponibles sont :

Context_Deadline.durationExpired
Représente une valeur booléenne indiquant si la durée d'échéance associée à l'activité a expiré.

Context_Deadline.expiryTime
Attribut contenant la date et l'heure auxquelles la durée d'échéance a expiré.

Mappages de sortie

L'événement émis contient certaines informations et il pourrait être utile de les mapper en retour au moteur de flux de travaux. L'événement contient des données d'événements principales et secondaires. Les données d'événement principales sont celles utilisées pour faire correspondre l'événement dès le début de sorte qu'il ne soit pas nécessaire d'effectuer un rétro mappage dans le processus. Cependant, les données d'événement secondaires peuvent être inconnues du moteur de flux de travaux et peuvent donc être mappées à ce dernier. En outre, puisqu'une activité d'attente d'événement peut mise en attente sur n'importe quel nombre d'événements, l'événement réel émis peut être intéressant et peut donc également être mappé au le moteur de flux de travaux. Enfin, l'utilisateur Cúram qui émet l'événement peut être intéressant et peut également être mappé au moteur de flux de travaux.

Il est à noter que si une instance d'activité doit attendre que toutes ses attentes d'événement associées correspondent, tous les mappages de sortie d'événement qui existent pour l'instance d'activité seront traités à chaque fois qu'un événement émis correspond à l'une des attentes d'événement.

Métadonnées

```
<event-wait-activity id="1" category="AC1">

    ...

    <event-output-mappings>
      <event-type wdo-name="CaseEventResult"
        name="eventType" />
      <output-data wdo-name="TaskCreateDetails"
        name="concernRoleID" />
      <raised-by wdo-name="CaseEventResult"
        name="eventRaisedBy" />
      <time-raised wdo-name="CaseEventResult"
        name="timeRaised" />
    </event-output-mappings>

    ...

  </event-wait-activity>
```

event-output-mappings
Cela contient les données qui peuvent être éventuellement mappées au moteur de flux de travaux de l'événement émis.

event-type
Cela contient l'événement métier émis que l'instance d'activité attendait.

output-data
Cela contient les données d'événement secondaires à mapper au moteur de flux de travaux.

raised-by
Cela contient le nom d'utilisateur Cúram qui a provoqué l'événement à émettre.

time-raised
Cela contient la date et l'heure auxquelles l'événement est émis.

Validations

- Le mappage de sortie d'événement du type d'événement, s'il est spécifié, doit être un attribut d'objet de données de flux de travaux valide et doit être de type STRING.
- Le mappage de sortie d'événement émis par le nom d'utilisateur, s'il est spécifié, doit être un attribut d'objet de données de flux de travaux valide et doit être de type STRING.
- Le mappage de sortie d'événement des données de sortie, s'il est spécifié, doit être un attribut d'objet de données de flux de travaux valide et doit être de type LONG.
- Le mappage de sortie émis en heure, s'il est spécifié, doit être un attribut d'objet de données de flux de travaux valide et doit être de type DATETIME.

Informations d'exécution

Lorsqu'un événement est émis dans l'application qu'une instance d'activité attend, tous les attributs d'objet de données de flux de travaux contenus dans les mappages de sortie d'événement qui ont été définis pour l'attente d'événement sont remplis et conservés avec les données pertinentes à partir de l'événement.

Description des WDO (objet de données de flux de travaux) de contexte

L'objet de données de flux de travaux Context_Event est disponible pour une utilisation dans un élément de données ou des conditions de fonction (voir «Conditions», à la page 112) pour une transition à partir d'une activité avec une attente d'événement. Les attributs de l'objet de données de flux de travaux Context_Event disponibles sont :

Context_Event.raisedByUserName
Nom de l'utilisateur qui a émis cet événement.

Context_Event.timeRaised
Heure à laquelle l'événement est émis.

Context_Event.fullyQualifiedEventType
Nom complet (la classe d'événement et le type d'événement) de l'événement métier émis.

Context_Event.outputData
Données d'événement secondaires associées à l'événement émis.

Rappels

Un rappel peut être défini sur n'importe quelle échéance associée à une activité manuelle, de décision, d'attente d'événement, manuelle parallèle ou à une activité de décision parallèle. Un montant arbitraire de rappels peut être spécifié. Les rappels utilisent les métadonnées de notification décrites dans la notification d'activité (voir le chapitre «Notifications d'activité», à la page 102). Cela signifie que l'objet, le corps de la notification, la stratégie et les actions d'allocation peuvent être spécifiés pour un rappel.

Métadonnées

```
<reminders>

         <reminder id="1" delivery-offset="DO1">
           <delivery-time>
             <seconds>93660</seconds>
           </delivery-time>

      or...

           <delivery-time>
             <mapped-delivery-time>
               <wdo-attribute  wdo-name="CaseWDO"
                               name="caseID"/>
             </mapped-delivery-time>
           </delivery-time>
      ...
           <notification delivery-mechanism="DM1">
              ...standard notification metadata
           </notification>
         </reminder>

       </reminders>
```

reminders
Ce paramètre est facultatif et encapsule toutes les balises de rappel de l'échéance.

reminder
Cela contient toutes les métadonnées de rappel de l'échéance, y compris les métadonnées de notification associées.

delivery-offset
Cela fait référence à une valeur de la table de codes `ReminderDeliveryOffset` indiquant les secondes ou l'heure de distribution mappée qui feront l'objet d'un décalage. Dans le cas d'une échéance, le décalage se fait à partir de l'heure d'expiration de l'échéance. Il s'agit du seul décalage actuellement pris en charge.

delivery-time
Cela contient la balise des secondes ou de l'heure de distribution mappée, selon l'élément spécifié.

seconds
Cette balise représente le nombre de secondes avant l'heure d'expiration de l'échéance à laquelle le rappel sera envoyé.

mapped-delivery-time
Cette balise représente un objet de données de flux de travaux contenant le nombre de secondes avant l'heure d'expiration de l'échéance à laquelle le rappel sera envoyé.

Validations

- Un rappel ne peut être créé si une échéance n'a pas été associée à l'activité appropriée. En outre, si une échéance existe, alors que le gestionnaire d'échéance n'a pas été défini ou que l'indicateur d'activité complète a été défini sur false, un rappel ne peut être créé.
- Chaque rappel a un identificateur. Il doit être unique à l'échéance à laquelle il est associé.
- L'heure ou les secondes de distribution mappée doivent être indiquées pour un rappel.
- Si des secondes sont indiquées, elles doivent précéder l'heure d'expiration de l'échéance.
- L'attribut d'objet de données de flux de travaux référencé par l'heure distribution mappée doit être de type INTEGER.
- Toutes les validations existantes des notifications d'activité (voir «Notifications d'activité», à la page 102) sont applicables aux métadonnées de notification associées au rappel.

Code

La fonction API du scanner d'échéance du flux de travaux `DeadlineScanner`. `scanDeadlines()` inclut un appel à la fonction `deliverReminders()` qui traite et fournit les rappels qui ont atteint leur heure de livraison.

Informations d'exécution

Lorsqu'une activité contenant des rappels a été exécutée, les rappels sont conservés sur l'entité `Reminders`. L'heure à laquelle un rappel doit être envoyé est calculée comme suit :

- La durée de distribution du rappel est extraite en secondes. Cela peut être spécifié directement en secondes ou dans un attribut d'objet de données de flux de travaux.
- La durée de l'échéance associée au rappel est extraite en secondes. Cela peut être spécifié directement en secondes ou dans un attribut d'objet de données de flux de travaux.
- Si la durée de distribution du rappel est un nombre positif et que ce nombre est inférieur à la durée d'échéance (les distributions de rappel ne peuvent être spécifiées pour les heures qui sont supérieures à la date et à l'heure d'échéance pour des raisons évidentes), l'heure de distribution de la notification de rappel est calculée comme suit : durée d'échéance - durée de distribution du rappel. Cette durée en secondes est ensuite convertie en date et en heure et est ajoutée à la date et à l'heure auxquelles le rappel est créé. Elle sera ensuite stockée dans l'enregistrement de rappel, comme la date et l'heure de la notification de rappel doivent être envoyées.

Les rappels configurés pour les échéances sont traités et envoyés en appelant le travail par lots `ScanTaskDeadlines` . Ce travail par lots appelle la fonction `DeadlineScanner scanDeadlines()` qui analyse des rappels qui arrivent à échéance et envoie les notifications de rappel associées (à l'aide de la stratégie d'allocation de notification de rappel pour déterminer les utilisateurs à qui les notifications doivent être envoyées). Les rappels envoyés sont supprimés de l'entité `Reminders` afin de s'assurer qu'ils ne sont pas envoyés une nouvelle fois. Lorsque l'activité se termine, tous les rappels configurés non envoyés de cette activité sont supprimés.

Manuel

Conditions préalables

- Les détails de base communs à tous les types d'activité pris en charge par le flux de travaux Cúram sont décrits dans «Activité de base», à la page 32 et sont applicables à l'activité manuelle décrite ici.

Présentation

Dans tout processus métier automatisé, il est nécessaire d'interagir avec des agents humains pour prendre des décisions, pour fournir des données supplémentaires ou pour effectuer des tâches dans le monde réel, comme passer un coup de téléphone à un client. Dans le flux de travaux Cúram, les étapes d'un processus sont modélisées à l'aide des activités manuelles. Une activité manuelle indique l'emplacement où l'intervention humaine du processus métier est requise. Elle indique également les informations que l'utilisateur recevra une fois averti qu'il doit effectuer une tâche et indique également la sélection des agents auxquels le travail sera affecté.

Détails de la tâche

Afin d'avertir un utilisateur qu'il doit effectuer un travail dans le cadre d'un processus métier automatisé, une tâche lui est affectée. Une tâche correspond à un message qui apparaît dans la boîte de réception des utilisateurs. Cette boîte de réception indique le travail que l'utilisateur est censé faire. La tâche peut également comporter une liste d'actions qui lui sont associées. Les actions sont des liens vers des pages de l'application Cúram où le travail nécessaire pour l'exécution de la tâche peut être effectué.

Métadonnées

```
<manual-activity id="1">
      ...
      <task>
        <message>
          <message-text>
            <localized-text>
              <locale language="fr">Le dossier
                suivant %1n pour %1s doit être approuvé</locale>
            </localized-text>
          </message-text>
          <message-parameters>
            <wdo-attribute wdo-name="TaskCreateDetails"
                           name="caseID"/>
            <wdo-attribute wdo-name=
                           "Claimant[Context_Loop.loopCount]"
                           name="caseID"/>
          </message-parameters>
        </message>
        <actions>
          <action page-id="Case_viewHome" principal-action="false"
           open-modal="false">
            <message>
              <message-text>
                <localized-text>
                  <locale language="en">
                    Page d'accueil du dossier : %1n</locale>
                </localized-text>
              </message-text>
              <message-parameters>
                <wdo-attribute wdo-name="TaskCreateDetails"
                               name="caseID"/>
              </message-parameters>
            </message>
            <link-parameter name="childID">
              <wdo-attribute wdo-name="ChildDependents"
                             name="identifier"/>
            </link-parameter>
            <link-parameter name="fullName">
              <wdo-attribute wdo-name="ChildDependents"
                             name="fullName"/>
            </link-parameter>
            <multiple-occurring-action>
              <list-wdo-name>ChildDependentList</list-wdo-name>
            </multiple-occurring-action>
          </action>
          <action page-id="Person_confirmPersonDetails"
                          principal-action="true"
                          open-modal="true">
            <message>
              <message-text>
                <localized-text>
                  <locale language="en">
                    Confirmez les coordonnées
                      de la personne %1s</locale>
                </localized-text>
              </message-text>
              <message-parameters>
                <wdo-attribute wdo-name=
                  "PersonDetailsList[Context_Loop.loopCount]"
                               name="fullName"/>
              </message-parameters>
            </message>
            <link-parameter name="identifier">
              <wdo-attribute wdo-name="
                PersonDetailsList[Context_Loop.loopCount]"
                             name="identifier"/>
            </link-parameter>
          </action>
        </actions>
        <task-priority>
```

task Cet élément contient tous les détails d'une tâche, y compris le message et les détails des actions associées. Les différentes métadonnées associées à une tâche sont décrites ci-après.

message
Cet élément contient les détails du message paramétré. Lorsqu'une activité manuelle est exécutée, une tâche est créée. Lorsqu'un utilisateur visualise ses tâches dans la boîte de réception, ce message représente l'objet de cette tâche.

message-text
Cet élément contient les détails du texte du message. Le texte de l'objet peut contenir des chaînes remplaçables (%k), qui seront remplacées par les paramètres de texte associés. Un paramètre de texte est un mappage à un attribut d'objet de données de flux de travaux. Le paramètre k dans la liste remplacera %k dans la chaîne de texte, où k représente l'ordre du paramètre dans la liste. Le paramètre %k peut être répété dans la chaîne, c'est pourquoi chaque attribut d'objet de données de flux de travaux doit être mappé une seule fois. Vous pouvez éventuellement spécifier un format pour les chaînes remplaçables en plaçant une autre lettre après la chaîne remplaçable, par exemple %1d, où d mettra en forme la valeur sous la forme d'une date.

Tableau 3. Conversion des données de texte de l'objet

Lettre du formatage	Format utilisé
s	chaîne
n	numérique
d	date
z	date/heure
t	heure

localized-text
Cet élément contient des détails sur le texte du message de la tâche localisable. Pour plus de détails sur le texte localisé et les métadonnées associées, voir «Texte localisé», à la page 33.

message-parameters
Il est possible d'associer des paramètres avec un message de tâche. Cet élément contient les détails des paramètres d'attribut de l'objet de données de flux de travaux utilisé pour remplacer les marques de réservation dans le texte associé. Pour plus de détails sur les objets de données de flux de travaux et leurs attributs, voir «Objets de données de flux de travaux», à la page 17.

actions
Cet élément contient les détails de toutes les actions associées à la tâche d'activité manuelle. Ces actions sont des liens vers des pages d'application Cúram où il est possible d'exécuter le travail nécessaire pour effectuer la tâche.

action Cette action contient la définition d'un lien hypertexte vers une page Cúram sur laquelle une tâche peut être effectuée. Les zones suivantes associées à l'action de tâche sont décrites ci-après.

page-id
Cet élément représente l'identificateur de la page Cúram cible sur laquelle un utilisateur peut commencer à effectuer l'action requise.

principal-action

Les actions peuvent être définies en tant qu'actions principales ou secondaires. Les actions principales contiennent généralement les liens vers les pages Cúram sur lesquelles un utilisateur peut commencer à exécuter le travail réel requis. Les actions secondaires contiennent généralement des liens vers les informations de support que l'utilisateur affecté au travail peut consulter lors de l'exécution de la tâche affectée.

open-modal

Les pages liées à partir d'une action de tâche peuvent être définies pour s'ouvrir dans une boîte de dialogue modale. Si cet indicateur est défini sur true, la page spécifiée par le lien d'action sera ouverte dans une boîte de dialogue modale. Si la valeur est false (valeur par défaut), l'infrastructure client décidera comment ouvrir le lien de la même manière qu'il le fait avec n'importe quel autre lien dans l'application (par exemple, si la page fait partie d'une configuration d'onglet, elle ouvrira l'onglet approprié - si ce n'est pas le cas, elle remplacera simplement la page d'accueil du lien de l'action dans la zone de contenu de l'onglet en cours).

message

Cet élément contient les détails du message paramétré qui est associé à l'action à effectuer, y compris le texte du message et les paramètres facultatifs qui peuvent être associés avec le texte.

link-parameter

Les liens vers les pages Cúram où le travail réel pour la tâche sera exécuté doivent contenir un identificateur de page (décrit ci-dessus) et les paramètres de page facultatifs. Ces paramètres de page sont décrits par ces métadonnées et ils représentent une paire nom/valeur dans laquelle l'attribut du nom correspond au nom d'un paramètre de lien (le nom de paramètre de page dans la page client Cúram liée) et où la valeur est fournie par un attribut d'objet de données de flux de travaux. La zone suivante associée au paramètre de lien est décrite ci-après.

name Nom du paramètre de lien.

multiple-occurring-action

Cet élément signifie que cette action représentera une action qui se produit plusieurs fois. Cela signifie que lorsque ces métadonnées sont spécifiées pour une action, le moteur de flux de travaux crée un enregistrement d'action pour chaque élément dans l'objet de données de flux de travaux de liste considérée comme l'action qui se produit plusieurs fois lorsque cette activité est exécutée.

Il est important de noter que lorsque l'action se produisant plusieurs fois est indiquée pour une action, un attribut de l'objet de données de flux de travaux de liste associé doit être utilisé comme paramètre de lien pour l'action.

list-wdo-name

Nom de l'objet de données de flux de travaux de type liste à utiliser avec l'action se produisant plusieurs fois.

attribut wdo

La valeur utilisée dans le paramètre de lien d'action est fournie par le mappage d'attribut d'objet de données de flux de travaux spécifié dans cet élément de métadonnées.

task-priority

Une tâche peut éventuellement contenir une priorité et ces métadonnées contiennent ces détails. La priorité d'une tâche est représentée selon l'un des formats ci-après :

priority

Dans ce cas, la priorité est sélectionnée dans l'outil de définition de processus et est extraite de la table de code `TaskPriority`.

mapped-priority

La priorité d'une tâche manuelle peut être mappée à l'aide d'un attribut d'objet de données de flux de travaux. Le fragment des métadonnées suivant fournit un exemple de la manière dont cela peut être réalisé :

```
<manual-activity id="1">
      ...
      <task>
        <message>
          ......
        </message>
        <actions>
          <action page-id="Case_viewHome" principal-action="true">
            .....
          </action>
        </actions>
        <task-priority>
          <mapped-priority>
            <wdo-attribute wdo-name="WorkflowTestWDO"
                           name="taskPriority"/>
          </mapped-priority>
        </task-priority>
        .....
      </task>
      ...
    </manual-activity>
```

allow-deadline-override

Cet élément représente un indicateur booléen utilisé pour indiquer si la date limite (voir «Echéance», à la page 53) associée à la tâche d'activité manuelle peut être remplacée. La définition de la valeur de cet indicateur sur `true` (la valeur par défaut est`false`) indique que l'heure d'échéance peut être modifiée une fois que la tâche a été créée par le moteur de flux de travaux.

allow-task-forwarding

Cet élément représente un indicateur booléen utilisé pour indiquer si la tâche générée en raison de l'exécution de l'activité manuelle associée peut être transférée à un autre utilisateur. Lorsqu'une tâche est générée, elle est attribuée à un agent pour qu'il effectue le travail. La définition de cette option sur `true` (la valeur par défaut est `true`) permet à l'agent de transmettre cette tâche à un autre agent pour réaliser le travail spécifié.

administration-sid

Cette zone permet de spécifier un identificateur de sécurité d'administration pour une tâche manuelle. Elle permet à un utilisateur dans un groupe associé à l'identificateur de sécurité spécifié de modifier les détails de la tâche, bien que la tâche peut être réservée par un autre utilisateur dans l'application.

initial-comment

Cette zone permet de spécifier un mappage de commentaire initial pour la tâche manuelle. La valeur de l'attribut d'objet de données de flux de travaux utilisée dans ce mappage permet de placer un enregistrement dans la table `TaskHistory` lorsque l'activité manuelle associée est exécutée.

Validations

- Un objet doit être défini pour la tâche d'activité manuelle. Il s'agit d'une chaîne localisable.
- Tous les objets de données de flux de travaux utilisés en tant que paramètres de texte d'objet dans le message d'objet de la tâche d'activité manuelle doivent être des attributs d'objet de données de flux de travaux valides dans le contexte de la définition de processus de flux de travaux qui le contient.
- Si un élément indexé d'un objet de données de flux de travaux de liste (par exemple, PersonDetailsList[Context_loop.loopcount]) est utilisé en tant que paramètre de texte d'objet, l'objet de données de flux de travaux de liste doit obligatoirement être un objet de données de flux de travaux de liste et l'activité contenant le mappage doit être contenue dans une boucle.
- Si l'objet de données de flux de travaux `Context_Parallel` est utilisé en tant que paramètre de texte d'objet, l'activité contenant le mappage doit être une activité manuelle `Parallel`.
- Si un élément indexé de l'objet de données de flux de travaux de liste parallèle est utilisé en tant que paramètre de texte d'objet, l'activité contenant le mappage doit être une activité parallèle (par exemple, ParallelListWDO[Context_Parallel.occurrenceCount]). L'objet de données de flux de travaux en cours d'indexation par le Context_Parallel de flux de données de l'objet doit être l'objet de données de flux de la liste d'activités parallèle.
- Si des actions ont été spécifiées pour la tâche d'activité manuelle, tous les attributs d'objet de données de flux de travaux utilisés en tant que mappages pour les paramètres de texte d'action doivent être valides dans le contexte de la définition de processus de flux de travaux qui le contient.
- Si des actions ont été spécifiées pour la tâche d'activité manuelle, tous les attributs d'objet de données de flux de travaux utilisés dans les mappages de paramètre de lien d'action d'une action d'activité manuelle doivent être valides dans le contexte de la définition de processus de flux de travaux qui le contient.
- Si un élément indexé d'un objet de données de flux de travaux de liste (par exemple, PersonDetailsList[Context_Loop.loopCount]) est utilisé dans les mappages de paramètre de lien d'action ou de texte d'action, l'objet de données de flux de travaux doit être un objet de données de flux de travaux de liste et l'activité contenant le mappage doit être contenue dans une boucle.
- Si l'objet de données de flux de travaux `Context_Parallel` est utilisé dans les mappages de paramètre de lien d'action ou de texte d'action, l'activité contenant le mappage doit être une activité manuelle `Parallel`.
- Si un élément indexé de l'objet de données de flux de travaux de liste parallèle est utilisé dans les mappages de paramètre de texte d'action ou de lien d'action, l'activité contenant le mappage doit être une activité parallèle (par exemple, ParallelListWDO[Context_Parallel.occurrenceCount]). L'objet de données de flux de travaux en cours d'indexation par le Context_Parallel de flux de données de l'objet doit être l'objet de données de flux de la liste d'activités parallèle.
- Le nombre de paramètres substituables utilisés dans le texte d'objet et le texte d'action de la tâche d'activité manuelle doit être égal au nombre d'attributs d'objet de données de flux de travaux mappés pour tous les paramètres régionaux définis.

- Il est possible d'indiquer la priorité d'une tâche manuelle à l'aide d'une valeur de code d'une table de codes ou d'un mappage d'attributs d'objet de données de flux de travaux, mais pas à l'aide des deux en même temps.
- Si une priorité mappée a été spécifiée pour la tâche d'activité manuelle, l'attribut d'objet de données de flux de travaux indiqué doit être valide dans le contexte de la définition de processus de flux de travaux qui le contient. Il doit également être du type STRING.
- Si un mappage de commentaire initial a été spécifié pour la tâche d'activité manuelle, l'attribut d'objet de données de flux de travaux spécifié doit être valide dans le contexte de la définition de processus de flux de travaux qui le contient. Il doit également être du type STRING.
- L'objet de données de flux de travaux spécifié devant être utilisé dans l'action récurrente doit être un objet de données de flux de travaux valide dans le cadre de la définition du processus de flux de travaux le contenant. Il doit également s'agir d'un objet de données de flux de travaux de liste.
- Au moins un attribut de l'objet de données de flux de travaux de liste d'action se produisant plusieurs fois doit être utilisé dans les paramètres de lien spécifiés pour une action se produisant plusieurs fois.

Code

Pages d'action et paramètres de page d'action

Les actions spécifiées pour la tâche d'activité manuelle sont des liens vers des pages de l'application Cúram où le travail nécessaire pour l'exécution de la tâche peut être effectué. Les pages indiquées dans les actions de tâche doivent être des pages Cúram valides et doivent être disponibles dans l'application Cúram. Les paramètres dans ces pages doivent correspondre aux paramètres spécifiés en tant que paramètres de lien d'action dans les actions de tâche associées.

API LocalizableStringResolver TaskStringResolver

L'objet de la tâche et les messages d'action de la tâche associés sont affichés dans la boîte de réception de l'utilisateur pour les informer du travail nécessaire pour terminer pour la tâche associée. L'API `LocalizableStringResolver.TaskStringResolver` contient les fonctions permettant de résoudre les objets de tâche et les messages d'action pour les bons paramètres régionaux de l'utilisateur. Le remplacement des paramètres substituables par les valeurs d'attribut d'objet de données de flux de travaux associées indiquées dans les mappages associés est également réalisé dans le cadre de ces fonctions.

API d'administration de tâches

Un certain nombre de fonctions ont été fournies sur la classe `TaskAdmin` afin de permettre la manipulation des tâches. Pour plus de détails sur les fonctions disponibles, voir la spécification Javadoc associée à la classe `TaskAdmin`.

API d'administration de l'historique des tâches

Divers événements de cycle de vie pour une tâche (par exemple, lorsqu'une tâche est créée ; lorsqu'une tâche est attribuée ; lorsqu'une tâche est fermée) sont écris dans la table `TaskHistory` au cours de la durée de vie d'une tâche. Un certain nombre de fonctions de recherche ont été fournies dans cette classe d'API afin d'examiner ces entrées. Pour plus de détails sur les fonctions disponibles, voir la spécification Javadoc associée à l'entité `TaskHistoryAdmin`.

API d'administration des échéances de flux de travaux

Un certain nombre de fonctions ont été fournies sur la classe

`WorkflowDeadlineAdmin` afin de permettre la manipulation des échéances de flux de travaux. Pour plus de détails sur les fonctions disponibles, voir la spécification Javadoc associée à la classe `WorkflowDeadlineAdmin`.

Informations d'exécution

Lorsqu'une activité manuelle est exécutée par le moteur de flux de travaux, une tâche est créée et est attribuée à un agent pour qu'il effectue ce travail (voir «Stratégie d'allocation»).

Description des WDO (objet de données de flux de travaux) de contexte

L'objet de données de flux de travaux Context_Task permet à l'identificateur unique de la tâche créée dans le cadre de l'exécution de l'activité manuelle associée d'être disponible pour une utilisation dans les divers mappages de métadonnées associés à une activité manuelle. Ces mappages incluent par exemple les mappages de données de correspondance d'événement (voir «Liste des événements», à la page 49) et les mappages d'entrée de la fonction d'échéance (voir «Echéance», à la page 53). Le seul attribut disponible sur cet objet de données de flux de travaux est le suivant :

Context_Task.taskID
L'attribut `taskID` représente l'identificateur unique de la tâche créée lorsque l'activité manuelle associée est exécutée.

Stratégie d'allocation

Une organisation comporte généralement de nombreux agents humains à différents niveaux de responsabilité qui peuvent effectuer un travail pour une définition de processus donnée. Pour sélectionner un agent ou groupe d'agents spécifique qui peut faire le travail pour une activité manuelle spécifique, une stratégie d'allocation est affectée à l'activité. Il existe quatre types de stratégie d'allocation actuellement pris en charge par le flux de travaux Cúram : fonction, règles classiques, règles Cúram Express et cible. Lorsqu'une stratégie d'allocation de type cible est sélectionnée, l'agent ou le groupe d'agents à qui le travail doit être affecté est directement nommé. La sélection d'une stratégie d'allocation de fonction entraîne l'appel de la fonction d'allocation spécifiée lorsque l'activité associée est exécutée par le moteur de flux de travaux. Enfin, si une stratégie d'allocation classique ou Cúram Express rules (CER) est sélectionnée, le jeu de règles spécifié est exécuté lorsque l'activité associée est exécutée.

Conditions préalables

- Si la stratégie d'allocation associée à une activité manuelle est du type `Function`, ces fonctions d'allocation représentent des méthodes métier Cúra avec une signature spécifique. Les mappages d'entrée pour les paramètres formels de ces méthodes et leurs métadonnées associées sont décrits dans «Automatique», à la page 34. Il est recommandé de consulter ce chapitre pour obtenir une description de ces mappages.

Métadonnées

Comme indiqué précédemment, il existe quatre types de stratégies d'allocation. Les métadonnées requises pour chacun de ces types sont décrites dans les sections suivantes.

allocation-strategy
Cet élément contient les détails de la stratégie d'allocation définie pour la tâche manuelle. Les zones suivantes associées à une stratégie d'allocation sont décrites ci-après.

type Cette élément contient le type de la stratégie d'allocation. Les quatre types de stratégie d'allocation actuellement en charge par le flux de travaux Cúram sont : fonction, règle classique, règles CER et cible.

identificateur
Cet élément représente l'identificateur de la stratégie d'allocation. Pour une stratégie d'allocation de type de fonction, cet identificateur représente le nom complet de la fonction d'allocation en cours d'utilisation. Pour une stratégie d'allocation de type règle ou règle CER, cet identificateur représente l'identificateur du jeu de règles en cours d'utilisation. Enfin, lorsqu'une stratégie d'allocation de type cible est sélectionnée, cet identificateur représente l'identificateur de la cible d'allocation en cours d'utilisation.

Stratégie d'allocation de fonction :

```
<manual-activity id="1" category="AC1">
      ...
      <task>
        ...
      </task>
      <allocation-strategy
        identifier="curam.core.sl.intf.
          WorkflowAllocationFunction.manualAllocationStrategy"
        type="function">
        <function-mappings>
          <formal-parameters>
            <formal-parameter index="0">
              <base-type type="INT32">
                <wdo-attribute wdo-name="Context_Task"
                               name="taskID"/>
              </base-type>
            </formal-parameter>
            <formal-parameter index="1">
              <base-type type="INT64">
                <wdo-attribute
                  wdo-name="Context_RuntimeInformation"
                  name="processInstanceID"/>
              </base-type>
            </formal-parameter>
            <formal-parameter index="2">
              <struct type="curam.struct.TaskDetails">
                <field name="taskID">
                  <base-type type="INT64">
                    <wdo-attribute wdo-name="Context_Task"
                                   name="taskID"/>
                  </base-type>
                </field>
                <field name="category">
                  <base-type type="STRING">
                    <wdo-attribute wdo-name="TaskCreateDetails"
                                   name="category"/>
                  </base-type>
                </field>
              </struct>
            </formal-parameter>
            <formal-parameter index="3">
              <struct type="curam.struct.PersonDetails">
                <field name="identifier">
                  <base-type type="INT64">
                    <wdo-attribute wdo-name=
                      "PersonDetailsList[Context_Loop.loopCount]"
                                   name="identifier"/>
                  </base-type>
                </field>
                <field name="fullName">
                  <base-type type="STRING">
                    <wdo-attribute wdo-name=
                      "PersonDetailsList[Context_Loop.loopCount]"
                                   name="fullName"/>
                  </base-type>
                </field>
              </struct>
            </formal-parameter>
          </formal-parameters>
        </function-mappings>
      </allocation-strategy>
      <event-wait>
        ...
      </event-wait>
    </manual-activity>
```

function-mappings

Cet élément contient les détails des mappages d'entrée pour les paramètres formels de la fonction d'allocation indiquée. Les fonctions d'allocation sont les méthodes métier Cúram (similaires à celles qui sont indiquées pour les activités automatiques) qui disposent d'une signature de retour différente (les fonctions d'allocation doivent disposer d'un type de retour `curam.util.workflow.struct.AllocationTargetList`). Par conséquent, les métadonnées utilisées pour ces mappages sont les mêmes que celles utilisées pour les mappages d'entrée pour les méthodes d'objet de processus métier qui sont associées aux activités automatiques. Il est conseillé de consulter la section «Mappages des entrées», à la page 36 du chapitre sur les activités automatiques pour plus de détails sur ces métadonnées et leurs significations.

Allocation de règles classiques :

```
<manual-activity id="1" category="AC1">
      ...
      <task>
        ...
      </task>
      <allocation-strategy type="rule"
                           identifier="PRODUCT_1">
        <ruleset-mappings>
          <rdo-mapping>
            <source-attribute wdo-name="TaskCreateDetails"
                              name="caseID" />
            <target-attribute rdo-name="TaskDetails"
                              name="caseID" />
          </rdo-mapping>
          <rdo-mapping>
            <source-attribute wdo-name="TaskCreateDetails"
                              name="concernRoleID" />
            <target-attribute rdo-name="TaskDetails"
                              name="concernRoleID" />
          </rdo-mapping>
        </ruleset-mappings>
      </allocation-strategy>
      <event-wait>
        ...
      </event-wait>
      ...
    </manual-activity>
```

ruleset-mappings

Cette zone contient les détails de tous les mappages pour le jeu de règles (*ruleset*) indiqué dans l'identificateur d'allocation. Il n'est pas obligatoire de mapper tous les attributs d'objet de données de règles spécifiés dans le jeu de règles (des mappages pour un sous-ensemble peuvent être créés).

rdo-mapping

Cette zone contient les détails d'un mappage entre un attribut d'objet de données de règles spécifié dans le jeu de règles d'allocation et son attribut d'objet de données de flux de travaux associé. Les métadonnées suivantes constituent un mappage valide :

source-attribute

Cette zone contient les détails de l'attribut source dans le mappage (par exemple, l'emplacement à partir duquel les données seront fournies au moment de l'exécution). Un attribut source se compose d'un objet de données de flux de travaux et du nom de son attribut associé (voir «Objets de données de flux de travaux», à la page 17).

target-attribute

Cette zone contient les détails de l'attribut cible dans le mappage (par exemple. l'emplacement vers lequel les données seront mappées au moment de l'exécution). Un attribut cible se compose d'un nom d'objet de données de règles et du nom de son attribut associé.

Allocation de règles CER :

```
<manual-activity id="1" category="AC1">
      ...
      <task>
        ...
      </task>
      <allocation-strategy type="curam express rule"
                           identifier="Sample Allocation Rules">
        <cer-set-mappings primary-class="SampleAllocationClass">
          <cer-class-mapping>
            <source-attribute wdo-name="TaskCreateDetails"
                              name="caseID" />
            <target-attribute cer-class-name="SampleAllocationClass"
                              name="caseID" />
          </cer-class-mapping>
          <cer-class-mapping>
            <source-attribute wdo-name="TaskCreateDetails"
                              name="subject" />
            <target-attribute cer-class-name="RuleClassA"
                              name="subject" />
          </cer-class-mapping>
          <cer-class-mapping>
            <source-attribute wdo-name="ListTaskDetails"
                              name="employerIDs" />
            <target-attribute cer-class-name="RuleClassA"
                              name="listOfEmployerIDs" />
          </cer-class-mapping>
          <cer-class-mapping>
            <source-attribute wdo-name="ListTaskDetails"
                              name="concernRoleID" />
            <target-attribute cer-class-name="SampleAllocationClass"
                              name="listConcernRoleIDs" />
          </cer-class-mapping>
          <cer-class-mapping>
            <source-attribute wdo-name="ListTaskDetails"
                              name="participantIDs" />
            <target-attribute cer-class-name="SampleAllocationClass.listRuleClassB.RuleClassB"
                              name="participantIDs" />
          </cer-class-mapping>
        </cer-set-mappings>
      </allocation-strategy>
      <event-wait>
        ...
      </event-wait>
      ...
    </manual-activity>
```

cer-set-mappings

Contient les détails de tous les mappages du *jeu de règles CER* indiqué dans l'identificateur d'allocation. La balise de métadonnées *primary-class* doit désigner une classe de règles CER qui contient un attribut appelé *targets*, car le moteur de flux de travaux utilise un attribut de ce nom pour déterminer la liste des cibles d'allocation pour la stratégie d'allocation spécifiée. Des mappages doivent être créés pour tous les attributs qui sont signalés comme spécifiés dans toutes les classes de règles CER utilisées pour la stratégie d'allocation.

cer-class-mapping

Cette balise contient les détails d'un mappage entre un attribut de classe de règles spécifié dans le jeu de règles CER et l'attribut d'objet de données de flux de travaux qui lui est associé. L'un de ces mappages doit exister pour chaque attribut de classe de règles CER signalé comme *spécifié* et qui est utilisé dans la stratégie d'allocation. Les métadonnées suivantes constituent un mappage valide :

source-attribute

Cette zone contient les détails de l'attribut source dans le mappage (par exemple, l'emplacement à partir duquel les données seront fournies au moment de l'exécution). Un attribut source se compose d'un objet de données de flux de travaux et du nom de son attribut associé (voir «Objets de données de flux de travaux», à la page 17). Des attributs d'objet de données de flux de travaux de liste peuvent également être utilisés ici si le mappage en cours de création est lié à un type de liste d'attributs de classe de règles CER.

target-attribute

Cette zone contient les détails de l'attribut cible dans le mappage (par exemple. l'emplacement vers lequel les données seront mappées au moment de l'exécution). Un attribut cible se compose d'un nom de classe CER et de son nom d'attribut associé. Certains types d'attribut de classe de règles CER sont pris en charge dans les mappages de stratégie d'allocation. Il s'agit de *String*, *Boolean*, *Number*, *Date* et *DateTime*. Une liste de classes de règles peut également être spécifiée ainsi que des listes de types de base décrits précédemment.

Stratégie d'allocation de cible :

```
<manual-activity id="1" category="AC1">
      ...
      <task>
      ...
      </task>
      <allocation-strategy type="target"
                           identifier="HEARINGSCHEDULER"/>
      <event-wait>
      ...
      </event-wait>
      ...
    </manual-activity>
```

Aucune métadonnée supplémentaire n'est requise pour décrire une stratégie d'allocation de type *target*. Comme indiqué précédemment, l'identificateur dans ce cas est l'identificateur de la cible d'allocation contenant l'agent ou groupe d'agents auquel la tâche sera affectée.

Validations

- Une stratégie d'allocation doit être définie pour une tâche manuelle.
- Si la stratégie d'allocation est de type fonction, la fonction d'allocation doit être valide et exister dans le chemin d'accès aux classes de l'application Cùram.
- Si la stratégie d'allocation est de type fonction, le type de retour de la fonction doit être `curam.util.workflow.struct.AllocationTargetList`.
- Si la stratégie d'allocation est de type fonction, tous les paramètres d'entrée de la fonction spécifiée qui sont mappés doivent être des attributs d'objet de données

de flux de travaux valides et le type de l'attribut d'objet de données de flux de travaux doit correspondre au type de la zone du paramètre d'entrée.

- Si la stratégie d'allocation est de type fonction et qu'un élément indexé d'un objet de données de flux de travaux de liste est utilisé dans un mappage d'entrée, l'objet de données de flux de travaux doit être un objet de données de flux de travaux de liste et l'activité contenant le mappage doit être contenue dans une boucle.
- Si la stratégie d'allocation est de type classique ou règle CER, le jeu de règles spécifié doit être valide.
- Si la stratégie d'allocation est de type règle CER, un nom de classe de règles CER principale doit être spécifié.
- Si la stratégie d'allocation est de type règle CER, la classe de règles CER principale spécifiée doit exister dans le jeu de règles CER spécifié.
- Si la stratégie d'allocation est de type règle CER, la classe de règles CER principale spécifiée doit étendre la classe de règles CER d'allocation de flux de travaux abstraite requise.
- Si la stratégie d'allocation est de type règle CER, la classe de règles CER principale spécifiée doit contenir un attribut appelé *targets*.
- Si la stratégie d'allocation est de type règle CER et si des classes de règles CER autres que les classes de règles CER principales ont été spécifiées dans les mappages d'entrée, la classe principale CER doit contenir des attributs qui référencent ces classes (un attribut par classe).
- Si la stratégie d'allocation est de type règle CER, tous les attributs source spécifiés dans les mappages doivent être des attributs d'objet de données de flux de travaux valides dans le contexte de la définition de processus de flux de travaux faisant office de conteneur. Tous les attributs cible doivent être des attributs de classe CER valides dans le contexte de le jeu de règles spécifié. Le type d'attribut d'objet de données de flux de travaux spécifié comme attribut source doit correspondre au type de l'attribut de classe CER spécifié comme attribut cible dans le mappage.
- Si la stratégie d'allocation est de type règle CER, aucun mappage d'attribut cible en double n'est admis. En d'autres termes, un attribut de classe de règles CER ne peut être spécifié qu'une fois dans une liste de mappages de classes CER.
- Si la stratégie d'allocation est de type règle CER, tous les attributs signalés comme *spécifiés* pour toutes les classes de règles CER utilisées pour la stratégie d'allocation doivent contenir un mappage d'entrée. Les attributs de classe CER qui ne sont pas signalés comme *spécifiés* ne doivent pas contenir de mappage d'entrée.
- Si la stratégie d'allocation est de type règle classique, tous les attributs source spécifiés dans les mappages doivent être des attributs d'objet de données de flux de travaux valides dans le contexte de la définition de processus de flux de travaux faisant office de conteneur. Tous les attributs cible doivent être des attributs d'objet de données de règles valides dans le contexte de le jeu de règles spécifié. Le type de l'attribut d'objet de données de flux de travaux spécifié comme attribut source doit correspondre au type de l'attribut d'objet de données de règles spécifié en tant qu'attribut cible dans le mappage.
- Si la stratégie d'allocation est de type règle classique, aucun mappage d'attribut cible en double n'est admis. En d'autres termes, un attribut d'objet de données de règles ne peut être spécifié qu'une seule fois dans une liste de mappages de jeu de règles.
- Si un élément indexé d'un objet de données de flux de travaux de liste (par exemple, PersonDetailsList[Context_Loop.loopCount]) est utilisé dans les mappages de stratégie d'allocation de règle CER, classique ou de fonction, l'objet

de données de flux de travaux doit être un objet de données de flux de travaux de liste et l'activité contenant le mappage doit être contenue dans une boucle.

- Si l'objet de données de flux de travaux `Context_Parallel` est utilisé dans les mappages de stratégie d'allocation de règle CER, classique ou de fonction, l'activité contenant le mappage doit être une activité `Parallel`.
- Si un élément indexé de l'objet de données de flux de travaux de liste parallèle est utilisé dans les mappages de stratégie d'allocation de règle CER, classique ou de fonction, l'activité contenant le mappage doit être une activité parallèle (par exemple, ParallelListWDO[Context_Parallel.occurrenceCount]). L'objet de données de flux de travaux en cours d'indexation par le Context_Parallel de flux de données de l'objet doit être l'objet de données de flux de la liste d'activités parallèle.

Code

Comme indiqué précédemment, n'importe quelle méthode d'objet de processus métier indiquée comme une fonction d'allocation doit renvoyer une structure de type `curam.util.workflow.struct.AllocationTargetList`.

Comme c'est le cas avec les méthodes métier associées à des activités automatiques, un échec de la fonction d'allocation lorsqu'une activité manuelle est exécutée entraîne l'appel de la stratégie de gestion des erreurs de flux de travaux. Ainsi, l'activité associée à la méthode en échec peut, par exemple, être réessayée plusieurs fois. De ce fait, les fonctions d'allocation associées aux stratégies d'allocation des activités de décision ou manuelles doivent généralement ne pas émettre d'exceptions à moins qu'une situation irrémédiable ne se produise.

L'application doit implémenter l'interface de rappel `curam.util.workflow.impl.WorkResolver` afin de déterminer comment les tâches seront affectées dans l'application. La propriété de l'application curam.custom.workflow.workresolver doit faire référence à la classe d'implémentation `curam.util.workflow.impl.WorkResolver` dans l'application étant donné que le moteur de flux de travaux utilisera cette propriété pour déterminer la fonction correcte pour attribuer la tâche.

La classe `curam.util.workflow.impl.WorkResolver` contient une méthode `resolveWork` surchargée car les différents types de stratégie d'allocation renvoient les cibles d'allocation dans différents formats. Toutefois, il s'agit d'un détail d'implémentation que les développeurs des classes de programme de résolution de travail personnalisées ne doivent pas avoir à traiter, notamment parce que le traitement métier pour toutes les versions de la méthode doit être le même.

```
package curam.util.workflow.impl;

      ...

      public interface WorkResolver {

        void resolveWork(
          final TaskDetails taskDetails,
          final Object allocationTargets,
          final boolean previouslyAllocated);

        void resolveWork(
            final TaskDetails taskDetails,
            final Map allocationTargets,
            final boolean previouslyAllocated);

        void resolveWork(
            final TaskDetails taskDetails,
            final String allocationTargetID,
            final boolean previouslyAllocated);

            ...
      }
```

Pour atténuer ce problème, `curam.core.sl.impl.DefaultWorkResolverAdapter` fournit un mécanisme plus pratique pour l'implémentation d'un programme de résolution de travail. Cette classe implémente les différentes méthodes et convertit leurs paramètres d'entrée en listes de cible d'allocation permettant aux développeurs de logique de résolution de travail personnalisé d'étendre cette classe et d'implémenter une méthode qui est appelée quelle que soit la source de cibles d'allocation.

```
package curam.core.sl.impl;

      ...

      public abstract class DefaultWorkResolverAdapter
        implements curam.util.workflow.impl.WorkResolver {

        public abstract void resolveWork(
          final TaskDetails taskDetails,
          final AllocationTargetList allocationTargets,
          final boolean previouslyAllocated);

            ...
      }
```

Outre cette classe d'adaptateur, l'application est livrée avec une implémentation de résolution de travail prête à l'emploi. Cette classe est appelée `curam.core.sl.impl.DefaultWorkResolver` et elle sert d'exemple sur la manière d'étendre l'adaptateur.

Informations d'exécution

Lorsqu'une activité manuelle est exécutée, le moteur de flux de travaux traite la stratégie d'allocation définie dans les métadonnées afin de récupérer la liste des cibles d'allocation pour cette tâche.

Si la stratégie d'allocation est du type fonction, le moteur de flux de travaux traite les mappages d'entrée définis pour la fonction d'allocation associée et l'appelle afin de récupérer la liste des cibles d'allocation.

Si la stratégie d'allocation est de type règle classique, le moteur de flux de travaux traite les mappages pour le jeu de règles spécifié et appelle le moteur de règles pour qu'il exécute le jeu de règles afin d'extraire la liste des cibles d'allocation.

Si la stratégie d'allocation est de type règle CER, le moteur de flux de travaux traite les mappages de classe CER qui ont été spécifiés pour la stratégie d'allocation. Les données provenant des attributs d'objet de données de flux de travaux sont mappées aux attributs de règle de classes CER correspondants. La classe principale est ensuite extraire et l'attribut *targets* est interrogé pour extraire la liste des cibles d'allocation.

Si la stratégie d'allocation est du type cible, la cible d'allocation est tout simplement celle qui est spécifiée dans les métadonnées et aucun traitement supplémentaire n'est requis.

Comme décrite dans les métadonnées pour un processus de flux de travaux (voir «Métadonnées d'une définition de processus», à la page 13), une stratégie d'allocation d'échec peut être spécifiée pour un processus. Cette stratégie sera traitée et utilisée si l'appel de la stratégie d'allocation associée avec les résultats de tâche n'entraîne pas l'envoi de cibles d'allocation.

Le moteur de flux de travaux utilise alors la propriété curam.custom.workflow.workresolver afin de déterminer l'implémentation de la fonction utilisée pour attribuer des tâches dans l'application. Cette fonction est ensuite appelée par le moteur de flux de travaux en lui transmettant la liste des cibles d'allocation comme indiquée par la stratégie d'allocation, ainsi que les détails de la tâche à allouer.

Après que le programme de résolution de travail a été appelé pour la tâche, le moteur de flux de travaux fait appel à la méthode `checkTaskAssignment` dans la classe `curam.core.sl.impl.TaskAssignmentChecker`. Cette fonction vérifie le statut d'affectation de la tâche (par exemple, pour s'assurer qu'elle a été affectée à au moins un utilisateur ou objet organisationnel (unité d'organisation, poste ou emploi), ou bien à une file d'attente). Si la tâche n'a pas été affectée, la propriété d'application curam.workflow.defaultworkqueue par défaut est examinée afin de déterminer ce qui a été spécifié en tant que file d'attente de travaux par défaut pour le flux de travaux. La tâche est alors affectée à cette file d'attente de travaux.

Si la tâche a été affectée à un utilisateur, et à un utilisateur seulement, après que le travail a été résolu, le système vérifie la valeur de la propriété d'application curam.workflow.automaticallyaddtasktousertasks. Cet indicateur contrôle si le système va automatiquement ajouter la tâche spécifiée qui a été traitée à la liste des tâches de cet utilisateur afin de lui permettre de travailler dessus. La valeur par défaut de la propriété est `NON`. Toutefois, si `OUI` a été défini, le système ajoutera automatiquement cette tâche à la liste Mes tâches de l'utilisateur dans sa Boîte de réception afin de lui permettre de travailler dessus.

Description des WDO (objet de données de flux de travaux) de contexte

L'objet de données de flux de travaux Context_Task est disponible pour les mappages de jeu de règles CER, classique et de fonction d'allocation. Cet objet de données de flux de travaux de contexte et son attribut ont déjà été décrits ci-dessus : (voir «Description des WDO (objet de données de flux de travaux) de contexte», à la page 68).

Associations d'objets métier

Les activités manuelles, et même le flux de travaux en général, effectuent des opérations sur des entités qui existent dans l'application. Pour cette raison, il peut être utile d'associer une tâche avec les entités connexes pour ce processus. Les associations d'objet métier fournissent essentiellement des liens entre une tâche et toute entité d'application d'intérêt pour ce processus. Les meilleurs exemples dans Cúram incluent les entités Cas et Rôle d'entreprise.

Métadonnées

```
<manual-activity id="1" category="AC1">
      ...
      <task>
        ...
      </task>
      <allocation-strategy type="target"
                           identifier="1"/>
      <event-wait>
        ...
      </event-wait>
      <biz-object-associations>
        <biz-object-association biz-object-type="BOT1">
          <wdo-attribute wdo-name="TaskCreateDetails"
                         name="caseID"/>
        </biz-object-association>
        <biz-object-association biz-object-type="BOT2">
          <wdo-attribute wdo-name=
            "PersonDetailsList[Context_Loop.loopCount]"
                         name="identifier"/>
        </biz-object-association>
      </biz-object-associations>
    </manual-activity>
```

biz-object-associations
Contient les détails de toutes les associations d'objet métier qui ont été spécifiées pour l'activité manuelle.

biz-object-association
Contient les détails d'une association d'objet métier qui a été spécifiée pour cette activité manuelle. Cet élément inclut le type d'objet métier et le mappage d'attribut d'objet de données de flux de travaux associé à ce type. Ce mappage d'attribut d'objet de données de flux de travaux représente l'identificateur unique de l'objet métier dans l'association (c'est-à-dire, pour une association d'objet métier de type Dossier, il s'agirait de l'identificateur unique du dossier en cours de liaison à la tâche).

biz-object-type
Détaille le type d'objet métier réel pour l'association d'objet métier de l'activité manuelle. Le type d'objet métier doit être sélectionné dans l'outil de définition de processus. Par ailleurs, il est extrait de la table de codes `BusinessObjectType`.

Validations

- Le type d'objet métier spécifié doit être un code valide contenu dans la table de codes `BusinessObjectType`.
- L'attribut d'objet de données de flux de travaux mappé au type d'objet métier d'une association d'objets métier d'activité manuelle doit être valide. Ce type d'attribut doit pouvoir être affecté à un type LONG parce que cela représente un mappage à un identificateur unique (par exemple, un identificateur d'observations ou un identificateur de participant).

- Si un élément indexé d'un objet de données de flux de travaux de liste (par exemple, PersonDetailsList[Context_Loop.loopCount]) est utilisé dans un mappage d'association d'objets métier, l'objet de données de flux de travaux doit être un objet de données de flux de travaux de liste et l'activité contenant le mappage doit être contenue dans une boucle.
- Si l'objet de données de flux de travaux `Context_Parallel` est utilisé dans un mappage d'association d'objets métier, l'activité contenant le mappage doit être une activité manuelle `Parallel`.
- Si un élément indexé de l'objet de données de flux de travaux de liste parallèle est utilisé dans un mappage d'association d'objets métier, l'activité contenant le mappage devra être une activité parallèle. (par exemple, [ParallelListWDO[Context_Parallel.occurrenceCount]). L'objet de données de flux de travaux en cours d'indexation par le Context_Parallel de flux de données de l'objet doit être l'objet de données de flux de la liste d'activités parallèle.

Code

API d'administration d'association d'objets métier

Un certain nombre de fonctions a été fourni sur la classe `BusinessObjectAssociationAdmin` afin de permettre la manipulation des associations d'objets métier. Pour plus de détails sur les fonctions disponibles, voir la spécification Javadoc associée à la classe `BusinessObjectAssociationAdmin`.

Informations d'exécution

Les associations d'objets métier n'ont pas d'impact fonctionnel sur l'exécution d'une activité manuelle. Le moteur de flux de travaux examine simplement les métadonnées et place un enregistrement sur l'entité `BizObjAssociation` pour chaque association d'objets métier spécifiée. Le type d'objet métier, la valeur du mappage d'attribut d'objet de données de flux de travaux d'attributs et l'identificateur de la tâche récemment créée associée à l'activité manuelle sont tous utilisés lors de la création de cet enregistrement.

Attente d'événement

Etant donné qu'une activité manuelle requiert que certaines actions soient effectuées par un utilisateur avant de pouvoir être terminée et que le processus puisse se poursuivre, il doit exister un moyen de notifier au moteur de flux de travaux que le travail requis a été effectué. Etant donné que cette sémantique est similaire à celle de l'activité d'attente d'événement, le mécanisme d'attente d'événement est réutilisé pour les activités manuelles.

Conditions préalables

- Les détails relatifs à une attente d'événement et à ses métadonnées associées (qui sont également applicables à une activité manuelle) se trouvent dans «Attente d'événement», à la page 49.

Description des WDO (objet de données de flux de travaux) de contexte

L'objet de données de flux de travaux Context_Task est disponible pour une utilisation dans les mappages d'entrée pour les fonctions d'échéance associées à l'attente d'événement d'une activité manuelle. Il est disponible pour les mappages d'entrée associés aux mappages d'entrée de règle CER, classique ou de fonction d'allocation. Il peut également être utilisé en tant que mappage pour les données de correspondance d'événement d'une attente d'événement spécifiée associée à une activité manuelle. Cet objet de données de flux de travaux de contexte et son

attribut ont déjà été décrits ci-dessus (voir «Description des WDO (objet de données de flux de travaux) de contexte», à la page 68).

Décision

Conditions préalables

- Les détails de base communs à tous les types d'activité pris en charge par le flux de travaux Cúram sont décrits dans «Activité de base», à la page 32 et sont applicables à l'activité de décision décrite ici.
- Il existe également des constructions de métadonnées de flux de travaux très fréquentes entre les activités manuelles et les activités de décision (de stratégie d'allocation, l'objet de la tâche, l'échéance de la tâche, etc). De plus amples informations sont disponibles sur «Manuel», à la page 61.

Présentation

Une exigence typique des processus métier est d'avoir un agent humain qui prenne des décisions qui ont des réponses simples. A titre d'exemple, une décision qui approuve ou rejette un dossier ou encore fournit certaines informations simples telles que l'âge du demandeur. Le recours aux activités manuelles pour demander des informations nécessite la disponibilité dans l'application d'un écran d'interface utilisateur différent pour chaque question. C'est lourd et étant donné que les définitions de processus peuvent changer au fil du temps, ces écrans d'interface utilisateur pourraient quelque peu être temporaires.

L'activité de décision est une spécialisation d'une activité manuelle qui gère une interface utilisateur pilotée par les métadonnées pour poser des questions simples. Les questions et réponses possibles sont dans les métadonnées d'activité. Ce qui permet à une interface utilisateur unique d'être utilisée pour un large éventail de questions. Deux types de questions sont actuellement pris en charge. Ce sont des questions de type choix multiple et des questions nécessitant une réponse qui peut être fournie dans une zone de l'interface utilisateur.

Détails de tâche

Tout comme un «Manuel», à la page 61, les activités de décision informeront les utilisateurs de leur obligation de faire un certain nombre de travaux et leur affectera une tâche en fonction de la stratégie d'allocation définie. La tâche génère automatiquement un lien vers une page interface utilisateur dans l'application qui regroupe les questions de décision à partir des métadonnées des questions d'activité de décision et réachemine le flux de travaux une fois que la réponse de décision a été fournie. Une activité de décision, par conséquent, ne peut avoir qu'une opération de tâche associée et ne nécessite la définition d' aucune page d'action pour cette action.

Outre l'opération de tâche, une activité de décision peut avoir entre zéro ou plusieurs actions secondaires qui lui sont associées. Les actions secondaires contiennent un lien vers une page qui peut fournir des informations supplémentaires pour aider l'utilisateur à répondre à la question de la décision.

Métadonnées

```
<decision-activity id="1">

    ...

      <allocation-strategy type="target" identifier="1" />
      <message>
        <message-text>
          <localized-text>
            <locale language="en">
            Decide the age of the user %1s for Case %2n.</locale>
          </localized-text>
        </message-text>
        <message-parameters>
          <wdo-attribute wdo-name="TaskCreateDetails"
            name="userName" />
          <wdo-attribute wdo-name=
            "CaseList[Context_Loop.loopCount]"
            name="identifier" />
        </message-parameters>
      </message>
      <decision-action>
        <message>
          <message-text>
            <localized-text>
              <locale language="en">
              Page d'accueil du participant pour le dossier %1n.
              </locale>
            </localized-text>
          </message-text>
          <message-parameters>
            <wdo-attribute wdo-name="TaskCreateDetails"
              name="concernRoleID" />
            <wdo-attribute wdo-name=
              "CaseList[Context_Loop.loopCount]"
              name="identifier" />
          </message-parameters>
        </message>
      </decision-action>
      <secondary-actions>
        <secondary-action page-id="Case_viewDetails">
          <message>
            <message-text>
              <localized-text>
                <locale language="en">View case details.</locale>
              </localized-text>
            </message-text>
          </message>
        </secondary-action>
        <secondary-action page-id="Case_viewUserDetails">
          <message>
            <message-text>
              <localized-text>
                <locale language="en">View details for user %1s.
                </locale>
              </localized-text>
            </message-text>
            <message-parameters>
              <wdo-attribute wdo-name=
                "ChildDependents[Context_Loop.loopCount]"
                name="userName" />
            </message-parameters>
          </message>
          <link-parameter name="userName">
            <wdo-attribute wdo-name="ChildDependents"
              name="childName" />
          </link-parameter>
          <multiple-occurring-action>
     <list-wdo-name>ChildDependents</list-wdo-name>
          </multiple-occurring-action>
        </secondary-action>
```

allocation-strategy
Cette section décrit la stratégie d'allocation utilisées pour déterminer l'utilisateur attribué à la tâche associée. Pour de plus amples informations sur les stratégies d'allocation, consultez «Stratégie d'allocation», à la page 68.

message
Cela représente le message objet paramétré de la tâche créée. Pour de plus amples informations sur les messages paramétrés, consultez «Manuel», à la page 61.

decision-action
Cela représente le message de texte d'action paramétré associé à la tâche. L'utilisateur clique sur ce texte d'action pour afficher l'écran de décision de l'interface utilisateur automatiquement généré à l'aide de la question pertinente.

échéance
Cette section décrit les détails relatifs à la date limite de l'activité de décision. Si une réponse n'a pas été fournie pour l'activité de décision au terme de l'échéance indiquée, alors la méthode de gestionnaire d'échéance associée est appelée. Pour de plus amples informations sur les échéances, consultez«Echéance», à la page 53

secondary-actions
Cette section décrit les actions secondaires facultatives pouvant être incluses dans l'activité de décision.

secondary-action
Une action secondaire contient un message paramétré et un lien de support d'informations paramétré pour aider l'utilisateur à répondre à la question de la décision. Pour plus d'informations sur les messages paramétrés et un lien de support d'informations paramétré, consultez «Métadonnées», à la page 62

page-id
Ceci représente l'identificateur de la page Cúram cible qui contient des informations supplémentaires liéds par l'action secondaire.

multiple-occurring-action
Ceci signifie que cette action secondaire représente une action récurrente. Cela signifie que si ces métadonnées sont définies pour une action secondaire, le moteur de flux de travaux crée un enregistrement d'action secondaire pour chaque élément dans l'objet de données de flux de travaux de liste défini comme action se produisant plusieurs fois, lorsque cette activité est exécutée.

Notez que lorsque l'action récurrente est définie pour une action de secondary, un attribut de l'objet de données de flux de travaux de liste associé doit être utilisé comme paramètre de lien pour l'action secondaire.

list-wdo-name
Nom de l'objet de données de flux de travaux de type liste à utiliser avec l'action se produisant plusieurs fois.

Validations

- Vous devez définir un sujet d'activité.

- Chaque attribut d'objet de données de flux de travaux mappé au sujet d'activité de décision doit être un attribut d'objet de données de flux de travaux valide.
- Si un élément indexé à partir d'un objet de données de flux de travaux de liste (. CaseList[Context_Loop.loopCount]) est utilisé en tant que paramètre de texte d'objet de décision, donc l'objet de données de flux de travaux doit être un objet de données de flux de travaux de liste et l'activité contenant le mappage doit être contenue dans une boucle.
- Si le flux de données de l'objet `Context_Parallel` est utilisé en tant que paramètre de texte d'objet de décision, l'activité contenant le mappage doit être un `Parallèle` de l'activité de décision.
- Si un élément indexé à partir de la liste de flux de données parallèles de l'objet est utilisé en tant que paramètre de texte d'objet de décision, l'activité contenant le mappage doit être une activité parallèle (. ParallelListWDO[Context_Parallel.occurrenceCount]). L'objet de données de flux de travaux en cours d'indexation par le Context_Parallel de flux de données de l'objet doit être l'objet de données de flux de la liste d'activités parallèle.
- Le nombre d'espaces réservés utilisés dans le texte de l'objet et le texte d'action doit être égal au nombre d'attributs d'objet de données de flux de travaux mappés(pour tous les environnements locaux).
- Si un élément indexé à partir d'un objet de données de flux de travaux de liste (. CaseList[Context_Loop.loopCount]) est utilisé en tant que paramètre de texte d'objet de décision, donc l'objet de données de flux de travaux doit être un objet de données de flux de travaux de liste et l'activité contenant le mappage doit être contenue dans une boucle.
- Si le flux de données de l'objet `Context_Parallel` est utilisé en tant que paramètre de texte d'objet de décision, l'activité contenant le mappage doit être un `Parallèle` de l'activité de décision.
- Si un élément indexé à partir de l'objet de données du flux de travaux de la liste parallèle est utilisé en tant que paramètre de texte d'action de décision, donc l'activité contenant le mappage doit être une activité parallèle (par exemple,ParallelListWDO[Context_Parallel.occurrenceCount]). L'objet de données de flux de travaux en cours d'indexation par le Context_Parallel de flux de données de l'objet doit être l'objet de données de flux de la liste d'activités parallèle.
- Vous devez définir une stratégie d'allocation.
- La cible d'allocation, la fonction ou le jeu de règles CER ou classique spécifié comme stratégie d'allocation doit être valide. Si le type d'allocation est une fonction, il doit s'agir d'une méthode métier Cúram valide qui doit exister dans le chemin d'accès aux classes de l'application. Si le type d'allocation est une règle classique ou CER, il doit s'agir d'un jeu de règles valide.
- Le gestionnaire d'échéance facultative, si indiquée, doit être une méthode métier Cúram valide.
- Tous les mappages d'entrée du gestionnaire d'échéance doivent être valides. Cela signifie que toutes les zones de paramètre d'entrée requises par la méthode spécifiée sont mappées à des attributs d'objet de données de flux de travaux valides de type correct.
- Chaque action secondaire doit disposer d'un lien de page spécifié, qui ne peut contenir d'espaces.
- Chaque action secondaire doit posséder un message spécifié.
- Le texte du message d'action secondaire doit contenir un nombre de caractères égal au nombre de paramètres de message spécifié.

- Les paramètres de message d'action secondaires doivent être mappés à des attributs d'objet de données de flux de travaux valide de type correct.
- Les paramètres de lien de pagination d'action secondaire doivent être mappés aux attributs d'objet de données de flux de travaux.
- Si un élément indexé à partir d'un objet de données de flux de travaux de liste (par exemple, ChildDependents[Context_Loop.loopCount])est utilisé dans les mappages de textes d'action secondaire ou de lien d'action secondaire, donc l'objet de données de flux de travaux doit être un objet de données de flux de travaux de liste et l'activité contenant le mappage doit être contenue dans la boucle.
- Si l'objet de données du flux de travaux `Context_Parallel` est utilisé dans les mappages du texte d'action secondaire ou lien d'action secondaire, donc l'activité contenant le mappage doit être une activité de décision `Parallel`.
- Si un élément est indexé à partir d'un objet de données de flux de travaux de la liste parallèle dans les mappages de paramètres d'action secondaire ou de lien d'action secondaire, donc l'activité contenant les mappages doivent être une activité parallèle (par exemple, ParallelListWDO[Context_Parallel.occurrenceCount]). L'objet de données de flux de travaux en cours d'indexation par le Context_Parallel de flux de données de l'objet doit être l'objet de données de flux de la liste d'activités parallèle.
- L'objet de données de flux de travaux spécifié devant être utilisé dans l'action récurrente doit être un objet de données de flux de travaux valide dans le cadre de la définition du processus de flux de travaux le contenant. Il doit également être de type Liste.
- Au moins un attribut de l'objet de données de flux de travaux de la liste des actions récurrentes doit être utilisé dans les paramètres de lien spécifiés pour une action récurrente.

Informations d'exécution

Lorsqu'une activité de décision est exécutée, le moteur de flux de travaux génère la tâche associée. Une image instantanée des données d'objet de données de flux de travaux requis pour le sujet d'activité de décision et les paramètres de texte d'action, et tout texte de message d'action secondaire et paramètres de liaison sont pris et stockés. La stratégie d'allocation associée à l'activité de décision est appelée pour déterminer l'utilisateur(s) qui sera affecté à la tâche de décision. Le moteur de flux de travaux crée également une attente d'événement pour l'événement `DECISION.MADE` à l'aide de l'identificateur de tâche associée sous forme de données de correspondance d'événement. Le flux de travaux est alors suspendu à ce stade dans l'attente de la mise au premier plan de cet événement qui indiquera le résultat de la décision prise.

Détails de la question

L'activité de décision prend actuellement en charge les questions à choix multiple et les questions en texte libre sous forme de formats de question. La page de décisions automatiquement générée examine le format de question requis et génère la question pertinente à partir des métadonnées question dès que l'utilisateur clique sur l'action associée à la tâche.

Métadonnées

À choix multiples :

```
<decision-activity id="1">

    ...

      <question>
        <message>
          <message-text>
            <localized-text>
              <locale language="en">
              Is the claimant, %1s, for Case %2n, over 18?
              </locale>
            </localized-text>
          </message-text>
          <message-parameters>
            <wdo-attribute wdo-name="Participant"
              name="userName" />
            <wdo-attribute wdo-name=
              "CaseList[Context_Loop.loopCount]"
              name="identifier" />
          </message-parameters>
        </message>
        <answers multiple-selection="false">
          <answer name="yesAnswer">
            <answer-text>
              <localized-text>
                <locale language="fr">Oui</locale>
              </localized-text>
            </answer-text>
            <choice-output-mapping>
              <wdo-attribute wdo-name="DecisionResult"
                             name="ageBracket" />
              <selected-value>18-65/selected-value>
              <not-selected-value>0-17</not-selected-value>
            </choice-output-mapping>
          </answer>
          <answer name="noAnswer">
            <answer-text>
              <localized-text>
                <locale language="fr">Non</locale>
              </localized-text>
            </answer-text>
            <choice-output-mapping>
              <wdo-attribute wdo-name="DecisionResult"
                             name="ageBracket" />
              <selected-value>0-17</selected-value>
              <not-selected-value>18-65</not-selected-value>
            </choice-output-mapping>
          </answer>
        </answers>
      </question>

    ...

    </decision-activity>
```

question

Ceci représente la question associée à l' activité de décision, qui, pour une question de texte libre contient les métadonnées décrites ci-après.

message

Cela représente le texte paramétré de la question à poser pour tous les environnements locaux.

réponses
Ceci représente une liste de réponses parmi lesquelles l'utilisateur peut choisir pour la question à choix multiple.

multiple-selection
Ceci représente un indicateur qui indique si l'utilisateur peut sélectionner plusieurs réponses à partir de celles fournies, ou s'il n'est possible de ne sélectionner qu'une seule.

répondre
Ceci représente une réponse que l'utilisateur peut sélectionner. Il doit exister au moins une réponse fournie pour une question à choix multiples.

name Cela représente le nom de la réponse. Une fois que l'utilisateur sélectionne une ou des réponses, les noms des réponses sélectionnés sont transmis au moteur de flux de travaux et le processus progresse. A mesure que le moteur traite ces réponses similaires aux attributs d'objet de données de flux, les noms de réponse doivent être des identificateurs Java valides.

answer-text
Cela représente le texte de réponse que l'utilisateur peut sélectionner pour tous les environnements locaux.

choice-output-mapping
Cette balise contient les métadonnées qui décrivent comment la sortie d'une réponse à choix multiple sera conservée.

attribut wdo
Le nom de l'attribut d'objet de données de flux utilisé pour conserver la valeur de la réponse à choix multiples.

valeur-sélectionnée
Si elle est spécifiée, la valeur de cet élément sera conservée à l'attribut d'objet de données de flux de travaux si cette réponse a été sélectionnée par l'utilisateur. Si l'attribut d'objet de données de flux de travaux est de type booléen, cette valeur ne devra pas être spécifiée. Une valeur par défaut `true` lui sera affectée.

valeur-non sélectionnée.
Si elle est spécifiée, la valeur de cet élément sera conservée à l'attribut d'objet de données de flux de travaux si cette réponse a été sélectionnée par l'utilisateur. Si l'attribut d'objet de données de flux de travaux est de type booléen, cette valeur ne devra pas être spécifiée. Une valeur par défaut `false` lui sera affectée.

Texte libre :

```
<decision-activity id="1">

    ...

      <question>
        <message>
          <message-text>
            <localized-text>;
              <locale language="en">
              What is the age of the claimant, %1s?
              </locale>
            </localized-text>
          </message-text>
          <message-parameters>
            <wdo-attribute wdo-name="Participant"
              name="userName" />
          </message-parameters>
        </message>
        <free-text type="INT32">
          <wdo-attribute wdo-name="DecisionResult"
            name="ageOfClaimant" />
        </free-text>
      </question>

    ...

    </decision-activity>
```

question
Ceci représente la question associée à cette activité de décision, qui, pour une question de texte libre contient les métadonnées décrites ci-après.

message
Cela représente le texte paramétré de la question à poser pour tous les environnements locaux.

free-text
Elle contient les détails de la réponse de texte libre que l'utilisateur doit fournir.

type Cela représente le type de données requis de la réponse de texte libre qui doit être fourni.

attribut wdo
Ceci représente l'attribut d'objet de données de flux de travaux qui mappe la réponse de texte libre dans le moteur de flux de travaux.

Validations

- Le format de réponse et le texte de question doivent être spécifiés pour une activité de décision.
- Le nombre d'espaces réservés utilisés dans le texte de question doit être égal au nombre d'attributs d'objet de données de flux de travaux mappés(pour tous les environnements locaux).
- Si un élément indexé à partir d'un objet de données de flux de travaux de liste (. CaseList[Context_Loop.loopCount]) est utilisé en tant que paramètre de texte d'objet de décision, donc l'objet de données de flux de travaux doit être un objet de données de flux de travaux de liste et l'activité contenant le mappage doit être contenue dans une boucle.

- Si l'objet de données de flux de travaux `Context_Parallel` est utilisé en tant que paramètre de texte de question, donc l'activité contenant le mappage doit être une activité de décision `Parallel`.
- Si un élément indexé à partir de l'objet de données de flux de travaux est utilisé en tant que paramètre de texte de question, donc l'activité contenant le mappage doit être une activité parallèle (par exemple, ParallelListWDO[Context_Parallel.occurrenceCount]). L'objet de données de flux de travaux en cours d'indexation par le Context_Parallel de flux de données de l'objet doit être l'objet de données de flux de la liste d'activités parallèle.
- S'agissant d'une question en format de réponse formulaire en forme libre, le type de données de réponse doit être spécifié et l'attribut de l'objet de flux de travaux doit être valide et correspondre au type de données de réponse. L'attribut de l'objet de données de flux de travaux mappé ne peut être un attribut d'objet de données de flux de travaux.
- S'agissant d' une question avec format de réponse de la liste, une option de réponse est au moins répertoriée. Tous les noms de réponse doivent être des noms d'attributs Java valides.

Informations d'exécution

Lorsqu'une réponse pour une question d'activité de décision a été fournie, le `DECISION.MADE` de l'événement est soulevée avec l'identificateur de tâche de la tâche d'activité de décision utilisée en tant que données de correspondance d'événement. Le gestionnaire d'événements de flux de travaux va traiter cet événement, ce qui fait avancer le processus de flux de travaux.

Si la réponse fournie est une réponse en texte libre, elle sera mappée à l'attribut d'objet de données de flux de travaux spécifié pour une utilisation ultérieure dans le processus, si nécessaire.

Description des WDO (objet de données de flux de travaux) de contexte

L'objet de données de flux de Context_Decision est disponible pour une utilisation dans un élément de données ou de la condition de fonction (voir «Conditions», à la page 112) pour une transition à partir d'une activité de décision. Les attributs disponibles varient sur le format de réponse définie pour l'activité.

Free Text Answer

Si le format de réponse est une réponse en texte libre, l'attribut disponible est:

Context_Decision.value

La valeur de la réponse de texte libre fournie. Cet attribut peut être utilisé dans des conditions de transition et peut être mappé à un attribut d'objet de données de flux de travaux spécifié.

Réponse à choix multiple

Dans ce cas, l'objet de données de flux de Context_Decision sera remplie avec les attributs pour chacun des réponses disponibles, chacune étant de type booléen. Indique si cette réponse avait été sélectionnée ou non. Dans le fragment de métadonnées de réponse à choix multiple ci-dessus, («À choix multiples», à la page 85, si l'utilisateur a sélectionné la première réponse (Oui), cela aura une répercussion sur l'attribut d'objet de données de flux de travaux Context_Decision suivant en cours de configuration sur `true`:

Context_Decision.yesAnswer

Ceci représente une valeur booléenne indiquant si la réponse oui

pour la question avait été cochée. Cela peut uniquement être utilisé dans les conditions de transition de l'activité de décision.

Autrement, si l'utilisateur a coché la seconde réponse (Non), ceci aura une répercussion sur l'attribut d'objet dedonnées de flux de travaux Context_Decision suivant en cours de configuration`true`:

Context_Decision.noAnswer
Ceci représente une valeur booléenne indiquant si la réponse non pour la question avait été cochée. Il n'est possible de ne l'utiliser que dans des conditions de transition de l'activité de décision.

Flux secondaire

Conditions préalables

- Les détails de base communs à tous les types d'activité pris en charge par les flux de travaux Cúram sont décrits dans «Activité de base», à la page 32 et s'appliquent à l'activité de flux secondaire décrite ici.

Présentation

Lors de sa conception, un processus métier complexe peut devenir trop volumineux pour être géré d'un seul bloc en tant que définition de processus. Une activité de flux secondaire permet de lancer une autre définition de processus, dans le cadre d'un autre processus.

La conception de définitions de processus sous la forme d'un ensemble de flux secondaires quels que soient les éventuels problèmes de taille peut s'avérer raisonnable. Ceci permettrait de modifier certaines sections du processus métier sans affecter les autres. Les processus du flux secondaires pourraient également être réutilisés par les clients au cours de la construction de leurs propres définitions de processus d'ordre supérieur.

Processus de flux secondaire

Pour lancer un processus en tant que flux secondaire, une activité de flux secondaire doit identifier par son nom le processus qui sera lancé. Comme pour les autres mécanismes de lancement de processus, c'est la version la plus récente du processus qui sera lancée.

Les flux secondaires peuvent être lancés de façon *synchrone*. Cela signifie que la branche du flux de travaux parent contenant l'activité de flux secondaire qui a démarré le processus de flux secondaire attend la fin de ce processus de flux secondaire pour pouvoir continuer.

Un flux secondaire peut également être lancé de façon *asynchrone*. Cela signifie qu'une fois que l'activité de flux secondaire démarre le processus de flux secondaire, la branche contenant cette activité de flux secondaire passe immédiatement au résultat du processus de flux secondaire, ce qui n'a aucun effet sur le processus parent.

Métadonnées

```
<subflow-activity id="1">
    ...

    <subflow workflow-process="ApproveCase" synchronous="true"/>

    ...
  </subflow-activity>
```

subflow

workflow-process
Nom du processus de flux de travaux à démarrer lorsque l'activité est exécutée. Les noms de processus sont sensibles à la casse et le nom du processus spécifié ici doit correspondre exactement à celui du processus à démarrer comme flux secondaire.

synchronous
Indicateur qui détermine si le flux secondaire doit être exécuté de manière synchrone ou non (voir : «Processus de flux secondaire», à la page 89) par rapport à son processus parent.

Validations

- Un processus de flux de travaux doit être spécifié pour l'activité de flux secondaire.
- Le processus de flux de travaux spécifié comme flux secondaire doit posséder au moins une version publiée.

Mappages des entrées

Les données sont fournies au flux secondaire lorsqu'elles sont adoptées par les objets de données de flux de travaux du processus parent. L'activité du flux secondaire définit le mappage entre les objets de données de flux de travaux du processus parent et les données d'adoption des flux secondaires.

Métadonnées

```
<subflow-activity id="1">
    ...

    <input-mappings>
      <mapping>
        <source-attribute wdo-name="ManintainCase"
                          name="caseID" />
        <target-attribute wdo-name="ApproveCase"
                          name="caseID" />
      </mapping>
      <mapping>
        <source-attribute wdo-name="MaintainCase"
                          name="concernRoleID" />
        <target-attribute wdo-name="ApproveCase"
                          name="concernRoleID" />
      </mapping>
      <mapping>
        <source-attribute wdo-name=
                     "PersonDetailsList[Context_Loop.loopCount]"
                          name="identifier" />
        <target-attribute wdo-name="PersonDetails"
                          name="identifier" />
      </mapping>
      <mapping>
        <source-attribute wdo-name="ChildDetailsList"
                          name="identifier" />
        <target-attribute wdo-name="ClaimantDependentList"
                          name="identifier" />
      </mapping>
    </input-mappings>
  </subflow-activity>
```

input-mappings

Indique la méthode de mappage des données du processus en cours d'exécution à un sous-processus sous la forme de données d'adoption lorsque le sous-processus est démarré. Le processus spécifié en tant que flux secondaire peut ne contenir aucun attribut d'objet de données de flux de travaux marqué comme "requis pour l'adoption", auquel cas aucun mappage d'entrée n'est requis.

mapping

L'élément `mapping` représente les données qui seront envoyées par un attribut d'objet de données de flux de travaux à un attribut du processus en cours d'adoption en tant que flux secondaire. Si une liste de données est requise pour adopter le processus de flux secondaire, les attributs provenant des objets de données de flux de travaux de type liste peuvent être utilisés dans ce but. Le nombre de mappages spécifié est régi par le nombre d'attributs marqués comme "requis pour l'adoption" dans le processus de flux secondaire, étant donné que tous ces attributs doivent être remplis au démarrage du processus.

source-attribute

Représente un attribut d'objet de données de flux de travaux provenant du processus parent à utiliser pour remplir l'attribut associé dans le flux secondaire au moment de son adoption.

target-attribute

Représente un attribut d'objet de données de flux de travaux provenant du flux secondaire à remplir à l'aide de données provenant de l'attribut associé dans le processus parent au moment de l'adoption.

source/target-attribute

wdo-name
Représente le nom d'un objet de données de flux de travaux Cúram, comme décrit dans le «Objets de données de flux de travaux», à la page 17.

name Représente le nom d'un attribut d'objet de données de flux de travaux Cúram, comme décrit dans le «Objets de données de flux de travaux», à la page 17.

Validations

- Chaque attribut d'objet de données de flux de travaux marqué comme *obligatoire pour l'adoption* dans le flux secondaire doit être spécifié dans les mappages d'entrée. Si aucun attribut d'objet de données de flux de travaux n'a été marqué comme "obligatoire pour l'adoption" dans le processus de flux secondaire, aucun mappage d'entrée ne doit être spécifié.
- Le type de données de l'attribut d'objet de données de flux de travaux spécifié par la balise `target-attribute` doit correspondre ou être attribuable à l'attribut spécifié par la balise `source-attribute`.
- Si un élément indexé provenant d'un objet de données de flux de travaux de type liste (par ex. PersonDetailsList[Context_Loop.loopCount]) est spécifié dans la balise `source-attribute` du mappage d'entrée du flux secondaire, cet objet de données de flux de travaux doit être un objet de données de flux de travaux de type liste et l'activité de flux secondaire contenant le mappage d'entrée doit être incluse dans une boucle. Le type de données de l'attribut d'objet de données de flux de travaux spécifié par la balise `target-attribute` doit correspondre ou être attribuable à l'attribut spécifié par la balise `source-attribute`.
- Si le mappage d'entrée de flux secondaire spécifié utilise un objet de données de flux de travaux de type liste, les attributs d'objet de données de flux de travaux pour les balises `source-attribute` parente et `target-attribute` du processus de flux secondaire doivent être des objets de données de flux de travaux de type liste.

Mappages de sortie

Les mappages de sortie s'appliquent uniquement aux activités des flux secondaires *synchrones*, étant donné que les flux secondaires asynchrones peuvent continuer leur exécution sans avoir à terminer l'activité. Les données sont fournies au processus parent par l'activité de flux secondaire une fois que celle-ci est terminée. L'activité de flux secondaire définit le mappage entre un attribut d'objet de données de flux de travaux du flux secondaire et l'attribut d'objet de données de flux de travaux du processus parent.

Métadonnées

```
<subflow-activity id="1">
  ...

    <output-mappings>
      <mapping>
        <source-attribute wdo-name="SubflowCaseWDO"
                          name="participantName" />
        <target-attribute wdo-name="CaseWDO"
                          name="participantName" />
      </mapping>
      <mapping>
        <source-attribute wdo-name="SubflowChildDetailsList"
                          name="identifier" />
        <target-attribute wdo-name="ChildDetailsList"
                          name="identifier" />
      </mapping>
    </output-mappings>

  ...
  </subflow-activity>
```

output-mappings
Indique la méthode de mappage des données du sous-processus appelé au processus parent lorsque le sous-processus est terminé. Le processus spécifié en tant que flux secondaire peut ne contenir aucun mappage de sortie défini, auquel cas le flux secondaire se termine normalement.

mapping
Représente les données qui seront envoyées par un attribut d'objet de données de flux de travaux de flux secondaire à un attribut du processus parent. Si une liste de données est envoyée par le processus de flux secondaire au processus parent, les attributs provenant des objets de données de flux de travaux de type liste peuvent être utilisés dans ce but. Le nombre de mappages spécifié est régi par le nombre de mappages de sortie spécifié.

source-attribute
Représente un attribut d'objet de données de flux de travaux provenant du processus de flux secondaire, utilisé pour remplir l'attribut associé dans le processus parent dès qu'il se termine.

target-attribute
Représente un attribut d'objet de données de flux de travaux provenant du parent à remplir à l'aide de données provenant de l'attribut associé dans le processus de flux secondaire lorsqu'il se termine.

source/target-attribute

wdo-name
Représente le nom d'un objet de données de flux de travaux Cúram, comme décrit dans le «Objets de données de flux de travaux», à la page 17.

name Représente le nom d'un attribut d'objet de données de flux de travaux Cúram, comme décrit dans le «Objets de données de flux de travaux», à la page 17.

Validations

- Les attributs d'objet de données de flux de travaux `target-attribute` parent et `source-attribute` du flux secondaire utilisés dans le mappage de sortie du flux secondaire doivent être valides dans le contexte de la définition de processus qui les contient.

- Le type de données de l'attribut d'objet de données de flux de travaux spécifié par la balise `target-attribute` parente doit correspondre ou être attribuable à l'attribut spécifié par la balise `source-attribute` du flux secondaire.
- Si le mappage de sortie de flux secondaire spécifié utilise un objet de données de flux de travaux de type liste, les attributs d'objet de données de flux de travaux mappés pour les balises `target-attribute` parente et `source-attribute` du processus de flux secondaire doivent être de type liste.

Début et fin de boucle

Conditions préalables

- Les détails de base communs à tous les types d'activité pris en charge par le flux de travaux Cúram sont décrits dans «Activité de base», à la page 32 et sont applicables aux activités boucle début/fin ici décrites.

Présentation

De nombreux processus métier doivent se *répéter* jusqu'à ce qu'une condition soit remplie. Dans Cúram, cela est mis en oeuvre à l'aide des activités début de boucle (loop-begin) et fin de boucle (loop-end). Toutes les activités qui se trouvent entre une boucle de début (loop-begin) et l'activité de fin de boucle (loop-end) correspondante sont répétées jusqu'à la fin de la boucle.

Dans une définition de processus, les activités de début de boucle (loop-begin) et de fin de boucle (loop-end) arrivent par paires et les métadonnées permettent à chaque boucle de début de connaître la fin de boucle correspondante et vice versa. Pour ajouter une séquence d'activités à une boucle, une transition est créée à partir de l'activité de début de boucle (loop-begin) vers la première activité à répéter. Les activités suivantes dans la séquence sont liées à l'aide des transitions, comme tel serait normalement le cas en dehors d'une boucle. Toutefois, la dernière activité dans la séquence a une transition vers l'activité de fin de boucle. Une impulsion commune consiste à ajouter également une transition de l'activité de fin de boucle au démarrage pour créer le cycle. Cependant, cette action est erronée et donne lieu à une définition de processus qui n'est pas valide.

Une boucle doit aussi indiquer les critères qu'elle va utiliser pour déterminer s'il convient d'arrêter ou non. Pour prendre en charge cela, une boucle dans le flux de travaux Cúram dispose d'une condition de sortie de boucle (loop-exit).

Des boucles peuvent en contenir d'autres tant qu'elles sont totalement imbriquées les unes les autres. Cela permet de s'assurer que les boucles ainsi que la définition de processus restent une structure de bloc valide, comme requis par le moteur de flux de travaux Cúram (voir «Structure d'un flux de travaux», à la page 118).

Types de boucle

En plus de la condition de sortie de boucle (loop-exit), une boucle indique également si la condition doit être testée avant l'exécution de la boucle (boucle while) ou à la fin de l'exécution d'une boucle (boucle do while). Une boucle while ne peut jamais exécuter les activités dans la boucle et passer à l'activité qui suit la boucle si la condition de sortie est remplie au début de la boucle, alors qu'une boucle do while exécutera les activités dans la boucle au moins une fois.

Métadonnées

Activité de début de boucle

```
<loop-begin-activity id="1">
    ...

    <loop-type name="do-while"/>

    ...

    <condition>
      <expression id="1" data-item-lhs="Context_Loop.loopCount"
        operation="&lt;" data-item-rhs="UserAccountWDO.size()"/>
    </condition>

    <block-endpoint-ref activity-id="5"/>

  </loop-begin-activity>
```

loop-type

La balise `loop-type` indique la méthode d'exécution de la boucle, comme décrit dans «Types de boucle», à la page 94. Les deux seules valeurs admises pour l'attribut `name` sont `while` et `do-while`.

condition

La balise `condition` indique la condition qui sera évaluée en se basant sur les valeurs de l'objet de données de flux de travaux (voir «Objets de données de flux de travaux», à la page 17). Lorsque des objets de données de flux de travaux de type liste sont présents dans le flux de travaux, deux attributs qui ne font pas partie de ces métadonnées d'objet de données de flux de travaux sont mis à disposition lors de la création d'une expression de condition de boucle à l'aide d'un objet de données de flux de travaux de type liste. Ces attributs sont les suivants :

- *size()* : a pour résultat un nombre (de type entier) afin d'indiquer le nombre d'éléments présents dans la liste.
- *isEmpty()* : a pour résultat un indicateur booléen afin d'indiquer si la liste contient des éléments ou non.

Les métadonnées de condition réelles sont utilisées à d'autres endroits des métadonnées de la définition de processus et sont décrites dans le «Conditions», à la page 112.

block-endpoint-ref

Dans ce contexte, l'élément `block-endpoint-ref` permet à l'élément `loop-begin-activity` de reconnaître son élément `loop-end-activity` associé. Ces informations s'avèrent utiles au moteur de flux de travaux lors de l'exécution de la boucle. Par exemple, lorsque la condition de sortie d'une boucle while a pour résultat `true` avant que la boucle ne s'exécute, l'élément `block-endpoint-ref` indique au moteur de flux de travaux à quelle activité il doit directement passer et continue l'exécution du processus.

Activité de fin de boucle

```
<loop-end-activity id="3">
    ...

    <block-endpoint-ref activity-id="1"/>

  </loop-end-activity>
```

block-endpoint-ref

Dans ce contexte, l'élément `block-endpoint-ref` permet à l'élément `loop-end-activity` de reconnaître son élément `loop-begin-activity` associé. Ces informations s'avèrent utiles au moteur de flux de travaux lors de l'exécution de la boucle. Par exemple, si la condition de sortie a pour résultat false suite à l'exécution d'une boucle, l'élément `block-endpoint-ref` indique au moteur de flux de travaux à quelle activité il doit directement passer afin de débuter une autre itération de la boucle.

Informations d'exécution

Toute activité au sein d'une boucle est censée s'exécuter plusieurs fois au cours de l'exécution d'une instance de processus. Pour empêcher les itérations suivantes d'endommager les données de l'instance de processus pour l'activité, chaque instance d'activité est associée à une itération spécifique de sorte à pouvoir être exclusivement identifiée par le moteur de flux de travaux quel que soit le nombre d'exécutions de la boucle.

Description des WDO (objet de données de flux de travaux) de contexte

L'objet de données de flux de travaux Context_Loop est disponible dans les situations suivantes :

- Lors de la création de la condition de fin de boucle associée à une activité de début de boucle.
- Lors de la création des conditions de transition sortantes à partir d'une activité de début de boucle ou de n'importe quelle activité contenue dans une boucle (voir «Conditions», à la page 112).
- Lors de la création des mappages d'entrée pour n'importe quelle activité automatique ou activité de flux secondaire au sein d'une boucle.
- Lors de la création des mappages d'entrée pour n'importe quelle fonction de stratégie d'allocation ou fonction de gestionnaire d'échéance présente dans une activité au sein d'une boucle.
- Lors de la spécification d'un paramètre texte d'objet pour une activité manuelle ou une activité de décision contenue dans une boucle, ou pour une notification connectée à une activité contenue dans une boucle.
- Lors de la spécification de paramètres texte d'action pour une activité manuelle ou une activité de décision contenue dans une boucle, ou pour une notification connectée à une activité contenue dans une boucle.
- Lors de la spécification de l'identificateur d'une association d'objet métier pour une activité manuelle contenue dans une boucle.
- Lors de la spécification d'un paramètre texte de question pour une question ouverte ou une question à choix multiple pour une activité de décision contenue dans une boucle.
- Lors de la spécification d'un paramètre texte de corps pour une notification connectée à une activité contenue dans une boucle.

Les attributs de l'objet de données de flux de travaux Context_Loop disponibles sont les suivants :

Context_Loop.loopCount
Nombre d'itérations d'une boucle.

Parallel

Conditions préalables

- Les détails de base communs à tous les types d'activité pris en charge par le flux de travaux Cúram sont décrits dans «Activité de base», à la page 32 et sont applicables à l'activité parallèle décrite ici.
- Etant donné que les activités parallèles encapsulent les activités existantes dans une définition de processus de flux de travaux, les métadonnées décrites dans «Manuel», à la page 61 et «Décision», à la page 80 sont également pertinentes pour l'activité parallèle décrite ici.

Présentation

Dans les processus métier, il peut être nécessaire d'envoyer plusieurs tâches à différents agents humains en même temps afin d'accélérer la progression du processus global. Lorsque le nombre de chemins parallèles est connu au moment du développement, ceci est facilement effectué à l'aide d'un fractionnement. Toutefois, dans certains cas, le nombre de chemins ne sera pas connu jusqu'à l'exécution. Ces situations peuvent être modélisées à l'aide des activités parallèles.

Une activité parallèle fonctionne comme un encapsuleur autour des activités existantes. L'utilisation de l'une de ces nouvelles activités au moment de l'exécution permet d'exécuter en parallèle plusieurs instances d'activités encapsulées. Jusqu'à présent, les seuls types d'activités encapsulées pris en charge sont les activités manuelles («Manuel», à la page 61) et les activités de décision («Décision», à la page 80). Ainsi, l'exécution d'une activité parallèle correspond actuellement à la création et à l'allocation de plusieurs tâches en parallèle.

Métadonnées

Une activité parallèle doit spécifier le type d'activité qu'elle encapsule. Un objet de données de flux de travaux de type liste doit également être associé à l'activité parallèle. Le nombre d'éléments présents dans cet objet de données de flux de travaux de type liste détermine par la suite le nombre d'instances de cette activité encapsulée qui seront créées par le moteur de flux de travaux au cours de l'exécution.

Métadonnées génériques pour une activité parallèle

```
<parallel-activity id="1" category="AC1">
      <list-wdo-name>EmployerDetailsListWDO</list-wdo-name>
      <manual-activity>
        <name>
          <localized-text>
            <locale language="en">
            CheckEmployerDetailsTasks</locale>
          </localized-text>
        </name>
        ..........
      </manual-activity>
    </parallel-activity>


    ou ........

    <parallel-activity id="1" category="AC1">
      <list-wdo-name>ChildDetailsListWDO</list-wdo-name>
      <decision-activity>
        <name>
          <localized-text>
            <locale language="fr">ValidateChildDetails</locale>
          </localized-text>
        </name>
        ..........
      </decision-activity>
    </parallel-activity>
```

manual-activity/decision-activity

Reflète le type d'activité encapsulée par l'activité parallèle. Deux types d'activité encapsulée sont actuellement pris en charge : «Manuel», à la page 61 et «Décision», à la page 80. Les types d'activité pouvant être encapsulées par une activité parallèle sont visibles dans la table de codes `ParallelActivityType`.

list-wdo-name

Chaque activité parallèle doit posséder un objet de données de flux de travaux de type liste associé. Le nombre d'instances de l'activité encapsulée créées au cours de l'exécution est déterminé par le nombre d'éléments présents dans cet objet de données de flux de travaux de type liste.

Métadonnées pour une activité manuelle parallèle

L'exemple ci-après illustre les métadonnées associées à l'activité encapsulée de type `Manuelle`. Ces métadonnées sont *exactement* identiques celles d'une activité manuelle, décrites dans le «Manuel», à la page 61. Elles ne seront donc pas décrites à nouveau ici. Les validations appartenant aux mappages d'activité manuelle parallèle sont également décrites dans le «Manuel», à la page 61. L'objet de données de flux de travaux `Context_Parallel` et un élément indexé provenant de l'objet de données de flux de travaux de type liste de l'activité parallèle peuvent être utilisés dans tous les mappages disponibles pour une activité manuelle parallèle. Voici plusieurs exemples de syntaxe pour ces métadonnées :

```
<parallel-activity id="1" category="AC1">
      <list-wdo-name>EmployerDetailsListWDO</list-wdo-name>
      <manual-activity>
        ...
        <task>
          <message>
            <message-text>
              <localized-text>
                <locale language="fr">Vérifiez les détails
                de l'employeur %1s. Il s'agit de l'employeur numéro : %1n.
                </locale>
              </localized-text>
            </message-text>
            <message-parameters>
              <wdo-attribute
                wdo-name=
    "EmployerDetailsListWDO[Context_Parallel.occurrenceCount]"
                name="fullName" />
              <wdo-attribute
                wdo-name=
                "Context_Parallel" name="occurrenceCount" />
            </message-parameters>
          </message>
          ...
        </task>
        ...
        <event-wait wait-on-all-events="false">
          <events>
            <event identifier="1" event-class="EMPLOYER"
                                  event-type="DETAILSCHECKED">
              <event-match-attribute wdo-name=
      "EmployerDetailsListWDO[Context_Parallel.occurrenceCount]"
                                     name="identifier" />
            </event>
          </events>
        </event-wait>
        <biz-object-associations>
          <biz-object-association biz-object-type="BOT2">
            <wdo-attribute
            wdo-name=
       "EmployerDetailsListWDO[Context_Parallel.occurrenceCount]"
            name="identifier" />
          </biz-object-association>
        </biz-object-associations>
      </manual-activity>
    </parallel-activity>
```

Métadonnées pour une activité de décision parallèle

L'exemple ci-après illustre les métadonnées associées à l'activité encapsulée de type `Décision`. Ces métadonnées sont *exactement* identiques celles d'une activité de décision, décrites dans le «Décision», à la page 80. Elles ne seront donc pas décrites à nouveau ici. Les validations appartenant aux mappages d'activité de décision parallèle sont également décrites dans le «Décision», à la page 80. L'objet de données de flux de travaux `Context_Parallel` et un élément indexé provenant de l'objet de données de flux de travaux de type liste de l'activité parallèle peuvent être utilisés dans tous les mappages disponibles pour une activité de décision parallèle. Voici plusieurs exemples de syntaxe pour ces métadonnées :

```
<parallel-activity id="1" category="AC1">
      <list-wdo-name>ChildDetailsListWDO</list-wdo-name>
      <decision-activity>
        ...
        <message>
          <message-text>
            <localized-text>
              <locale language="fr">Les détails de cette tâche
              pour l'enfant %1s doivent être validés. Il s'agit de l'enfant
              numéro : %1n.
              </locale>
            </localized-text>
          </message-text>
          <message-parameters>
            <wdo-attribute
              wdo-name=
     "ChildDetailsListWDO[Context_Parallel.occurrenceCount]"
              name="fullName" />
            <wdo-attribute
              wdo-name=
              "Context_Parallel" name="occurrenceCount" />
          </message-parameters>
        </message>
        <decision-action>
          <message>
            <message-text>
              <localized-text>
                <locale language="fr">Validez les détails de l'enfant
                pour %1s associés à ce dossier %2n.</locale>
              </localized-text>
            </message-text>
            <message-parameters>
              <wdo-attribute
                wdo-name=
       "ChildDetailsListWDO[Context_Parallel.occurrenceCount]"
              name="fullName" />
              <wdo-attribute wdo-name="CaseDetails"
                             name="identifier" />
            </message-parameters>
          </message>
        </decision-action>
        ...
        <question>
          <message>
            <message-text>
              <localized-text>
                <locale language="fr">Les détails de cet enfant
                dont le prénom est %1s et le nom est
                %2s sont-ils corrects ?</locale>
              </localized-text>
            </message-text>
            <message-parameters>
              <wdo-attribute
                wdo-name=
       "ChildDetailsListWDO[Context_Parallel.occurrenceCount]"
              name="firstName" />
              <wdo-attribute
                wdo-name=
       "ChildDetailsListWDO[Context_Parallel.occurrenceCount]"
              name="surname" />
            </message-parameters>
          </message>
          <answers multiple-selection="false">
            <answer name="answerYes">
              <answer-text>
                <localized-text>
                  <locale language="fr">Oui</locale>
                </localized-text>
              </answer-text>
            </answer>
            <answer name="answerNo">
              <answer-text>
                <localized-text>
```

Validations

- Un objet de données de flux de travaux doit être indiqué pour une activité parallèle. Il doit s'agir d'un objet de données de flux de travaux de liste et il doit être valide dans le contexte de la définition de processus de flux de travaux qui le contient.
- Toutes les autres validations relatives aux activités parallèles sont décrites dans les chapitres qui décrivent les activités qu'une activité parallèle peuvent encapsuler (c'est-à-dire, «Manuel», à la page 61 et «Décision», à la page 80).

Informations d'exécution

Le moteur de flux de travaux charge les données d'instance pour l'objet de données de flux de travaux de liste associé à l'activité parallèle. Pour chaque élément dans l'objet de données de flux de travaux de liste, une nouvelle instance de l'activité encapsulée est créée et exécutée. Les détails de ce qui se produit lorsque ces instances de l'activité encapsulée sont exécutées se trouvent dans les chapitres correspondants décrivant les activités qu'une activité parallèle peut encapsuler («Manuel», à la page 61 et «Décision», à la page 80).

Lors de l'exécution, le moteur de flux de travaux traite une activité parallèle comme s'il s'agissait de plusieurs activités, contenues dans un bloc *Parallel (AND) Split/Join*. Une instance d'activité est créée pour chaque élément de la liste WDO d'activité parallèle (par exemple, si cette liste contient trois éléments, trois instances d'activité seront créées). Cela permet de s'assurer que toutes les instances d'activité associées à l'activité parallèle doivent être terminées avant que l'activité parallèle réelle ne soit considérée comme terminée et que le flux de travaux puisse continuer.

Afin de résoudre les mappages associés à une activité parallèle, chaque instance de l'activité encapsulée est associée à un élément de la liste WDO d'activité parallèle. L'élément est indexé à l'aide de l'objet de données de flux de travaux Context_Parallel (par exemple, ChildDetailsListWDO[Context_Parallel.occurrenceCount]).

Description des WDO (objet de données de flux de travaux) de contexte

Chaque instance d'activité parallèle est associée à un élément parmi les objets de données de flux de travaux de la liste d'activités parallèles. On accède à cet élément à l'aide de l'objet de données de flux de travaux Context_Parallel afin d'indexer l'objet de données de flux de travaux de la liste d'activités parallèles (par exemple, ChildDetailsListWDO[Context_Parallel.occurrenceCount]). Les éléments indexés peuvent ensuite être utilisés afin de mapper les données de la manière habituelle. Des exemples de mappage sont donnés dans les exemples de métadonnées indiqués ci-dessus (voir «Métadonnées pour une activité manuelle parallèle», à la page 98 et «Métadonnées pour une activité de décision parallèle», à la page 99. Le seul attribut disponible sur cet objet de données de flux de travaux est le suivant :

Context_Parallel.occurrenceCount

Chaque instance d'activité parallèle est associée à un élément parmi les objets de données de flux de travaux de la liste d'activités parallèles. L'attribut `occurrenceCount` est l'index de cet élément dans l'objet de données de flux de travaux de la liste d'activités parallèles. Il est de type INTEGER et il s'agit d'un index de base zéro.

Notifications d'activité

Présentation

Le moteur du flux de travaux est en mesure de notifier les utilisateurs intéressés au sujet de la progression d'une instance de processus de flux de travaux. Pour résumer, le moteur de flux de travaux peut générer une notification lorsqu'une activité s'exécute si la notification a été spécifiée dans les métadonnées de définition de processus associées. Une notification est spécifiée pour une activité en tant que métadonnées supplémentaires qui peuvent être associées à n'importe quel type d'activité.

Lorsque le moteur de flux de travaux exécute une activité, il vérifie si une notification a été configurée pour cette activité. S'il en existe une, une notification est créée par le moteur de flux de travaux indiquant qu'une étape donnée dans le processus de flux de travaux a été effectuée. La distribution de ces notifications à l'utilisateur est déterminée par le mécanisme de distribution de notification configuré dans l'application Cúram. Les notifications peuvent être délivrées via courrier électronique, sous la forme d'alertes envoyées dans la boîte de réception d'un utilisateur ou en utilisant les deux méthodes.

Détails de notification

Une notification correspond à des informations qui sont envoyées à un agent humain lorsqu'une étape du processus s'exécute. Les notifications se manifestent sous la forme d'alertes dans la boîte de réception d'un utilisateur ou sous la forme de courriers électroniques. Les agents auxquels la notification doit être envoyée sont déterminés par la stratégie d'allocation (voir «Stratégie d'allocation de notification», à la page 106) spécifiée pour la notification. Les informations présentées à l'utilisateur dans l'alerte ou le courrier électronique sont indiquées dans les métadonnées de notification.

Métadonnées

```
<manual-activity id="1" category="AC1">

  ...

    <notification delivery-mechanism="DM1">
      <subject>
       <message>
          <message-text>
           <localized-text>
              <locale language="en">
              Le numéro de dossier %1n pour le demandeur %2s a
              été clôturé.
              </locale>
            </localized-text>
          </message-text>
          <message-parameters>
            <wdo-attribute wdo-name=
              "CaseList[Context_Loop.loopCount]"
              name="identifier" />
            <wdo-attribute wdo-name="PersonDetails"
              name="userName" />
          </message-parameters>
        </message>
      </subject>
      <body>
        <message>
          <message-text>
            <localized-text>
              <locale language="en">
              Ce dossier concernait %1n et le demandeur %2s.
              </locale>
            </localized-text>
          </message-text>
          <message-parameters>
            <wdo-attribute wdo-name=
              "CaseList[Context_Loop.loopCount]"
              name="identifier" />
            <wdo-attribute wdo-name="PersonDetails"
              name="fullName" />
          </message-parameters>
        </message>
      </body>
      <allocation-strategy type="target" identifier="1" />
      <actions>
        <action page-id="viewTaskHome" principal-action="false">
          <message>
            <message-text>
              <localized-text>
                <locale language="en">
                Affichez la tâche associée au dossier %1n.
                </locale>
              </localized-text>
            </message-text>
            <message-parameters>
              <wdo-attribute wdo-name="TaskCreateDetails"
                name="caseID" />
            </message-parameters>
          </message>
          <link-parameter name="childID">
            <wdo-attribute wdo-name="ChildDependents"
              name="childID" />
          </link-parameter>
          <multiple-occurring-action>
      <list-wdo-name>ChildDependents</list-wdo-name>
          </multiple-occurring-action>
        </action>
        <action page-id="viewCaseHome" principal-action="false">
          <message>
            <message-text>
              <localized-text>
                <locale language="en">
                Affichez les détails du dossier pour %1n.
                </locale>
```

delivery-mechanism
Décrit le mécanisme utilisé pour distribuer la notification. Les mécanismes de distribution disponibles sont spécifiés dans la table de codes `DeliveryMechanism` de l'application. L'application et les clients Cúram peuvent développer cette table de codes et ajouter des mécanismes de distribution au besoin. Le mécanisme de distribution spécifié ne joue aucun rôle fonctionnel dans le moteur de flux de travaux étant donné qu'il se contente d'appeler le mécanisme de distribution configuré dans l'application pour distribuer la notification récemment créée.

subject
Représente un message texte paramétré qui détaille l'objet de la notification pour tous les environnements locaux. Cet objet s'affiche dans la boîte de réception de l'utilisateur pour l'alerte de notification. Pour plus de détails sur les messages paramétrés, voir «Manuel», à la page 61.

body Représente un message texte paramétré qui détaille le corps du texte associé à cette notification pour tous les environnements locaux. Lorsque l'utilisateur clique sur l'objet de la notification dans la boîte de réception, ce texte s'affiche en tant que texte intégral de la notification.

allocation-strategy
Représente la stratégie d'allocation utilisée pour déterminer les agents auxquels cette notification sera envoyée (voir «Stratégie d'allocation de notification», à la page 106).

actions
De la même manière qu'une activité manuelle («Manuel», à la page 61) peut disposer d'actions associées à sa tâche, une notification peut disposer d'actions associées que l'utilisateur averti peut effectuer. Cet élément de métadonnées représente les détails de ces actions de notification. Les détails des métadonnées pour les actions sont disponibles dans «Détails de la tâche», à la page 61.

multiple-occurring-action
Indique que cette action de notification va se produire plusieurs fois. Si ces métadonnées sont spécifiées pour une action de notification, le moteur de flux de travaux crée un enregistrement d'action pour chaque élément de l'objet de données de flux de travaux de type liste spécifié en tant qu'action à plusieurs occurrences lorsque cette activité est exécutée.

Remarquez que lorsque l'action répétitive est spécifiée pour une action de notification, alors un attribut de l'objet de données du flux de travaux de la liste associé doit être utilisé comme paramètre de lien pour l'action de notification.

list-wdo-name
Nom de l'objet de données de flux de travaux de type liste à utiliser avec l'action se produisant plusieurs fois.

Validations

- Un objet doit être défini pour la notification.
- Chaque attribut d'objet de données de flux de travaux mappé à un objet de notification doit exister dans la définition de processus qui le contient et doit être un objet de données de flux de travaux valide.
- Si un élément indexé d'un objet de données de flux de travaux de liste (par exemple, CaseList[Context_Loop.loopCount]) est utilisé en tant que paramètre de

texte d'objet de notification, l'objet de données de flux de travaux doit être un objet de données de flux de travaux de liste et l'activité contenant le mappage doit être contenue dans une boucle.

- Si l'objet de données de flux de travaux `Context_Parallel` est utilisé en tant que paramètre de texte d'objet de notification, l'activité contenant la notification doit donc être une activité `Parallel`.
- Si un élément indexé de l'objet de données de flux de travaux de liste parallèle est utilisé en tant que paramètre de texte d'objet de notification, l'activité contenant le mappage doit être une activité parallèle (par exemple, ParallelListWDO[Context_Parallel.occurrenceCount]). L'objet de données de flux de travaux étant indexé par l'objet de données de flux de travaux Context_Parallel doit correspondre à l'objet de données de flux de travaux de liste d'activité parallèle.
- Un corps de texte de notification doit être défini.
- Chaque attribut d'objet de données de flux de travaux mappé à un corps de texte de notification doit exister dans la définition de processus qui le contient et doit être un objet de données de flux de travaux valide.
- Si un élément indexé d'un objet de données de flux de travaux de liste (par exemple, CaseList[Context_Loop.loopCount) est utilisé en tant que paramètre de texte du corps de notification, l'objet de données de flux de travaux doit être un objet de données de flux de travaux de liste et l'activité contenant le mappage doit être contenue dans une boucle.
- Si l'objet de données de flux de travaux `Context_Parallel` est utilisé en tant que paramètre de texte de corps de notification, l'activité contenant la notification doit être une activité `Parallel`.
- Si un élément indexé de l'objet de données de flux de travaux de liste parallèle est utilisé en tant que paramètre de texte de corps de notification, l'activité contenant le mappage doit être une activité parallèle (par exemple, ParallelListWDO[Context_Parallel.occurrenceCount]). L'objet de données de flux de travaux étant indexé par l'objet de données de flux de travaux Context_Parallel doit correspondre à l'objet de données de flux de travaux de liste d'activité parallèle.
- Une stratégie d'allocation doit être définie pour une notification d'activité.
- Si une fonction est spécifiée en tant que stratégie d'allocation de notification, il doit s'agir d'une méthode métier Cúram valide qui renvoie un objet `AllocationTargetList`.
- Si le type d'allocation est une règle CER ou classique, le jeu de règles spécifié doit être valide.
- Un mécanisme de distribution doit être défini pour une notification d'activité.
- Les attributs d'objet de données de flux de travaux mappés aux paramètres de texte d'action de notification et de lien d'action de notification doivent exister dans la définition de processus qui le contient.
- Si un élément indexé d'un objet de données de flux de travaux de liste (par exemple, PersonDetailsList[Context_Loop.loopCount) est utilisé en tant que mappage de paramètre de texte d'action de notification ou de lien d'action de notification, l'objet de données de flux de travaux doit être un objet de données de flux de travaux de liste et l'activité contenant le mappage doit être contenue dans une boucle.
- Si l'objet de données de flux de travaux `Context_Parallel` est utilisé en tant que mappage de paramètre de texte d'action de notification ou de lien d'action de notification, l'activité contenant la notification doit donc être une activité `Parallel`.

- Si un élément indexé de l'objet de données de flux de travaux de liste parallèle est utilisé en tant que mappage de paramètre de texte d'action de notification ou de lien d'action de notification, l'activité contenant le mappage doit être une activité parallèle (par exemple, ParallelListWDO[Context_Parallel.occurrenceCount]). L'objet de données de flux de travaux étant indexé par l'objet de données de flux de travaux Context_Parallel doit correspondre à l'objet de données de flux de travaux de liste d'activité parallèle.
- Le nombre de paramètres substituables utilisés dans le texte d'objet de notification, le texte d'action de notification et le texte de corps de notification doit être égal au nombre d'attributs d'objet de données de flux de travaux mappés.
- L'objet de données de flux de travaux spécifié pour être utilisé dans l'action se produisant plusieurs fois doit être un objet de données de flux de travaux valide dans le contexte de la définition de processus de flux de travaux qui le contient. Il doit également être de type List.
- Au moins un attribut de l'objet de données de flux de travaux de liste d'action se produisant plusieurs fois doit être utilisé dans les paramètres de lien spécifiés pour une action se produisant plusieurs fois.

Code

Pour chaque action définie, la page d'action doit faire référence à une page Cúram valide dans l'application dont les paramètres de page sont entièrement remplis par les paramètres de lien d'action contenus dans les métadonnées de notification.

Une API `LocalizableStringResolver` est fournie à l'application afin de résoudre les chaînes de message paramétrées. Les méthodes de cette API vont résoudre et renvoyer le message spécifié pour l'environnement local requis. En parallèle, tout objet de données de flux de travaux à utiliser dans les paramètres substituables de message sera résolu et inclus dans la chaîne renvoyée.

Dans le cadre de l'API `LocalizableStringResolver`, une interface `NotificationStringResolver` est fournie afin de résoudre les messages paramétrés associés à des notifications. L'objet, le corps et le texte d'action d'une notification peuvent être résolus pour une utilisation dans l'application à l'aide des méthodes contenues dans cette API. L'application doit utiliser ces méthodes pour traiter la notification après que le moteur de flux de travaux a fait appel à la méthode de distribution de notification associée dans l'application.

Informations d'exécution

Une fois que le moteur de flux de travaux a terminé l'exécution d'une activité, il vérifie si une notification associée a été définie pour cette activité. Si une notification a été définie, le moteur détermine les utilisateurs à avertir à partir de la stratégie d'allocation employée et appelle la méthode de distribution des notifications dans l'application avec les détails de notification.

Stratégie d'allocation de notification

Conditions préalables

La stratégie d'allocation de notification détermine l'utilisateur ou les utilisateurs à avertir une fois que l'activité associée s'est produite. La définition de la stratégie d'allocation de notification à utiliser est exactement la même que celle utilisée pour les tâches de l'activité manuelle (voir «Stratégie d'allocation», à la page 68).

Code

L'application doit implémenter l'interface de rappel `NotificationDelivery` afin de déterminer le traitement des notifications dans l'application.

Le moteur de flux de travaux appellera la méthode `deliverNotification` dans la classe d'implémentation `curam.util.workflow.impl.NotificationDelivery` afin de traiter la notification. Le moteur transmettra à la fois la liste des cibles d'allocation déterminées par la stratégie d'allocation et les détails de la notification requis pour cette méthode d'application.

La propriété d'application curam.custom.notifications.notificationdelivery définit l'interface d'implémentation `NotificationDelivery` qui sera utilisée par le moteur de flux de travaux afin de traiter la notification.

La méthode `deliverNotification` dans cette classe d'implémentation par défaut est surchargée. Ceci est dû au fait que les différents types de stratégie d'allocation renvoient les cibles d'allocation dans différents formats. Toutefois, il s'agit d'un détail d'implémentation que les développeurs de classes de distribution de notification personnalisées ne doivent pas avoir à traiter, notamment parce que le processus métier pour toutes les versions de la méthode doit être le même.

```
package curam.util.workflow.impl;

      ...

      public interface NotificationDelivery {

        boolean deliverNotification(
          final NotificationDetails notificationDetails,
          final Object allocationTargets);

        boolean deliverNotification(
            final NotificationDetails notificationDetails,
            final Map allocationTargets);


        boolean deliverNotification(
            final NotificationDetails notificationDetails,
            final String allocationTargetID);

            ...
      }
```

Afin d'atténuer ce problème, `curam.core.sl.impl.DefaultNotificationDeliveryAdapter` fournit un mécanisme plus pratique pour l'implémentation d'un programme de résolution de travail. Cette classe implémente les différentes méthodes et convertit leurs paramètres d'entrée en listes de cible d'allocation permettant aux développeurs de logique de distribution des notifications personnalisées d'étendre cette classe et d'implémenter une méthode qui est appelée quelle que soit la source des cibles d'allocation.

```
package curam.core.sl.impl;

      ...

      public abstract class DefaultNotificationDeliveryAdapter
        implements curam.util.workflow.impl.NotificationDelivery {

        public abstract boolean deliverNotification(
          final NotificationDetails notificationDetails,
          final AllocationTargetList allocationTargets);

            ...
      }
```

Outre cette classe d'adaptateur, l'application est livrée avec une implémentation de distribution de notification prête à l'emploi. Cette classe est appelée `curam.core.sl.impl.DefaultNotificationDelivery` et elle sert d'exemple sur la manière d'étendre l'adaptateur.

Les stratégies de distribution de notification sont répertoriées dans la table de codes DELIVERYMECHANISM. L'ajout d'une nouvelle stratégie correspond simplement à l'extension de cette table de codes avec une nouvelle stratégie (SMS, par exemple) et à l'implémentation d'une stratégie de distribution qui reconnaît ce code et exécute la logique appropriée. Cependant, puisque la classe de distribution de notification est définie à l'aide d'une propriété d'application unique, le remplacement de la classe `curam.core.sl.impl.DefaultNotificationDelivery` désactive les mécanismes de distribution prête à l'emploi. Si le but est d'étendre le mécanisme de distribution prête à l'emploi au lieu de le remplacer, des classes personnalisées doivent étendre la classe `curam.core.sl.impl.DefaultNotificationDelivery` dans un format qui préserve la fonctionnalité d'origine. La classe `curam.core.sl.impl.DefaultNotificationDelivery` a été implémentée dans ce but.

```
package curam.core.sl.impl;

 public class DefaultNotificationDelivery
     extends DefaultNotificationDeliveryAdapter {

   public boolean deliverNotification(
       NotificationDetails notificationDetails,
       AllocationTargetList allocationTargetList) {
     return selectDeliveryMechanism(
       notificationDetails, allocationTargetList);
   }

   protected boolean selectDeliveryMechanism(
     NotificationDetails notificationDetails,
      AllocationTargetList allocationTargetList) {

     boolean notificationDelivered = false;
     if (notificationDetails.deliveryMechanism.equals(
         curam.codetable.DELIVERYMECHANISM.STANDARD)) {
       notificationDelivered = standardDeliverNotification(
         notificationDetails, allocationTargetList);
     } else if (
     ...
     return notificationDelivered;
   }

   ...

 }
```

La classe curam.core.sl.impl.DefaultNotificationDelivery implémente la méthode deliverNotification à partir de l'adaptateur abstrait, mais délègue immédiatement l'identification du mécanisme afin d'utiliser une méthode protégée. La méthode protégée selectDeliveryMechanism peut être remplacée par des sous-classes afin d'identifier tout mécanisme de distribution personnalisée et d'effectuer les opérations appropriées telles qu'illustrées dans l'exemple ci-après.

```
public class CustomNotificationDeliveryStrategy
     extends DefaultNotificationDelivery {

   protected boolean selectDeliveryMechanism(
     NotificationDetails notificationDetails,
      AllocationTargetList allocationTargetList) {

     boolean notificationDelivered = false;
     boolean superNotificationDelivered = false;
     superNotificationDelivered = super.selectDeliveryMechanism(
       notificationDetails, allocationTargetList);
     if (notificationDetails.deliveryMechanism.equals(
       curam.codetable.DELIVERYMECHANISM.CUSTOM)) {
       notificationDelivered = customDeliverNotification(
         notificationDetails, allocationTargetList);
     }
     return (superNotificationDelivered || notificationDelivered);
   }
 }
```

Notez que la méthode selectDeliveryMechanism dans la classe personnalisée délègue tout d'abord sa super-classe avant d'exécuter l'une des ses propres logiques. L'extension de la fonctionnalité permet aux classes personnalisées d'appeler le mécanisme de distribution prête à l'emploi sans avoir à connaître les codes spécifiques reconnus par la classe parent. Cette approche permet également des mises à niveau aisées, étant donné que, si une version future de Cúram prend en charge plus de mécanismes de distribution prête à l'emploi, une classe personnalisée implémentée comme indiqué ici n'aura pas besoin d'être modifiée pour bénéficier des nouvelles fonctionnalités.

L'indicateur booléen renvoyé par la fonction de distribution de notification ci-dessus est utilisé pour indiquer au moteur de flux de travaux que la notification a été distribuée à au moins un utilisateur sur le système. Si ce n'est pas le cas, le moteur écrit un enregistrement d'audit de flux de travaux détaillant ce fait.

Transitions

Présentation

Les transitions fournissent des liens entre les activités. Elles représentent la principale structure de contrôle des flux, et dictent l'ordre d'exécution des activités. Des transitions sont unidirectionnelles et une activité peut avoir multiples transitions entrantes et sortantes formant des points de branchement et de synchronisation. Etant donné que chaque définition de processus doit posséder une activité de début et une activité de fin (voir «Activité de base», à la page 32), une définition de processus peut être considérée de manière officieuse comme un graphe orienté dans lequel les activités sont les vertex, les transitions sont les arcs et chaque chemin partant de l'activité de début mène au final à l'activité de fin.

Métadonnées

```
<workflow-process id="32456" ..... >
    <name>WorkflowTestProcess</name>
    ...
    <wdos>
    ...
    </wdos>
    <activities>
      <start-process-activity id="512">
        ...
      </start-process-activity>
      <route-activity id="513" category="AC1">
        ...
      </route-activity>
      <route-activity id="514" category="AC1">
        ...
      </route-activity>
      <end-process-activity id="515">
        ...
      </end-process-activity>
    </activities>
    <transitions>
      <transition id="1" from-activity-idref="512"
                         to-activity-idref="513" />
      <transition id="2" from-activity-idref="513"
                         to-activity-idref="514">
        <condition>
          <expression id="5"
                 data-item-lhs="TaskCreateDetails.reservedByInd"
                 operation="==" data-item-rhs="true"
                 opening-brackets="2"/>
          <expression id="6"
                 data-item-lhs="TaskCreateDetails.subject"
                 operation="&amp;gt;"
                 data-item-rhs="&amp;quot;MANUAL&amp;quot;"
                 conjunction="and" closing-brackets="1"/>
          <expression id="7"
                 data-item-lhs="TaskCreateDetails.status"
                 operation="!="
                 data-item-rhs="&amp;quot;OPEN&amp;quot;"
                 conjunction="or"/>
          <expression id="8"
                 data-item-lhs="TaskCreateDetails.status"
                 operation="&amp;lt;="
                 data-item-rhs="&amp;quot;INPROGRESS&amp;quot;"
                 conjunction="or" closing-brackets="1"/>
        </condition>
      </transition>
      <transition id="3" from-activity-idref="514"
                         to-activity-idref="515">
    </transitions>
  </workflow-process>
```

transitions

Une définition de processus de flux de travaux doit contenir au moins une transition. Cet élément contient les détails de toutes les transitions entre les activités de la définition de processus de flux de travaux spécifiée.

transition

Cet élément contient les détails d'une transition entre deux activités de la définition de processus de flux de travaux spécifiée. Les zones obligatoires qui composent une transition sont décrites ci-après.

id Il s'agit d'un identificateur `64 bits` fourni par le serveur de clés Cúram lorsque les transitions sont créées dans l'outil de définition de processus (PDT). L'identificateur de transition doit

obligatoirement être unique au sein d'une définition de processus. Toutefois, l'unicité globale au sein de toutes les définitions de processus sur le système n'est pas obligatoire.

from-activity-idref
Il s'agit de l'identificateur `64 bits` de l'activité source de la transition.

to-activity-idref
Il s'agit de l'identificateur `64 bits` de l'activité cible de la transition.

condition
Les transitions peuvent éventuellement posséder une condition qui détermine si la transition donnée sera suivie ou non. Une condition est une liste d'expressions qui effectuent des opérations logiques sur les attributs des objets de données de flux de travaux. Les conditions sont décrites en détails dans le «Conditions», à la page 112.

Validations

- L'activité source définie pour la transition doit être une activité valide au sein de la définition de processus de flux de travaux qui la contient.
- L'activité cible définie pour la transition doit être une activité valide au sein de la définition de processus de flux de travaux qui la contient.
- Les activités source et cible définies pour une transition ne peuvent pas être identiques.
- L'activité de processus de début d'une définition de processus de flux de travaux ne doit pas contenir de transition entrante.
- L'activité de processus de fin d'une définition de processus de flux de travaux ne doit pas contenir de transition sortante.
- Toutes les activités définies dans la définition de processus de flux de travaux, à l'exception de l'activité de processus de fin, doivent contenir au moins une transition entrante.
- Toutes les activités définies dans la définition de processus de flux de travaux, à l'exception de l'activité de processus de début, doivent contenir au moins une transition sortante.

Informations d'exécution

Les activités qui effectuent un travail lié à l'application (par opposition au travail avec le moteur de flux de travaux uniquement, tel que les activités d'acheminement et de processus de fin) nécessitent une limite transactionnelle transparente entre le moteur et le code d'application. Il s'avère également utile de posséder des appels asynchrones entre le moteur de flux de travaux et l'application (c'est-à-dire qu'un utilisateur ne devrait pas avoir à attendre que le flux de travaux passe à l'activité suivante avant que le contrôle ne lui soit renvoyé dans l'interface utilisateur).

Il existe trois fonctions différentes dans une activité de flux de travaux : `start()`, `execute()` et `complete()`. Après qu'une activité soit terminée dans l'instance du processus de flux de travaux, le moteur de flux de travaux appelle la fonction pour continuer le processus. Cette fonction évalue les transitions sortantes provenant de cette activité afin de déterminer laquelle ou lesquelles seront suivies.

Pour chaque activité à suivre, la fonction `start()` correspondante est appelée. Les données de l'instance d'activité appropriée sont ensuite configurées pour cette activité. Si l'activité doit être exécutée directement, sans messagerie JMS (l'API Java

Message Service fait partie de Java EE), (à savoir qu'une activité d'acheminement est toujours exécutée directement car aucun travail lié à l'application n'est impliqué), la méthode `execute()` est appelée dans ce cas. Sinon, un message JMS est envoyé pour exécuter l'activité spécifiée (à savoir une activité automatique). Le gestionnaire de message de flux de travaux résout le processus et l'activité spécifiés dans le message et appelle la fonction `execute()` sur l'activité.

Après avoir appelé le code d'application pour réaliser le travail spécifié par l'activité, un autre message est envoyé pour terminer l'activité. Le gestionnaire de message de flux de travaux résout à nouveau le processus et l'activité spécifiés dans le message et appelle la fonction `complete` pour l'activité. Après avoir marqué l'activité comme terminée, la fonction permettant de continuer le processus est de nouveau appelée pour résoudre l'ensemble de transitions à suivre à partir de l'activité terminée et le processus redémarre.

Conditions

Présentation

Les constructions de contrôle de flux décrites dans «Transitions», à la page 109 et «Début et fin de boucle», à la page 94 requièrent ou prennent en charge l'évaluation des conditions afin de déterminer la manière dont le flux de travaux doit se poursuivre. Les activités de début de boucle doivent posséder des métadonnées qui spécifient les conditions de sortie de boucle, alors que les transitions peuvent éventuellement avoir une condition pour déterminer si la transition donnée sera suivie ou non.

décrit la construction de métadonnées de définition de processus qui représente une condition. Une condition est une liste d'expressions qui effectuent des opérations logiques sur les attributs des objets de données de flux de travaux. La condition elle-même est un composant dont la valeur est une conjonction ou une disjonction d'expressions de ses constituants. Les constructions parent (boucles et transitions) sont chargées de prendre des mesures appropriées à la suite de l'évaluation des conditions.

Métadonnées

```
<workflow-process id="32456" ..... >
    ...
    <activities>
    ...
    </activities>
    <transitions>
      <transition id="1" from-activity-idref="512"
                         to-activity-idref="513">
        <condition>
          <expression id="5"
               data-item-lhs="TaskCreateDetails.reserveToMeInd"
               operation="==" data-item-rhs="true"
               opening-brackets="2"/>
          <expression id="6"
               data-item-lhs="TaskCreateDetails.caseID"
               operation="&amp;amp;gt;"
               data-item-rhs="2" conjunction="and"
               closing-brackets="1"/>
          <expression id="7"
               data-item-lhs="TaskCreateDetails.status"
               operation="!="
               data-item-rhs="&quot;Completed&quot;"
               conjunction="or"/>
          <expression id="8"
               data-item-lhs="TaskCreateDetails.status"
               operation="&amp;amp;lt;="
               data-item-rhs="&quot;Closed&quot;"
               conjunction="or" closing-brackets="1"/>
        </condition>
      </transition>
      <transition id="2" from-activity-idref="512"
                         to-activity-idref="513">
        <condition>
          <expression id="9" function="isNothing"
               data-item-rhs="TaskCreateDetails.subject"/>
        </condition>
      </transition>
      <transition id="3" from-activity-idref="513"
                         to-activity-idref="514">
        <condition>
          <expression id="10"
                data-item-rhs="TaskCreateDetails.reserveToMeInd"
                conjunction="and" function="not" />
        </condition>
      </transition>
      <transition id="4" from-activity-idref="514"
                         to-activity-idref="515">
        <condition>
          <expression id="6"
               data-item-lhs
               ="ClaimantDependents[Context_Loop.loopCount]"
               operation="&amp;amp;gt;"
               data-item-rhs="20"
               conjunction="and"
               closing-brackets="1"/>
        </condition>
      </transition>
    </transitions>
  </workflow-process>
```

condition

Ces métadonnées sont obligatoires pour une activité de début de boucle (car une condition de sortie doit être spécifiée pour une boucle) mais sont facultatives pour une transition (la spécification d'une condition n'est pas obligatoire pour une transition). Cet élément contient les détails de toutes les expressions définies pour la condition.

expression

Contient les détails d'une expression incluse dans une condition. Une ou plusieurs expressions peuvent être spécifiées pour une condition associée. Deux types d'expression peuvent être définis dans une condition. Il s'agit d'expressions de fonction (qui utilisent une ou deux fonctions prédéfinies : `not()` et `isNothing()`) et d'expressions de données élémentaires (où l'expression de condition créée applique l'opérateur choisi à deux attributs d'objet de données de flux de travaux ou à un attribut d'objet de données de flux de travaux et une constante). Une expression de transition se compose des attributs suivants :

id
Représente un identificateur `64 bits` fourni par le serveur de clés Cúram lors de la création des expressions de transition dans l'outil de définition de processus. L'identificateur d'expression doit obligatoirement être unique au sein d'une définition de processus. Toutefois, l'unicité globale au sein de toutes les définitions de processus sur le système n'est pas obligatoire.

data-item-rhs
Représente le nom de l'élément de données à utiliser dans la partie droite de l'expression de condition. Dans le cas d'une expression de condition pour un élément de données, il peut représenter un attribut d'objet de données de flux de travaux (voir «Objets de données de flux de travaux», à la page 17) ou une valeur constante à laquelle l'opérateur choisi sera appliqué. Pour les expressions de condition de fonction, cet élément représente un attribut d'objet de données de flux de travaux que l'une des deux fonctions prédéfinies utilisera pour évaluer la condition.

data-item-lhs
Cette balise de métadonnées est facultative car elle n'est pas obligatoire pour une expression de condition de fonction. Dans le cas d'une expression de condition pour un élément de données, elle représente le nom de l'élément de données à utiliser dans la partie gauche de la condition (c'est-à-dire un attribut d'objet de données de flux de travaux).

operation
Cette balise de métadonnées est facultative car elle n'est pas obligatoire pour une expression de condition de fonction. Dans le cas d'une expression de condition pour un élément de données, elle représente un identificateur pour l'opération logique qui sera appliqué à deux attributs d'objet de données de flux de travaux ou à un attribut d'objet de données de flux de travaux et une valeur constante. La liste ci-après répertorie les opérateurs valides qui peuvent être utilisés dans une expression de condition pour un élément de données :

Tableau 4. Opérateurs d'expression de condition

Opérateur	Explication
==	égal à
!=	différent de
<=	inférieur ou égal à
>=	supérieur ou égal à
<	inférieur à
>	supérieur à

conjunction

Représente un identificateur pour une conjonction logique qui peut être utilisée dans une expression de fonction ou de condition d'élément de données. Deux valeurs peuvent être définies pour cet attribut : `and` (valeur par défaut) et `or`. Lorsqu'une condition contient plusieurs expressions, la conjonction logique est utilisée dans l'évaluation de la condition complète.

function

Il s'agit d'un élément facultatif étant donné qu'il est uniquement utilisé lors de la spécification d'une expression de condition de fonction. Comme mentionné précédemment, il existe deux fonctions prédéfinies : `Not()` et `isNothing()`. La fonction `Not()` joue le rôle d'opérateur d'inversion logique. En règle générale, elle est appliquée à une valeur booléenne. La fonction `isNothing()` peut être appliquée à n'importe quel type d'attribut d'objet de données de flux de travaux autre qu'une valeur booléenne. Elle peut être utilisée pour tester les scénarios dans lesquels les données obligatoires n'existent pas ou n'ont pas été fournies. La fonction renvoie la valeur booléenne `True` si l'attribut d'objet de données de flux de travaux examiné ne contient aucune donnée.

opening-brackets

Elément facultatif (valeur par défaut : 0) car il peut ne pas être spécifié pour un type d'expression de condition. Il représente le nombre de crochets ouvrants à insérer au début de l'expression.

closing-brackets

Elément facultatif (valeur par défaut : 0) car il peut ne pas être spécifié pour un type d'expression de condition. Il représente le nombre de crochets fermants à insérer à la fin de l'expression.

Une expression individuelle n'est pas contrainte à contenir autant de crochets ouvrants que de crochets fermants (à moins qu'il n'existe une seule expression dans la condition). La condition doit contenir autant de crochets ouvrants et fermants (avec toutes les expressions incluses). Par conséquent, soyez attentif lorsque vous spécifiez le nombre et la position des crochets ouvrants et fermants au sein d'une expression individuelle et de la condition dans son intégralité, car ces crochets aident à déterminer la méthode d'évaluation de la condition et des expressions individuelles au sein de cette condition. Soyez également prudent lorsque vous spécifiez la conjonction d'une expression, car un échec peut provoquer des résultats inattendus.

Validations

- L'attribut d'objet de données de flux de travaux indiqué comme l'élément de données situé à droite de l'expression de la condition doit être un attribut d'objet de données de flux de travaux dans le contexte de la définition du processus de flux de travaux.
- L'attribut d'objet de données de flux de travaux indiqué comme l'élément de données situé à droite de l'expression de la condition doit être un attribut d'objet de données de flux de travaux dans le contexte de la définition du processus de flux de travaux.

- L'opérateur indiqué dans une expression de condition d'élément de données doit être un opérateur valide et pris en charge.
- La fonction indiquée dans une expression de condition de fonction doit être une fonction valide et prise en charge.
- La conjonction indiquée dans une expression de condition de fonction doit être une conjonction valide et prise en charge.
- Le nombre de parenthèses ouvrantes et le nombre de parenthèses fermantes doivent être égaux dans le contexte de la condition générale.
- Si la fonction `Non()` est indiquée pour une expression de condition de fonction, le type d'attribut d'objet de données de flux de travaux spécifié à l'élément de données à droite de l'expression doit être de type BOOLEAN.
- Si la fonction `isNothing()` est indiquée pour une expression de condition de fonction, le type d'attribut d'objet de données de flux de travaux spécifié comme l'élément de données à droite de l'expression ne doit pas être de type BOOLEAN.
- Si l'élément de données à droite d'une expression de condition d'élément de données est un attribut d'objet de données de flux de travaux, le type de cet attribut doit être compatible avec l'attribut d'objet de données de flux de travaux d'élément de données à droite. De même, si l'élément de données à droite a été indiqué en tant que valeur constante, il doit être compatible avec le type d'attribut d'objet de données de flux de travaux d'élément de données à gauche.
- Si une expression de condition de transition à gauche ou à droite contient un élément indexé à partir d'un objet de données de flux de travaux de liste (c'est-à-dire ChildDependents[Context_Loop.loopCount].age), alors l'objet de données de flux de travaux associé doit être un objet de données de flux de travaux de liste et les activités incluses dans la transition doivent être contenues dans une boucle.
- Pour une expression de condition, soit le côté droit ou le côté gauche de l'expression indique l'attribut *taille()* pour un objet de données de flux de travaux, ensuite cet objet de données de flux de travaux doit être un objet de données de flux de travaux.
- Pour une expression de condition, soit le côté droit ou gauche de l'expression indique l'attribut *taille()* d'un objet de données de flux de travaux, ensuite l'élément sur l'autre côté de l'expression doit être affectable au type INTEGER.
- Pour une expression de condition de boucle, si le côté droit ou gauche de l'expression indique l'attribut *isEmpty()* d'un objet de données de flux de travaux, alors l'objet de données de flux de travaux doit être une liste d'objet de données de flux de travaux.
- Pour une expression de condition de boucle, si le côté droit ou le côté gauche de l'expression indique l'attribut *isEmpty()* pour un objet de données de flux de travaux, alors l'autre côté de l'expression doit être affectable au type BOOLEAN.

Fractionnement/Jointure

Introduction

Les transitions permettent de lier des activités dans une définition de processus. Dans la configuration des activités et des transitions la plus basique, chaque activité ne comporte qu'une seule transition entrante et une seule transition sortante. Toutefois, il est souvent utile de suivre plusieurs chemins pour une activité qui entraîne un fractionnement (par exemple, plusieurs transitions provenant d'une activité). Pour prendre en charge une structure de bloc valide dans une définition de processus (voir «Structure d'un flux de travaux», à la page 118

118), chaque fractionnement doit correspondre à une jointure (par exemple, plusieurs transitions se rencontrant dans une activité). En général, un fractionnement permet d'effectuer plusieurs unités d'exécution de travail en même temps tandis qu'une jointure représente le point de synchronisation réciproque pour ces unités d'exécution.

Il existe deux raisons pour qu'une activité dispose d'un fractionnement (et par extension, que d'autres activités disposent d'une jointure). La première consiste à permettre d'effectuer un travail qui n'a pas de dépendance en parallèle, tandis que la seconde vise à permettre de faire un choix entre un certain nombre de différents chemins dans le flux de travaux.

Au niveau des métadonnées, chaque activité possède un fractionnement et un type de jointure. Lorsque l'activité ne comporte qu'une seule transition entrante ou sortante, un type `none` est attribué au fractionnement ou à la jointure, respectivement. Les deux autres types de fractionnement et de jointure, `choice` (également appelé XOR) et `parallel` (également appelé AND) sont suffisamment explicites et représentent l'objet principal de ce chapitre.

Division XOR de choix

Métadonnées

```
<manual-activity id="1" category="AC1">
    ...
    <join type="and"/>
    <split type="xor">
        <transition-id idref="1"/>
        <transition-id idref="2"/>
        <transition-id idref="3"/>
        <transition-id idref="4"/>
    </split>
    <task>
      ...
    </task>
    <allocation-strategy type="target"
                         identifier="HEARINGSCHEDULE"/>
    <event-wait>
      ...
    </event-wait>
  </manual-activity>
```

split Cet élément est présent pour chaque activité et contient les détails de la division à partir de l'activité. Il inclut une liste des transitions provenant de l'activité spécifiée qui seront résolues par le moteur de flux de travaux à la fin de l'activité associée afin d'examiner si elles peuvent être suivies ou non.

L'ordre des transitions dans cette liste est important pour un type de division `XOR`, car c'est la première transition, admissible dans la liste des transitions classées, qui sera suivie par le moteur de flux de travaux. Dans l'exemple de métadonnées ci-dessus, si les conditions de transition pour les identificateurs de transition 2, 3 et 4 sont remplies, la transition portant l'identificateur 2 sera suivie car il s'agit de la première transition admissible dans la liste des transitions classées.

type Représente le type de division. Comme décrit ci-dessus, il existe trois types de division possibles. Le type de division `none` indique qu'il existe une seule transition sortante provenant de l'activité spécifiée. Le type de division `xor` indique un choix. La première

transition admissible dans la liste des transitions classées sera suivie. Le type de division `and` indique un chemin d'exécution parallèle qui garantit que toutes les transitions admissibles répertoriées dans la liste des transitions classées seront suivies en parallèle.

transition-id

Contient une référence à la transition spécifiée. Cette balise de métadonnées contient plusieurs entrées lorsque le type de division est `xor` ou `and`.

idref Contient une référence à une transition dans la définition de processus de flux de travaux.

Division AND parallèle

Métadonnées

```
<manual-activity id="1" category="AC1">
      ...
    <join type="none"/>
    <split type="and">
      <transition-id idref="1"/>
      <transition-id idref="2"/>
      <transition-id idref="3"/>
      <transition-id idref="4"/>
    </split>
    <task>
      ...
    </task>
    <allocation-strategy type="target"
                         identifier="HEARINGSCHEDULE"/>
    <event-wait>
      ...
    </event-wait>
  </manual-activity>
```

Les métadonnées du type de division `and` sont identiques à celles du type de division `xor` (voir «Division XOR de choix», à la page 117), à la différence près que le type de division est spécifié sous la forme `and`. Lorsque le moteur de flux de travaux détermine la liste de transitions à suivre à partir d'une activité spécifiée, l'ordre des transitions de cette liste n'est pas important car toutes les transitions admissibles dans une division `and` seront suivies. La liste des transitions classées est conservée dans cette instance pour ce type de division afin de faciliter le changement de type de division de `and` à `xor`, auquel cas l'ordre des transitions redevient important.

Structure d'un flux de travaux

Présentation

La structure d'un processus de flux de travaux est déterminée par les activités du processus et par les transitions entre elles. Ainsi, un flux de travaux représente un graphe dans lequel les activités sont des vertex et les transitions sont des arcs (le graphe qui représente un flux de travaux peut être visualisé à l'aide de la fonction *Visualize Workflow Process* (Visualiser les processus de flux de travaux) de l'outil de définition de processus).

Pour que le moteur de flux de travaux puisse interpréter et exécuter un processus avec succès, le graphe représenté par ce processus doit répondre à certains critères. Ce chapitre présente ces critères sous deux rubriques : Structure d'un graphe et Structure d'un bloc.

Structure de graphe

Etant donné qu'un ensemble d'activités et de transitions dans un processus forme un graphe, une théorie de graphe peut être appliquée pour intercepter plusieurs problèmes structurels bien connus avant qu'un processus ne soit exécuté.

Théorie de graphe : La théorie de graphe est une branche des mathématiques. Heureusement, les composants de la théorie de graphe qui s'appliquent au flux de travaux sont très simples. C'est pourquoi, le présent chapitre ne nécessite aucune connaissance préalable de la théorie de graphe (aucun diplôme de mathématiques n'est nécessaire !). De nombreuses informations sur la théorie de graphe sont disponibles sur Internet, où il est également possible de trouver facilement des discussions concernant les nombreux sujets abordés dans ce chapitre.

Prenons par exemple un processus dans lequel une activité comporte une transition vers une autre activité, qui à son tour comporte une transition vers la première activité. Ceci forme un cycle dans le graphe du processus.

Si aucune condition ne se trouve dans les transitions, il est sûr que le processus se termine par une boucle sans fin. Ces boucles sont appelées boucles informelles (ou boucles 'ad hoc') et leur présence rend impossibles de nombreuses validations structurelles utiles. Pour cette raison (entre autres), le flux de travaux de Cúram fournit des constructions formelles pour délimiter les sections itératives d'un processus (activités de début et de fin de boucle). Ceci lui permet de détecter la présence de boucles ad-hoc dans les processus et empêche leur publication.

Analogies de code : De nombreux développeurs seront familiers avec l'instruction GOTO du langage de programmation et les accolades couramment utilisées pour délimiter le début ({) et la fin (}) d'une boucle formelle.

GOTO est semblable à des boucles ad hoc dans un flux de travaux. Les accolades sont analogues aux activités de début et de fin de boucle formelle dans un flux de travaux.

Structure de bloc

Il existe plusieurs éléments de flux de travaux qui peuvent affecter le choix du ou des chemins de flux dans un flux de travaux au moment de l'exécution. Celles-ci comprennent :

- «Division XOR de choix», à la page 117
- «Division AND parallèle», à la page 118
- «Début et fin de boucle», à la page 94

Ces éléments vont toujours par paires. Cela vient du fait qu'ils délimitent des zones où le processus doit faire preuve d'un comportement spécifique (lié au flux de contrôle). Ces zones sont normalement appelées 'blocs', car elles disposent d'un point de départ qui doit correspondre à un point de fin.

Prenons par exemple un processus disposant d'une structure dans laquelle tous les chemins qui sortent d'un fractionnement de choix (un seul chemin sortant suivi garanti) convergent tous vers une jointure parallèle (qui attendra que tous les

chemins entrants se terminent avant d'exécuter l'activité suivante). Dans ce cas, il est garanti que le processus bloque à la jointure parallèle. Voici un exemple de problème avec la structure de bloc qui peut être interceptée par des validations avant même qu'un processus ne soit exécuté.

Analogie de blocs

Il est possible de faire une analogie pour savoir comment les "blocs" fonctionnent dans un flux de travaux avec le rôle que jouent les parenthèses (comme celles-ci !) dans une phrase. Les parenthèses disposent d'un point de début explicite '(', qui correspond toujours à un point de fin spécifique ')'. Elles délimitent une zone de la phrase qui a une signification particulière.

La manière dont les parenthèses fonctionnent dans une expression mathématique est une analogie plus étroite. Outre les parenthèses d'ouverture et de fermeture, une expression mathématique peut utiliser plusieurs types de parenthèses. Les expressions entre parenthèses peuvent être imbriquées dans une autre, mais elles ne peuvent pas être entrelacées. Ceci ressemble au fonctionnement des blocs dans un flux de travaux.

Types de bloc pris en charge par les flux de travaux

Les sections suivantes décrivent les différents types de blocs dans le flux de travaux Cúram, leur début et leur fin, ainsi que leurs objectifs.

Bloc de 'choix' (XOR) : Un *bloc de choix* commence par un fractionnement de choix (XOR) et se termine par une jointure de choix (XOR) (les 'parenthèses'). Il indique que, parmi les chemins possibles dans le bloc, *un* seul peut être suivi.

Le fractionnement comporte plusieurs transitions qui sortent de celui-ci, indiquant les chemins qu'une instance de processus peut suivre. Etant donné qu'il s'agit d'un bloc de choix, les chemins sont mutuellement exclusifs et un seul chemin sera suivi par une instance de processus donnée.

Le fractionnement de choix doit être accompagné d'une jointure de choix correspondante. Il indique le point au niveau duquel le processus cesse d'être distinct pour chaque chemin, de sorte que les chemins sont à nouveau fusionnés (en d'autres termes, le reste du processus est commun).

Bloc 'parallèle' (AND) : Un *bloc parallèle* démarre au niveau d'une division (AND) parallèle et se termine au niveau d'une liaison (AND) parallèle (les 'crochets'). Il indique que *plusieurs ou tous* les chemins possibles au sein du bloc peuvent être suivis.

Le fractionnement comporte plusieurs transitions qui sortent de celui-ci, indiquant les chemins qu'une instance de processus peut suivre. Etant donné qu'il s'agit d'un bloc parallèle, n'importe quel nombre de chemins peuvent être suivis en parallèle (en supposant que leurs conditions de transition sont remplies).

La division parallèle doit correspondre à une liaison parallèle. Celle-ci indique le point auquel tous les chemins parallèles doivent être synchronisés pour que le flux de travaux puisse continuer.

Bloc de type 'boucle' : Un *bloc de boucle* est démarré au cours d'une activité de début de boucle et prend fin au cours d'une activité de fin de boucle (les 'crochets'). Il indique que la section du flux de travaux délimitée par les activités de début et de fin de boucle doit être répétée tant que la condition de fin de boucle n'est pas remplie.

L'activité de début de boucle marque le point auquel l'exécution doit revenir si la condition de fin de boucle est remplie (c'est-à-dire, l'endroit où revenir si le moteur détermine que la boucle doit itérer). L'activité de fin de boucle marque le point auquel l'exécution doit directement passer si la condition de fin de boucle n'est *pas* remplie.

Règles structurelles

Les concepteurs de flux de travaux devraient connaître certaines règles structurelles lors de la construction de définitions de processus. Lorsqu'un processus de flux de travaux Cúram est validé, les validations estiment si la structure du processus est conforme à ces règles. Comme pour toutes les validations, l'objectif est de vérifier que le processus peut être exécuté par le moteur de flux de travaux.

Règles de la structure d'un graphe

Un processus Cúram doit représenter un graphe qui possède les propriétés suivantes : orienté, connecté et acyclique. Cela peut sembler complexe, mais il ne s'agit que des termes techniques décrivant certaines propriétés de graphe très simples.

- Un graphe "orienté" est un graphe dans lequel chaque arête prend une seule direction. En termes de flux de travaux, cela signifie qu'une transition de l'activité A vers l'activité B ne peut pas être utilisée pour revenir de B à A. Il s'agit d'une évidence dans les flux de travaux Cúram. Cette propriété est mentionnée ici car la propriété 'acyclique' (voir ci-après) est définie de manière différente pour les graphes et les graphes orientés.
- Un graphe "connecté" est un graphe dans lequel chaque vertex peut être atteint. En termes de flux de travaux, cela signifie que chaque activité du processus doit pouvoir être atteinte en prenant au moins un chemin partant de l'activité de départ et arrivant à l'arrivée de fin.

  Cette propriété évite aux flux de travaux d'avoir une structure dans laquelle une ou plusieurs activités ne pourraient jamais être exécutées.
- Pour finir, un graphe orienté "acyclique" est un graphe dans lequel il n'existe aucun cycle *orienté*. En termes de flux de travaux, cela signifie qu'il ne peut y avoir aucune boucle ad hoc (c'est-à-dire, des boucles construites à l'aide de transitions plutôt que des activités de début et de fin de boucle).

  Les boucles ad hoc peuvent sembler appropriées, mais (comme les instructions GOTO dans les langages de programmation) elles peuvent rendre la lecture et la compréhension d'un processus très difficiles. L'utilisation de constructions de boucle explicite rend les définitions de processus plus transparentes et plus compréhensibles.

  De plus, elle permet au moteur de savoir où la boucle peut intervenir, et peut ainsi suivre l'évolution du nombre d'itérations d'une boucle au cours de l'exécution.

Règles de structure de bloc

Comme indiqué précédemment, la manière dont les parenthèses sont utilisées dans une expression mathématique est semblable à la manière dont les "blocs" sont utilisés dans un flux de travaux. Rappel - il existe plusieurs types de blocs : choix, parallèle et boucle. Dans un flux de travaux Cúram :

- Toutes les constructions de début de bloc (fractionnement de choix, fractionnement parallèle ou activité de début de boucle) doivent se terminer par une construction de fin de bloc correspondante (jointure de choix, jointure parallèle ou activité de fin de boucle, respectivement).

Dans le cas des fractionnements et des jonctions, tous les chemins sortants d'un fractionnement doivent converger vers la jointure correspondante.

Rationale : La correspondance requise des fractionnements et des jointures (par exemple) permet d'améliorer la lisibilité. Dans une section contenant plusieurs chemins, cela permet de clairement indiquer s'il est possible de suivre un seul ou plusieurs chemins. Cela permet alors de savoir si une synchronisation est requise au niveau du point où les chemins fusionnent.

Si la correspondance n'était pas requise, il serait possible (voire facile) de modéliser les processus dont le blocage est garanti, ou ceux dans lesquels la fin du processus peut être atteinte avant que l'exécution de certaines activités soit terminée.

- Des blocs peuvent être imbriqués les uns dans les autres (par exemple, un fractionnement de choix à l'intérieur d'une boucle), mais ils ne peuvent pas être entrelacés (par exemple, aucune des transitions du fractionnement de choix ne peut accéder à une activité en dehors de la boucle).

  Cela permet d'éviter des situations dont le traitement est difficile à exécuter par le moteur et qui ne sont pas du tout intuitives pour les développeurs de flux de travaux.

  Prenons par exemple une boucle qui contient une jointure, où la jointure comporte deux transitions entrantes : une à partir d'une activité à l'intérieur de la boucle, l'autre à partir d'une activité en dehors de la boucle.

  Il est très difficile dans cette situation de déterminer comment la synchronisation de jointure doit fonctionner. Une transition entrante ne peut se déclencher qu'une seule fois, l'autre peut se déclencher plusieurs fois. Toutes les règles pour traiter une telle situation seraient arbitraires et elles ne seraient pas intuitives.

  Les flux de travaux définis à l'aide de blocs choix, parallèle et boucle possèdent une structure claire et simple dont la signification peut être comprise en un clin d'oeil.

Validations

Un flux de travaux Cúram valide doit représenter un graphe orienté, connecté et acyclique structuré en blocs. Ces propriétés (orienté, connecté, acyclique) sont pour la plupart distinctes, de sorte à pouvoir être vérifiées indépendamment par l'outil de définition de processus (PDT) avant de publier un processus. Les validations structurelles sont réalisées sur une définition de processus dans un ordre différent, décrit ci-après.

Vérifications syntaxiques simples

Le premier ensemble de validations structurelles effectuées sont les *vérifications syntaxiques simples*. Cette vérification permet de garantir que les liaisons et les divisions d'activités (voir «Fractionnement/Jointure», à la page 116) au sein de la définition de processus sont correctement définies. Ces validations sont les suivantes :

- Toutes les activités, à l'exception des activités de *début* et de *fin*, doivent posséder au moins une transition entrante et une transition sortante.
- Un type de liaison (différent de *NONE*) *doit* être spécifié pour toute activité dotée de plusieurs transitions entrantes.
- Un type de division (différent de *NONE*) *doit* être spécifié pour toute activité dotée de plusieurs transitions sortantes.
- Le type de liaison *NONE* doit être spécifié pour toute activité possédant *exactement* une transition entrante.

- Le type de division *NONE doit* être spécifié pour toute activité possédant *exactement* une transition sortante.
- Le type de division d'une activité `parallèle` doit être *NONE*.
- Le type de liaison d'une activité `parallèle` doit être *NONE*.
- Une activité `parallèle` doit posséder *exactement* une transition entrante.
- Une activité `parallèle` doit posséder *exactement* une transition sortante.
- Le type de division de l'activité à l'extrémité de la transition entrante se terminant à une activité `parallèle` *doit* être *NONE*.
- Le type de liaison de l'activité à l'extrémité de la transition sortante démarrant à une activité `parallèle` *doit* être *NONE*.

Vérifications de graphe

Le deuxième ensemble de validations structurelles effectuées sont les vérifications de *graphe*. Elles garantissent que le graphe de flux correspond à un graphe acyclique connecté et dirigé. Ces validations sont les suivantes :

- Le flux de travaux doit former un graphe 'connecté'. Cela signifie que chaque activité *doit* apparaître dans au moins un chemin entre l'activité de *début* et l'activité de *fin*.
- Le flux de travaux doit constituer un diagraphe acyclique. Cela signifie qu'*aucun* chemin via le flux de travaux n'impacte deux fois la même activité. Cette validation n'effectue la vérification que pour les cycles créés par les transitions (les cycles créés avec des activités de début de boucle et de fin de boucle sont parfaitement valides).
- Chaque sous-graphe d'instance dans le graphe de flux de travaux doit se terminer correctement. Cela signifie que tous les chemins possibles dans le flux de travaux, commençant par l'activité de *début*, doivent se terminer par l'activité de *fin*.

Vérifications des blocs

Le troisième ensemble de validations structurelles effectuées correspond aux vérifications des *blocs*. Cette vérification permet de s'assurer que le graphe de flux est correctement structuré par bloc.

Les constructions de début de bloc sont : activité de début de processus, activité de début de boucle, fractionnement parallèle (AND) et fractionnement de choix (XOR). Les constructions de fin de bloc correspondantes sont : activité de fin de processus, activité de fin de boucle, jointure parallèle (AND) et jointure de choix (XOR).

En partant de cela, les validations de structure de bloc suivantes sont exécutées :

- Pour chaque bloc de début, il doit y avoir un bloc de fin correspondant (par exemple, s'il existe une activité de début de boucle dans le flux de travaux, il doit exister une activité de fin de boucle correspondante).
- Les types de début/fin de bloc doivent correspondre (par exemple, s'il existe un fractionnement parallèle (AND) dans le graphe de flux de travaux, il doit exister une jointure parallèle (AND) correspondante.
- Les blocs peuvent être imbriqués, mais ne peuvent pas être entrelacés.

Services Web de flux de travaux

Présentation

Les flux de travaux Cúram peuvent interopérer avec d'autres systèmes de flux des travaux via la prise en charge d'aspects spécifiques de la norme *Business Process Execution Language* (BPEL) du groupe Oasis. Les processus BPEL peuvent adopter des processus de flux de travaux Cúram et être avertis une fois que le processus est terminé.

Le moteur de flux de travaux Cúram n'est pas censé être un moteur d'orchestration BPEL à part entière. Les flux de travaux Cúram devraient plutôt pouvoir participer aux processus orchestrés BPEL. Cette opération es effectuée en fournissant des fonctionnalités pour exposer les processus de flux des travaux de Cúram en tant que services web pouvant être appelés à partir des liens de partenaire de processus BPEL.

Exposition d'un service Web de flux de travaux

Les services Web de flux de travaux sont générés sur le support des services Web Cúram existant. Le moteur de flux de travaux nécessite particulièrement un objet de processus métier (BPO) modélisé sous la forme d'un service Web orienté document (pour plus de détails, voir le chapitre *Services Web entrants Cúram* du *Guide de référence de la modélisation Cúram*).

Le service Web BPO n'est qu'une vitrine pour l'API d'adoption de flux de travaux (`curam.util.workflow.impl.EnactmentService`). Ainsi, un seul objet de processus métier est requis par application. Un objet de processus métier approprié est déjà fourni dans l'application Cúram : `Logical View::MetaModel::Curam::Facades::Workflow::WebService::WorkflowProcessEnactment`.

Pour utiliser le service Web de flux de travaux, l'objet de processus métier nommé `Logical View::MetaModel::Curam::Facades::Workflow::WebService::WorkflowProcessEnactment` doit se voir affecter un composant serveur de stéréotype service Web.

Les services Web Cúram peuvent être personnalisés d'autres manières, par exemple en les sécurisant à l'aide de la sécurité de services Web, comme décrit dans le chapitre *Sécurisation des services Web* du *Guide de référence de la modélisation Cúram*. Toutes les personnalisations des services Web de flux de travaux doivent être apportées à cet objet de processus métier.

Remarque : Etant donné que tous les services Web de flux de travaux sont gérés par le même objet de processus métier, toute personnalisation aura un impact sur toutes les définitions de processus exposées en tant que services Web.

Adoption d'un processus

L'exposition d'une définition de processus de flux de travaux Cúram en tant que service Web nécessite uniquement de la marquer comme telle dans l'outil de définition de processus ou directement dans les métadonnées, comme décrit dans le «Métadonnées d'une définition de processus», à la page 13. Une fois que les définitions de processus ont été marquées comme services Web, le fichier EAR du serveur et le fichier EAR des services Web doivent être régénérés.

Comme pour les autres services Web Cúram, le code WSDL du service est uniquement accessible une fois que le fichier EAR des services Web a été déployé.

Le nom du service Web de flux de travaux est identique au nom du processus. Par conséquent, le code WSDL est accessible à partir d'une adresse URL identique à la suivante : http://testserver:9082/CuramWS/services/<NomProcessus>?wsdl

Le contenu du code WSDL est déterminé en partie par l'entrée du processus (les attributs d'objet de données de flux de travaux marqués comme "requis pour l'adoption") et sa sortie (les attributs d'objet de données de flux de travaux marqués comme sortie de processus) (voir «Métadonnées», à la page 19). Le type de port WSDL correspond au nom du processus et l'opération d'adoption d'un processus est toujours `startProcess`.

```
<wsdl:portType name="SomeCuramWorkflow">
    <wsdl:operation name="startProcess">
      <wsdl:input message="intf:startProcessRequest"
                  name="startProcessRequest"/>
      <wsdl:output message="intf:startProcessResponse"
                   name="startProcessResponse"/>
      <wsdl:fault message="intf:InformationalException"
                  name="InformationalException"/>
      <wsdl:fault message="intf:AppException"
                  name="AppException"/>
    </wsdl:operation>
  </wsdl:portType>
```

Figure 2. Type de port d'adoption de processus

Rappel à la fin d'un processus

Un système externe (probablement un processus BPEL, mais pas nécessairement) qui adopte un flux de travaux Cúram via les services Web devra souvent être averti de la fin du processus et nécessitera probablement certaines données de sortie provenant de la définition de processus. Ceci nécessite un service Web qui sera appelé à la fin du processus pour être spécifié pour chaque définition de processus.

Le service Web de rappel est spécifié dans les métadonnées de la définition de processus à l'aide de l'outil de définition de processus ou directement dans les métadonnées, comme décrit dans le «Métadonnées d'une définition de processus», à la page 13.

Remarque : Avant d'être utilisé dans une définition de processus de flux de travaux, le service Web de rappel doit être enregistré en tant que connecteur de service Web sortant Cúram, comme décrit dans le chapitre *Connecteurs de services Web sortants Cúram* du *Guide de référence de modélisation Cúram*.

Le service Web de rappel doit être implémenté par un système externe, mais doit être conforme à une définition de type de port spécifiée par le service Web de flux de travaux Cúram. La rubrique «Appel à partir de processus BPEL» contient plus de détails.

Appel à partir de processus BPEL

La création de processus BPEL qui adoptent les processus de flux de travaux Cúram n'est pas traitée dans le présent document. Toutefois, le code WSDL pour chaque service Web de processus de flux de travaux contient des informations pouvant être utilisées par les processus BPEL.

Type de port de rappel

Il existe dans le code WSDL un type de port pour un service Web de flux de travaux Cúram qui n'est pas implémenté par le service lui-même. Le

nom de ce type de port est le nom du processus auquel le suffixe "Complete" est ajouté (<NomProcessus>Complete).

L'objectif de ce type de port non implémenté est de définir l'interface de service Web qu'un service Web de flux de travaux Cúram s'attend à voir implémenter par le processus BPEL qui l'a adopté. Ce type de port doit être implémenté par le service Web de rappel configuré dans la définition de processus (voir «Rappel à la fin d'un processus», à la page 125).

```
<!--Implémenté par le processus BPEL-->
  <wsdl:portType name="SomeCuramWorkflowComplete">
    <wsdl:operation name="processCompleted">
      <wsdl:input message="intf:processCompletedRequest"
                  name="processCompletedRequest"/>
    </wsdl:operation>
  </wsdl:portType>
```

Figure 3. Type de port de rappel

Type de lien partenaire

D'un point de vue technique, la seule manière d'autoriser un processus de flux de travaux Cúram à participer à un processus orchestré BPEL est d'exposer le processus en tant que service Web. Il est toutefois possible d'ajouter certaines métadonnées afin d'aider le développeur de processus BPEL à définir les types de port impliqués dans le lien partenaire et les rôles qu'ils jouent.

La spécification BPEL permet aux types de lien partenaire d'être définis dans le code WSDL du service à appeler dans le lien partenaire à l'aide du mécanisme d'extension WSDL. Le code WSDL généré pour un service Web de flux de travaux Cúram définit le type de lien partenaire auquel il s'attend à participer et indique les types de port qui jouent chaque rôle.

```
<!--Type de lien partenaire-->
  <partnerLinkType name="CuramWorkflowPartnerLink"
    xmlns="http://schemas.xmlsoap.org/ws/2003/05/partner-link/">
    <role name="curamService">
      <portType name="tns1:SomeCuramWorkflow"/>
    </role>
    <role name="partnerService">
      <portType name="tns1:SomeCuramWorkflowComplete"/>
    </role>
  </partnerLinkType>
```

Figure 4. Extensions WSDL pour BPEL

Emplacements des fichiers

Présentation

Bien qu'il existe des utilitaires tels que l'outil de définition de processus (PDT) et d'autres interfaces utilisateur d'administration, les sorties de ces outils doivent souvent être exportées et la version doit être contrôlée. Bien sûr, ces fichiers externalisés doivent être replacés dans le système d'exécution lors de la génération ou de l'installation de Cúram. Le masque indiqué dans Cúram consiste à placer ces fichiers dans un dossier source prédéfini à partir duquel ils sont chargés dans la base de données (peut-être après pré-traitement). Ce chapitre décrit l'emplacement des fichiers source liés au flux de travaux.

Fichiers de définition de processus de flux de travaux

Les définitions de processus de flux de travaux (publiées et non publiées) peuvent être importées sur la table de base de données appropriée en utilisant la cible **build database** standard.

Ces définitions de processus de flux de travaux doivent être enregistrées dans des fichiers XML, dans un sous-répertoire `workflow` dans le répertoire du composant serveur Cúram approprié (par exemple, `...\EJBServer\components\core\workflow` pour le composant `core` ou `...\EJBServer\components\Appeal\workflow` pour le composant `Appeal`, etc.).

Chaque composant dans l'application Cúram peut disposer d'un répertoire de flux de travaux contenant les fichiers XML de définition de processus qui s'y rapportent. Tous les fichiers de définition de processus stockés dans ces répertoires de flux de travaux seront automatiquement importés lorsque la cible **build database** est exécutée. Si les fichiers de définition de processus ne sont pas valides ou si le nom et la version des définitions ne correspondent pas à ceux utilisés dans les noms de fichier, l'importation échoue.

Les fichiers XML de définition de processus de flux de travaux sur le système de fichiers doivent suivre une convention d'attribution de nom stricte. Celle-ci est la suivante : `Nom de processus_vVersion de processus.xml` où :

- `Nom de processus` représente le nom du processus de flux de travaux.
- `Version de processus` représente la version du processus de flux de travaux.

La même version d'une définition de processus peut exister dans plusieurs composants dans l'application Cúram. La version importée sera toujours extraite du composant dont la priorité d'ordre de composant est la plus élevée. La priorité d'ordre de composant est configurée à l'aide de la variable d'environnement COMPONENT_ORDER_PRECEDENCE.

Chaque définition de processus lorsqu'elle est importée se verra affecter un nouvel identificateur de définition de processus unique pour la base de données dans laquelle elle est importée. Le même identifiant unique sera affecté à différentes versions de la même définition de processus et une seule version non publiée d'une définition de processus peut être importée. Afin de gérer les définitions de processus de flux de travaux non valides chargées pendant l'exécution de la cible build database, des validations strictes sont présentes dans le moteur de flux de travaux. Celles-ci garantissent qu'une définition de processus de flux de travaux ne peut ni être chargée dans le cache de définition de processus, ni être exécutées, sauf si elle transmet tout d'abord toutes les validations de processus. Ces validations sont décrites dans les chapitres précédents de ce document.

Personnalisation des fichiers de définition de processus de flux de travaux

Création de nouveaux fichiers de définition de processus de flux de travaux : Tous les nouveaux fichiers de définition de processus de flux de travaux doivent être créés dans le sous-répertoire du flux de du répertoire `...\EJBServer\components\custom` . Pour créer un nouveau fichier de définition de processus, le composant PDT peut être utilisée pour créer la définition requise et entrer tous les détails. La définition peut alors être exportée vers un fichier ç l'aide de l'outil et placée dans l'emplacement spécifié ci-dessus.

Modification d'un fichier de définition de processus de flux de travaux existant : A l'aide du composant PDT, affichez la dernière version de la définition de processus qui nécessite une modification. Créer une nouvelle version de cette définition de processus à l'aide de l'outil. Apportez les modifications, validez-les et publiez le flux de travaux.

Exportez la définition de processus de flux de travaux nouvellement publiée à l'aide du composant PDT et placez-la dans le sous-répertoire du flux de travaux du répertoire `...\EJBServer\components\custom` .

Fichiers de définition d'événement

Les événements fournissent un mécanisme permettant aux composants de l'application Cúram couplés de façon souple de communiquer des informations sur les modifications d'état dans le système. Lorsqu'un module de l'application émet un événement, un ou plusieurs autres modules reçoivent une notification indiquant qu'un événement s'est produit pourvu que ces modules soient inscrits en tant que programmes d'écoute de cet événement. Pour utiliser cette fonctionnalité, il faut définir des événements, des codes d'application doivent émettre ces événements et certains gestionnaires d'événements doivent être définis et enregistrés en tant que programmes d'écoute de ces événements.

Les événements sont définis dans Cúram dans des fichiers XML, qui indiquent à la fois les classes d'événements et les types d'événement. Ces fichiers sont créés avec une extension `.evx` et sont placés dans les `événements` d'un composant Curam (par exemple `...EJBServer\components\core\events`) à partir duquel ils sont récupérés et traités par les scripts de génération.

Il existe deux types de sorties générées par la commande **`evgen`** ; les fichiers `.java` (pour les constantes de code qui utilisent des événements présentant moins d'erreurs) et les fichiers `.dmx` (scripts de base de données Cúram pour le chargement des définitions d'événement dans la base de données). Les artefacts Java générés à partir de fichiers d'événement fusionnés les sont placés dans le répertoire `/build/svr/events/gen/[package]` , où `[package]` est l'attribut du module spécifié dans le fichier de définition d'événement. Les scripts de base de données générés à partir de fichiers d'événement fusionnés sont placés dans le répertoire `/build/svr/events/gen/dmx` .

Le chapitre 10 du *Guide du développeur du serveur Cúram* fournit une description complète des événements et la façon dont ils peuvent être utilisés dans l'application Cúram.

Configuration

Présentation

La plupart des options de configuration ne sont pas présentes sur toutes les définitions de processus de flux de travaux. Elles sont plutôt spécifiques à chaque définition et, de ce fait, elles sont abordées dans la définition même de processus. Cela dit, il existe un nombre réduit de propriétés d'application qui affectent le système de gestion de flux de travaux dans son ensemble. Ce chapitre décrit ces propriétés.

Propriétés d'application

Les propriétés d'applications suivantes peuvent être définies dans le ficher Application.prx :

Nom de la propriété	Description
curam.custom. workflow.workresolver	*Objectif :* Le nom qualifié complet de la classe d'application qui implémente l'interface de rappel `WorkResolver`. Pour plus d'informations, voir «Stratégie d'allocation», à la page 68. *Type :* Chaîne *Valeur par défaut :* curam.core.sl.impl.DefaultWorkResolver
curam.workflow. automaticallyaddtasktousertasks	*Objectif :* Après la résolution des cibles d'allocation pour une tâche, si cette tâche est affectée à un seul utilisateur et que la valeur de cette propriété est définie sur yes/true, le système ajoute automatiquement cette tâche à la liste Mes tâches d'un utilisateur dans sa boîte de réception afin de lui permettre de travailler dessus. *Type :* Chaîne *Valeur par défaut :* NO
curam.custom.notifications. notificationdelivery	*Objectif :* Le nom qualifié complet de la classe d'application qui implémente l'interface de rappel `NotificationDelivery`. Pour plus d'informations, voir «Stratégie d'allocation de notification», à la page 106. *Type :* Chaîne *Valeur par défaut :* curam.core.sl.impl.NotificationDeliveryStrategy
curam.workflow.disable.audit. wdovalueshistory.before.activity	*Objectif :* La table d'audit des données WDO de l'instance du processus, 'WDOValuesHistory', est remplie par le moteur de flux de travaux à trois points distincts durant l'exécution d'une instance de ce processus de flux de travaux (avant l'exécution d'une activité, après l'exécution d'une activité et avant l'évaluation des transitions à partir d'une activité). Si elle est définie sur true, cette propriété permet de vérifier qu'aucune donnée n'est écrite sur la table d'audit des données WDO avant l'exécution d'une activité. *Type :* BOOLEAN *Valeur par défaut :* FALSE
curam.workflow.disable.audit. wdovalueshistory.after.activity	*Objectif :* La table d'audit des données WDO de l'instance du processus, 'WDOValuesHistory', est remplie par le moteur de flux de travaux à trois points distincts durant l'exécution d'une instance de ce processus de flux de travaux (avant l'exécution d'une activité, après l'exécution d'une activité et avant l'évaluation des transitions à partir d'une activité). Si elle est définie sur true, cette propriété permet de vérifier qu'aucune donnée n'est écrite sur la table d'audit des données WDO avant l'exécution d'une activité. *Type :* BOOLEAN *Valeur par défaut :* FALSE
curam.workflow.disable.audit.wdovalueshistory .transition.evaluation	*Objectif :* La table d'audit des données WDO de l'instance du processus, 'WDOValuesHistory', est remplie par le moteur de flux de travaux à trois points distincts durant l'exécution d'une instance de ce processus de flux de travaux (avant l'exécution d'une activité, après l'exécution d'une activité et avant l'évaluation des transitions à partir d'une activité). Si elle est définie sur true, cette propriété permet de vérifier qu'aucune donnée n'est écrite sur la table d'audit des données WDO avant l'évaluation des transitions à partir d'une activité. *Type :* BOOLEAN *Valeur par défaut :* FALSE

Nom de la propriété	Description
curam.custom.workflow.processcachesize	*Objectif :* Le moteur de flux de travaux met en mémoire cache les versions publiées des définitions de processus en mémoire (afin de réduire les frais généraux lors de la recherche de métadonnées). Cette propriété détermine le nombre maximal de versions de processus stockées dans la mémoire cache. Lorsque ce nombre a été atteint, le moteur va commencer à éjecter les versions de processus à partir de la mémoire cache, à l'aide d'une règle d'éjection de la version la moins utilisée récemment. Les modifications de la valeur de cette propriété seront appliquées la prochaine fois que le moteur de flux de travaux tente d'insérer une version de processus dans le cache. *Type :* entier *Valeur par défaut :* 250
curam.batchlauncher.dbtojms.notification.batchlaunchermode	Voir *Guide de traitement par lots Cúram, Section 5.3* pour plus d'informations.
curam.batchlauncher.dbtojms.notification.encoding	Voir *Guide de traitement par lots Cúram, Section 5.3* pour plus d'informations.
curam.batchlauncher.dbtojms.notification.host	Voir *Guide de traitement par lots Cúram, Section 5.3* pour plus d'informations.
curam.batchlauncher.dbtojms.messagespertransaction	Voir *Guide de traitement par lots Cúram, Section 5.3* pour plus d'informations.
curam.batchlauncher.dbtojms.notification.port	Voir *Guide de traitement par lots Cúram, Section 5.3* pour plus d'informations.

JMSLite

Introduction

JMSLite est un serveur Java Message Service (JMS) simple développé par Cúram qui s'exécute avec l'environnement de test basé sur RMI. Par conséquent, il peut s'exécuter à l'intérieur des environnements de développement intégrés (IDE) pris en charge.

Cela permet aux définitions de processus d'être testées dans un environnement de développement intégré, c'est-à-dire sans que l'application ne soit déployée sur un serveur EJB. Lorsqu'il est utilisé en conjonction avec l'outil de définition de processus, JMSLite permet aux développeurs de définir, de déployer et d'adopter des flux de travaux, le tout dans leur environnement de développement intégré.

Rôles de JMSLite

JMSLite est un serveur JMS qui implémente uniquement ces sections de la spécification JMS nécessaires pour prendre en charge les tests des flux de travaux Cúram basés sur l'environnement de développement intégré : à savoir, la messagerie transactionnelle, point à point. Cela signifie que JMSLite prend en charge les transactions ACID impliquant la base de données de l'application *et* les destinations de file d'attente de flux de travaux définies par l'infrastructure. Il ne prend pas en charge les files d'attente personnalisées (définies par l'application) ou le domaine de publication/abonnement (par exemple, les rubriques).

Par conséquent, JMSLite permet au service d'adoption de flux de travaux et au moteur de flux de travaux d'envoyer des messages JMS de manière asynchrone. Cela signifie que les appels d'application vers les API d'infrastructure relatives aux flux de travaux (par exemple, le service d'adoption et le service d'événement) ne sont pas bloqués. Les API transmettent des messages au moteur de flux de travaux, qui lance des instances de processus de manière asynchrone (il exécute, par exemple, des activités automatiques, crée et attribue des tâches, etc.).

Objectif de JMSLite

L'objectif de JMSLite est de rendre le comportement du moteur de flux de travaux dans un environnement de développement intégré de la façon la plus proche possible de la manière dont il se comporte lorsqu'il est déployé sur un serveur d'applications. Cela accroît la probabilité de détecter rapidement les problèmes (lors du test dans l'environnement de développement intégré) plutôt que tardivement (lors d'un test sur un serveur d'applications), réduisant ainsi les risques et les coûts.

Par exemple, envisagez la situation suivante : à supposer que WMS (en cours d'exécution dans un environnement de développement intégré) devait adopter des flux de travaux de façon *synchrone*.

Rappel : Dans la production, les flux de travaux sont adoptés de façon *asynchrone*, car ils sont censés être de longue durée (qui se compte en heures, en jours ou en semaines) par rapport aux opérations normales de l'utilisateur (qui se comptent en secondes ou en millisecondes).
A supposer également qu'un développeur devait écrire une méthode qui a adopté un flux de travaux d'approbation de dossier automatisé et a ensuite (immédiatement après l'appel au service d'adoption) essayé d'exploiter le résultat (par exemple, vérifier si le dossier a été automatiquement approuvé). Etant donné que l'environnement de test fonctionne d'une manière différente (de manière synchrone) à partir de l'environnement de production, le code fonctionnerait correctement dans le test, mais échouerait dans la production (il s'agit d'un exemple de bogue de "couplage temporelle").

Toutefois, étant donné que JMSLite s'exécute de façon asynchrone, ce problème devrait apparaître dans l'environnement de développement intégré de la même façon que sur un serveur d'applications, permettant ainsi au développeur de le détecter rapidement.

Utilisation de JMSLite

Le serveur JMSLite interroge les files d'attente et analyse tous les messages qu'elles contiennent. Ces messages lancent des appels du serveur JMSLite au serveur RMI qui est requis pour les tests basés sur l'environnement de développement intégré des méthodes Cúram (ou communément `StartServer`). Le serveur JMSLite est lancé en tant qu'unité d'exécution lors de l'appel du processus (`StartServer`) .
Etant donné que le serveur JMSLite achemine les messages vers le moteur de flux de travaux en cours d'exécution sur le serveur RMI, il est nécessaire de démarrer StartServer en mode débogage lors du débogage des méthodes de flux de travaux.

Débogage des flux de travaux

Normalement, les méthodes de l'infrastructure Cúram sont appelées par l'application. Toutefois, dans le flux de travaux, l'appel est souvent effectué à l'envers, c'est-à-dire le moteur de flux de travaux (infrastructure) appelle une méthode d'application (par exemple, une méthode d'allocation de travail). Dans ce cas, il n'est pas possible pour un développeur d'applications de passer à l'étape de l'appel à la méthode `curam.util.workflow.impl.EnactmentService.startProcess()` dans leur méthode d'application (Allocation de travail). Dans ce cas, le développeur doit définir des points d'arrêt dans la méthode qu'il souhaite déboguer et exécuter la méthode qui adopte le flux de travaux. Le moteur de flux de travaux appelle ensuite (en mode asynchrone) la méthode d'application,

entraînant ainsi la réalisation du point d'arrêt. Le débogueur suspend ensuite l'exécution au niveau du point d'arrêt spécifié, permettant ainsi un débogage normal.

Les méthodes d'application qui se trouvent dans la catégorie ci-dessus sont les suivantes :

- Méthodes d'activités automatiques
- Fonctions d'allocation de travail
- Méthode de livraison de la notification d'application
- Méthode du programme de résolution du travail d'application

Boîte de réception et gestion des tâches

Présentation

Les tâches sont utilisées pour affecter et suivre le travail des utilisateurs système et sont générées lorsque des activités «Manuel», à la page 61, «Décision», à la page 80 ou «Parallel», à la page 97 sont exécutées par le moteur de flux de travaux. La boîte de réception et les fonctions de gestion des tâches associées sont utilisées par les utilisateurs de l'application Cúram pour gérer ces tâches. Les sections suivantes décrivent les options de configuration et de personnalisation qui sont disponibles pour les zones Boîte de réception et Gestion des tâches de Cúram WMS.

Configuration de la boîte de réception

Paramètres de configuration des tailles de liste de la boîte de réception

Il existe un certain nombre de vues de liste de tâches disponibles dans la boîte de réception. Ces vues sont les suivantes :

- *Mes tâches ouvertes* : Liste des tâches sur lesquelles l'utilisateur travaille actuellement.
- *Mes tâches différées* : Liste des tâches sur lesquelles l'utilisateur travaille, mais qu'il a reporté à une date ultérieure.
- *Tâches disponibles* : Liste des tâches disponibles sur lesquelles l'utilisateur peut travailler.
- *Résultat de la recherche de requête de tâche* : Liste des tâches qui sont le résultat de l'exécution d'une requête de tâche.
- *Tâches de file d'attente des travaux* : Liste des tâches affectées à une file d'attente des travaux.

Il existe également une liste dans la boîte de réception qui affiche les notifications qui ont été distribuées à un utilisateur.

- *Mes notifications* : Liste des notifications qui ont été distribuées à l'utilisateur.

Les vues de la liste Boîte de réception peuvent être configurées pour limiter le nombre des enregistrements renvoyés à l'utilisateur. Les propriétés d'application suivantes peuvent être définies dans le fichier Application.prx afin d'appliquer cette modification.

Tableau 5. Paramètres de configuration des tailles de liste de la boîte de réception

Nom de la propriété	Description
curam.inbox.max.task.list.size	*Objectif :* La valeur de la propriété contrôle le nombre de tâches affichées dans les différentes vues de la liste des tâches de la boîte de réception. Les pages de la liste de tâches de la boîte de réception affectées par la valeur de cette propriété sont les suivantes : Mes tâches ouvertes ; Mes tâches différées ; Tâches disponibles ; Recherche de requête de tâches ; Tâches de la file d'attente de travaux. Si le nombre de tâches à afficher dépasse la valeur spécifiée, un message s'affiche informant l'utilisateur que tous les enregistrements qui correspondent aux critères de recherche de la page ne sont pas affichés. Ce message affiche à la fois le nombre de tâches affichées et le nombre total de tâches qui correspondent aux critères de recherche. *Type :* entier *Valeur par défaut :* 100
curam.notification.max.list.size	*Objectif :* La valeur de la propriété contrôle le nombre de notifications affichées dans la vue de la liste Mes notifications de la boîte de réception. Si le nombre de notifications à afficher dépasse la valeur spécifiée, un message s'affiche informant l'utilisateur que tous les enregistrements qui correspondent aux critères de recherche de la page ne sont pas affichés. Ce message affiche à la fois le nombre de notifications affichées ainsi que le nombre total de notifications qui correspondent aux critères de recherche. *Type :* entier *Valeur par défaut :* 100

Paramètres de configuration de l'obtention de la tâche suivante

Il existe un certain nombre de fonctions de raccourci disponibles dans la boîte de réception permettant d'extraire la tâche suivante sur laquelle travailler. Ces fonctions comprennent les éléments suivants :

- Obtenir la tâche suivante - extrait la tâche suivante à partir des tâches disponibles pour l'utilisateur.
- Obtenir la tâche suivante à partir de l'unité d'organisation préférée - extrait la tâche suivante affectée à l'unité d'organisation préférée de l'utilisateur.
- Obtenir la tâche suivante à partir de la file d'attente préférée - extrait la tâche suivante affectée à la file d'attente préférée de l'utilisateur.
- Obtenir la tâche suivante à partir de la file d'attente - extrait la tâche suivante affectée à une file d'attente sélectionnée par l'utilisateur.

L'algorithme utilisé par ces fonctions de raccourci permettant d'extraire la tâche suivante peut être configuré à l'aide des propriétés d'application suivantes dans le fichier Application.prx :

Tableau 6. Paramètres de configuration de l'obtention de la tâche suivante

Nom de la propriété	Description
curam.workflow. reservenexttaskwithpriorityfilter	*Objectif :* La valeur de la propriété contrôle si l'algorithme de l'obtention de la tâche suivante utilise la priorité d'une tâche afin de déterminer la tâche suivante à extraire. Si elle est définie sur *OUI*, qui est la valeur par défaut, la priorité de la tâche est utilisée à cet effet (priorités telles que définies dans la propriété curam.workflow.taskpriorityorder). Dans le cas contraire, la tâche qui doit être extraite est basée sur les tâches qui ont été affectées à l'utilisateur depuis le plus longtemps. *Type :* Chaîne *Valeur par défaut:* Oui
curam.workflow.taskpriorityorder	*Objectif :* Il existe trois priorités de tâche spécifiées dans le système de gestion de flux de travaux, à savoir Elevée, Moyenne et Faible (qui correspondent aux codes TP1, TP2 et TP3 dans la table de codes `TaskPriority`). Dans certains cas, les clients peuvent être dans l'obligation d'ajouter une nouvelle priorité de tâche (par exemple, critique avec une valeur de code de la table des codes de TP4). L'extraction des tâches à l'aide de la priorité de tâche contenant cette valeur permet ainsi de s'assurer que les tâches critiques apparaîtront après celles ayant une priorité basse (lorsque l'objectif est que les tâches avec cette priorité doivent être extraites en premier). Cette propriété permet aux priorités de tâche d'être spécifiées dans l'ordre nécessaire pour répondre aux exigences du client. *Type :* Chaîne *Valeur par défaut :* TP1,TP2,TP3

Paramètres de réacheminement de tâches et de blocage d'allocation

Le réacheminement de tâches permet à l'utilisateur de réacheminer des tâches vers un autre utilisateur, un objet d'organisation (unité d'organisation, poste ou emploi) ou une file d'attente de travaux pendant une durée spécifiée. Le blocage d'allocation des tâches permet à l'utilisateur de s'assurer qu'aucune tâche ne lui est affectée pendant une durée spécifiée. L'utilisateur peut accéder à cette fonctionnalité dans la zone Préférences de tâche de la Boîte de réception. Cependant, les utilisateurs sur le système peuvent ne pas tous avoir besoin d'accéder à la configuration des périodes de réacheminement de tâche ou de blocage d'allocation des tâches pour eux-mêmes. Pour faciliter cette exigence, ces zones de fonctionnalité dans la boîte de réception peuvent être désactivées pour des utilisateurs particuliers via l'utilisation d'identificateurs de sécurité. Le tableau suivant détaille les identificateurs de sécurité dont un utilisateur doit disposer pour bénéficier de cette fonctionnalité.

Tableau 7. Identificateurs de sécurité et actions associées

Nom de l'identificateur de sécurité	Action autorisée
UserTaskRedirection. listTaskRedirectionHistoryForUser	Permet à un utilisateur d'afficher toutes les périodes de réacheminement des tâches qui lui sont spécifiées.
UserTaskRedirection. redirectTasksForUser	Permet à un utilisateur de créer une période de réacheminement des tâches pour lui-même.
UserTaskRedirection. clearTaskRedirectionForUser	Permet à un utilisateur d'effacer l'une de ses périodes de réacheminement des tâches.
UserTaskAllocationBlocking.list. TaskAllocationBlockingHistoryForUser	Permet à un utilisateur d'afficher toutes les périodes de blocage d'allocation des tâches qui lui sont spécifiées.

Tableau 7. Identificateurs de sécurité et actions associées (suite)

Nom de l'identificateur de sécurité	Action autorisée
UserTaskAllocationBlocking. blockTaskAllocationForUser	Permet à un utilisateur de créer une période de blocage d'allocation des tâches pour lui-même.
UserTaskAllocationBlocking. clearTaskAllocationBlockForUser	Permet à un utilisateur d'effacer l'une de ses périodes de blocage d'allocation des tâches.

Personnalisation de la boîte de réception

Le comportement par défaut des fonctionnalités relatives aux actions de la boîte de réception, aux actions de la tâche et à la recherche de tâches peut être modifié à l'aide de Guice afin d'appeler un code personnalisé qui remplace le comportement par défaut.

Remarque : Guice est un cadre d'applications développé par *Google* et dépasse la portée de ce document. Pour plus d'informations sur Guice, consultez le guide d'utilisation de Guice.

Le système de gestion de flux de travaux Cúram contient les points de personnalisation suivants et leurs implémentations correspondantes par défaut :

Tableau 8. Points de personnalisation

Point de personnalisation	Classe d'interface	Classe d'implémentation par défaut
Actions de la boîte de réception	`curam.core.hook.task.impl.InboxActions`	`curam.core.hook.task.impl.InboxActionsImpl`
Actions de la tâche	`curam.core.hook.task.impl.TaskActions`	`curam.core.hook.task.impl.TaskActionsImpl`
Recherche de tâches et Recherche de tâches disponibles	`curam.core.hook.task.impl.SearchTask`	`curam.core.hook.task.impl.SearchTaskImpl`
Requête de tâche	`curam.core.hook.task.impl.TaskQuery`	`curam.core.hook.task.impl.TaskQueryImpl`
Génération SQL de recherche de tâches	`curam.core.hook.task.impl.SearchTaskSQL`	`curam.core.hook.task.impl.SearchTaskSQLImpl`

Les *actions de la boîte de réception* suivantes peuvent être personnalisées :

- Obtenir la tâche suivante
- Obtenir la tâche auprès de l'unité d'organisation préférée
- Obtenir la tâche suivante depuis la file d'attente préférée
- Obtenir la tâche suivante auprès de la file d'attente
- Abonner un utilisateur à la file d'attente des travaux
- Désabonner l'utilisateur de la file d'attente des travaux

Les *actions de la tâche* suivantes peuvent être personnalisées :

- Ajouter un commentaire
- Fermer
- Create
- Différer
- Redémarrer

- Réacheminer
- Modifier le temps travaillé
- Modifier la priorité
- Modifier l'échéance
- Réallouer
- Ajouter à Mes tâches

Les méthodes *Recherche de tâches* et *Recherche de tâches disponibles* suivantes peuvent être personnalisées :

- `countAvailableTasks`
- `countTasks`
- `searchAvailableTasks`
- `searchTask`
- `validateSearchTask`

Les méthodes *Requête de tâche* suivantes peuvent être personnalisées :

- `createTaskQuery`
- `modifyTaskQuery`
- `runTaskQuery`
- `validateTaskQuery`

Les méthodes de génération SQL de recherche de tâches suivantes peuvent être personnalisées. Ces méthodes sont utilisées pour générer le code SQL pour toutes les fonctionnalités de recherche de tâches ci-dessus.

- `getBusinessObjectTypeSQL`
- `getCategorySQL`
- `getCountSQLStatement`
- `getCreationDateSQL`
- `getDeadlineSQL`
- `getFromClause`
- `getOrderBySQL`
- `getOrgObjectSQL`
- `getPrioritySQL`
- `getReservedBySQL`
- `getRestartDateSQL`
- `getSelectClause`
- `getSQLStatement`
- `getStatusSQL`
- `getTaskIDSQL`
- `getWhereClause`

Comment personnaliser la boîte de réception

La description suivante permet de personnaliser l'action de la boîte de réception `curam.core.hook.task.impl.InboxActionsImpl.getNextTask`. Il est possible de suivre le même processus afin de personnaliser tous les autres points de personnalisation.

Une classe de point d'ancrage personnalisée doit être créée. Cette classe *doit* étendre la classe d'implémentation par défaut. Le diagramme ci-après affiche les

relations entre les classes :

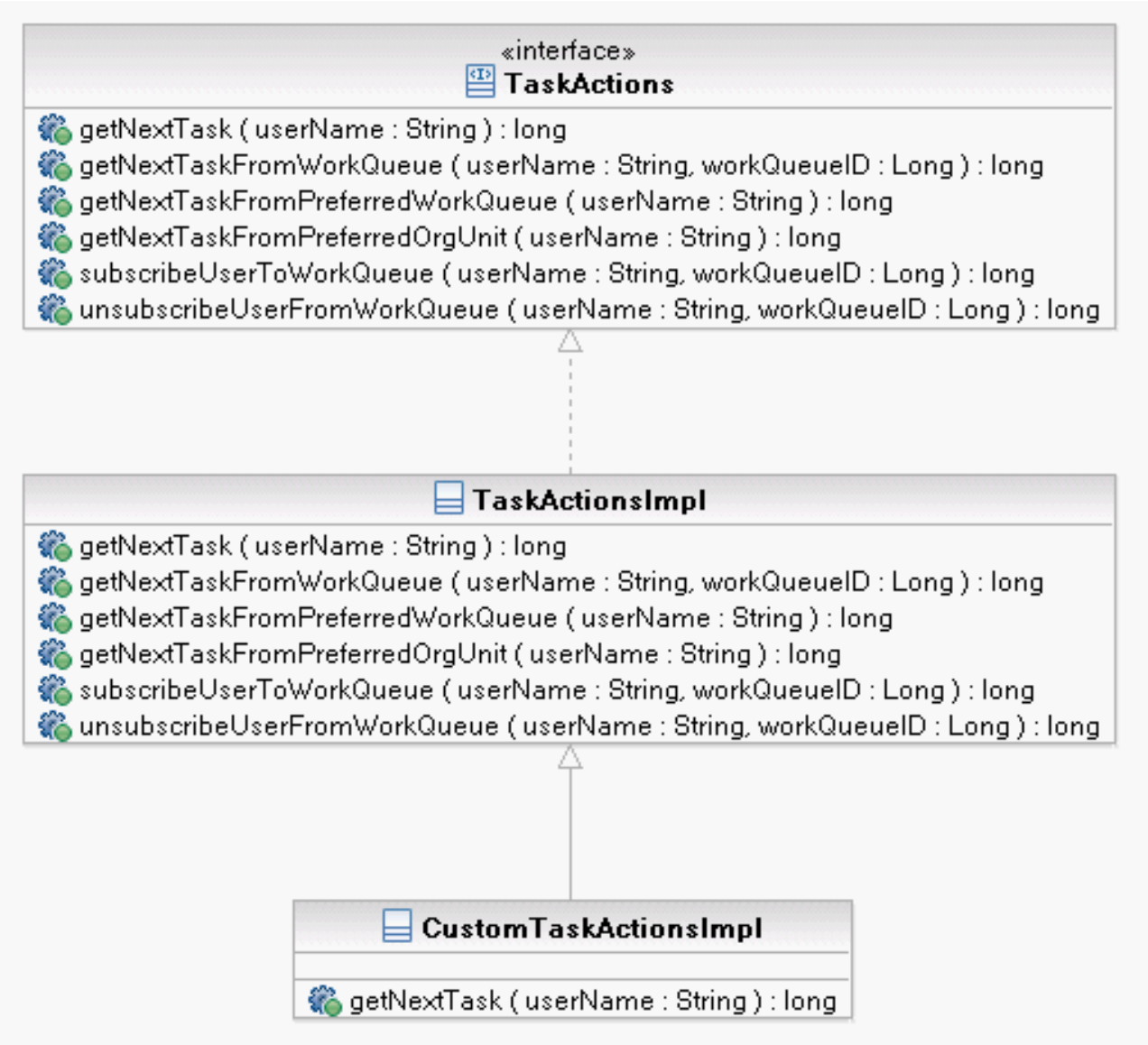

Figure 5. Diagramme de classe de personnalisation

Remarque : La classe personnalisée ne doit *jamais* directement implémenter la classe d'interface, car cela pourrait entraîner des exceptions de temps de compilation lors d'une mise à niveau si de nouvelles méthodes ont été ajoutées à l'interface. Dans ce cas, la classe personnalisée n'implanterait pas les nouvelles méthodes et le contrat entre la classe d'interface et la classe d'implémentation serait rompu, entraînant ainsi des exceptions de temps de compilation.

Personnalisation de l'implémentation par défaut : La signature de la fonction `getNextTask` sur l'interface `curam.core.hook.task.impl.InboxActions` est la suivante :

```
package curam.core.hook.task.impl;

@ImplementedBy(InboxActionsImpl.class)
public interface InboxActions {
```

```
  public long getNextTask(String userName);

   .
   .
   .
   .
}
```

L'implémentation par défaut de la fonction est définie dans la classe curam.core.hook.task.impl.InboxActionsImpl

```
package curam.core.hook.task.impl;

public class InboxActionsImpl implements InboxActions {

  public long getNextTask(String userName) {
    // Le code d'implémentation par défaut se trouve ici....
  }

  .
  .
  .
  .
}
```

Pour personnaliser getNextTask, la méthode doit être implémentée dans la nouvelle classe personnalisée créée précédemment qui étend la classe d'implémentation curam.core.hook.task.impl.InboxActionsImpl par défaut.

```
package custom.hook.task.impl;

public class CustomInboxActionsImpl extends InboxActionsImpl {

  public long getNextTask(final String userName) {
    // Le code de l'implémentation personnalisée doit être placé ici
  }

}
```

Afin de s'assurer que l'application exécute la nouvelle classe personnalisée au lieu de l'implémentation par défaut, une nouvelle classe custom.hook.task.impl.Module.java, qui étend com.google.inject.AbstractModule, doit être écrite avec la méthode configure implémentée comme dans l'exemple suivant :

```
package custom.hook.task.impl;

public class Module extends com.google.inject.AbstractModule {
  protected void configure() {
    bind(
      curam.core.hook.task.impl.InboxActions.class).to(
        custom.hook.task.impl.CustomInboxActionsImpl.class);
    }
}
```

Enfin, le nom de la classe custom.hook.task.impl.Module doit être inséré dans la colonne *ModuleClassName* de la table de base de données *ModuleClassName*. Celui-ci peut être inséré en ajoutant une ligne supplémentaire au fichier ModuleClassName.DMX ou directement dans la table de base de données, si nécessaire.

Grâce à cette approche, lorsque l'application est redéployée, le système appellera la version personnalisée de la fonction getNextTask au lieu de l'implémentation par défaut.

Remarques

Le présent document peut contenir des informations ou des références concernant certains produits, logiciels ou services IBM non annoncés dans ce pays. Pour plus de détails, référez-vous aux documents d'annonce disponibles dans votre pays, ou adressez-vous à votre partenaire commercial IBM. Toute référence à un produit, logiciel ou service IBM n'implique pas que seul ce produit, logiciel ou service puisse être utilisé. Tout autre élément fonctionnellement équivalent peut être utilisé, s'il n'enfreint aucun droit d'IBM. Il est de la responsabilité de l'utilisateur d'évaluer et de vérifier lui-même les installations et applications réalisées avec des produits, logiciels ou services non expressément référencés par IBM. IBM peut détenir des brevets ou des demandes de brevet couvrant les produits mentionnés dans le présent document. La remise de ce document ne vous accorde aucune licence pour ces brevets. Si vous désirez recevoir des informations concernant l'acquisition de licences, veuillez en faire la demande par écrit à l'adresse suivante :

IBM Director of Licensing

IBM Corporation

North Castle Drive

Armonk, NY 10504-1785

U.S.A.

Pour le Canada, veuillez adresser votre courrier à :

IBM Director of Commercial Relations

IBM Canada Ltd

3600 Steeles Avenue East

Markham, Ontario

L3R 9Z7 Canada

Les informations sur les licences concernant les produits utilisant un jeu de caractères double octet peuvent être obtenues par écrit à l'adresse suivante :

Intellectual Property Licensing

Legal and Intellectual Property Law.

IBM Japan Ltd.

19-21, Nihonbashi-Hakozakicho, Chuo-ku

Tokyo 103-8510, Japon

Le paragraphe suivant ne s'applique ni au Royaume-Uni, ni dans aucun autre pays dans lequel il serait contraire aux lois locales. LE PRESENT DOCUMENT EST LIVRE EN L'ETAT SANS AUCUNE GARANTIE EXPLICITE OU IMPLICITE. IBM DECLINE NOTAMMENT TOUT RESPONSABILITE RELATIVE A CES INFORMATIONS EN CAS DE CONTREFACON AINSI QU'EN CAS DE DEFAUT D'APTITUDE A L'EXECUTION D'UN TRAVAIL DONNE. Certaines juridictions n'autorisent pas l'exclusion des garanties implicites, auquel cas l'exclusion ci-dessus ne vous sera pas applicable.

Le présent document peut contenir des inexactitudes ou des coquilles. Ce document est mis à jour périodiquement. Chaque nouvelle édition inclut les mises à jour. IBM peut, à tout moment et sans préavis, modifier les produits et logiciels décrits dans ce document.

Les références à des sites Web non IBM sont fournies à titre d'information uniquement et n'impliquent en aucun cas une adhésion aux données qu'ils contiennent. Les éléments figurant sur ces sites Web ne font pas partie des éléments du présent produit IBM et l'utilisation de ces sites relève de votre seule responsabilité.

IBM pourra utiliser ou diffuser, de toute manière qu'elle jugera appropriée et sans aucune obligation de sa part, tout ou partie des informations qui lui seront fournies. Les licenciés souhaitant obtenir des informations permettant : (i) l'échange des données entre des logiciels créés de façon indépendante et d'autres logiciels (dont celui-ci), et (ii) l'utilisation mutuelle des données ainsi échangées, doivent adresser leur demande à :

IBM Corporation

Dept F6, Bldg 1

294 Route 100

Somers NY 10589-3216

U.S.A.

Ces informations peuvent être soumises à des conditions particulières, prévoyant notamment le paiement d'une redevance.

Le programme sous licence décrit dans ce document et tous les éléments sous licence associés sont fournis par IBM selon les termes de l'IBM Customer Agreement, de l'IBM International Program License Agreement ou de tout contrat équivalent.

Les données de performance indiquées dans ce document ont été déterminées dans un environnement contrôlé. Par conséquent, les résultats peuvent varier de manière significative selon l'environnement d'exploitation utilisé. Certaines mesures évaluées sur des systèmes en cours de développement ne sont pas garanties sur tous les systèmes disponibles. En outre, elles peuvent résulter d'extrapolations. Les résultats peuvent donc varier. Il incombe aux utilisateurs de ce document de vérifier si ces données sont applicables à leur environnement d'exploitation.

Les informations concernant des produits non IBM ont été obtenues auprès des fournisseurs de ces produits, par l'intermédiaire d'annonces publiques ou via d'autres sources disponibles.

IBM n'a pas testé ces produits et ne peut confirmer l'exactitude de leurs performances ni leur compatibilité. Elle ne peut recevoir aucune réclamation concernant des produits non IBM. Toute question concernant les performances de produits non IBM doit être adressée aux fournisseurs de ces produits.

Toute instruction relative aux intentions d'IBM pour ses opérations à venir est susceptible d'être modifiée ou annulée sans préavis et doit être considérée uniquement comme un objectif.

Tous les tarifs indiqués sont les prix de vente actuels suggérés par IBM et sont susceptibles d'être modifiés sans préavis. Les tarifs appliqués peuvent varier selon les revendeurs.

Ces informations sont fournies uniquement à titre de planification. Elles sont susceptibles d'être modifiées avant la mise à disposition des produits décrits.

Le présent document peut contenir des exemples de données et de rapports utilisés couramment dans l'environnement professionnel. Ces exemples mentionnent des noms fictifs de personnes, de sociétés, de marques ou de produits à des fins illustratives ou explicatives uniquement. Toute ressemblance avec des noms de personnes, de sociétés ou des données réelles serait purement fortuite.

LICENCE DE COPYRIGHT :

Ces informations contiennent des exemples de programmes d'application en langage source qui illustrent des techniques de programmation sur diverses plateformes d'exploitation. Vous avez le droit de copier, de modifier et de distribuer ces exemples de programmes sous quelque forme que ce soit et sans paiement d'aucune redevance à IBM, à des fins de développement, d'utilisation, de vente ou de distribution de programmes d'application conformes aux interfaces de programmation des plateformes pour lesquels ils ont été écrits ou aux interfaces de programmation IBM. Ces exemples de programmes n'ont pas été rigoureusement testés dans toutes les conditions. Par conséquent, IBM ne peut garantir expressément ou implicitement la fiabilité, la maintenabilité ou le fonctionnement de ces programmes. Les exemples de programmes sont fournis "EN L'ÉTAT", sans garantie d'aucune sorte. IBM décline toute responsabilité relative aux dommages éventuels résultant de l'utilisation de ces exemples de programmes.

Toute copie intégrale ou partielle de ces exemples de programmes et des oeuvres qui en sont dérivées doit inclure une mention de droits d'auteur libellée comme suit :

© (nom de votre société) (année). Des segments de code sont dérivés des exemples de programmes d'IBM Corp.

© Copyright IBM Corp. _année ou années_. All rights reserved.

Si vous visualisez ces informations en ligne, il se peut que les photographies et illustrations en couleur n'apparaissent pas à l'écran.

Politique de confidentialité

Les Logiciels IBM, y compris les Logiciels sous forme de services ("Offres Logiciels") peuvent utiliser des cookies ou d'autres technologies pour collecter des informations sur l'utilisation des produits, améliorer l'acquis utilisateur, personnaliser les interactions avec celui-ci, ou dans d'autres buts. Bien souvent,

aucune information personnelle identifiable n'est collectée par les Offres Logiciels. Certaines Offres Logiciels vous permettent cependant de le faire. Si la présente Offre Logiciels utilise des cookies pour collecter des informations personnelles identifiables, des informations spécifiques sur cette utilisation sont fournies ci-après.

Selon la configuration déployée, la présente Offre Logiciels peut utiliser des cookies de session et des cookies persistants destinés à collecter le nom et le mot de passe des utilisateurs pour les fonctions de gestion des session et d'authentification, pour faciliter l'utilisation des produits, pour la configuration de la connexion unique et/ou pour d'autres fonctions de suivi ou buts fonctionnels. Ces cookies ou d'autres technologies similaires ne peuvent pas être désactivés.

Si les configurations déployées de cette Offre Logiciels vous permettent, en tant que client, de collecter des informations permettant d'identifier les utilisateurs par l'intermédiaire de cookies ou par d'autres techniques, vous devez solliciter un avis juridique sur la réglementation applicable à ce type de collecte, notamment en termes d'information et de consentement.

Pour plus d'informations sur l'utilisation à ces fins des différentes technologies, y compris celle des cookies, consultez les Points principaux de la Déclaration IBM de confidentialité sur Internet à l'adresse http://www.ibm.com/privacy, la section "Cookies, pixels espions et autres technologies" de la Déclaration IBM de confidentialité sur Internet à l'adresse http://www.ibm.com/privacy/details, ainsi que la page "IBM Software Products and Software-as-a-Service Privacy Statement" à l'adresse http://www.ibm.com/software/info/product-privacy.

Marques commerciales

IBM, le logo IBM et ibm.com sont des marques ou des marques déposées d'International Business Machines Corp. dans de nombreux pays. Les autres noms de produits et services peuvent être des marques d'IBM ou d'autres sociétés. La liste actualisée de toutes les marques d'IBM est disponible sur la page Web "Copyright and trademark information" à http://www.ibm.com/legal/us/en/copytrade.shtml.

Apache est une marque d'Apache Software Foundation.

Java ainsi que tous les logos et toutes les marques incluant Java sont des marques d'Oracle et/ou de ses sociétés affiliées.

Les autres noms peuvent être des marques de leurs propriétaires respectifs. Les autres noms de sociétés, de produits et de services peuvent appartenir à des tiers.