IBM Cúram Social Program Management

Cúram Workflow - Referenzhandbuch

Version 6.0.5

IBM Cúram Social Program Management

Cúram Workflow - Referenzhandbuch

Version 6.0.5

Hinweis

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die Informationen in „Bemerkungen" auf Seite 145 gelesen werden.

Überarbeitung: 18. Mai 2013

Diese Ausgabe bezieht sich auf IBM Cúram Social Program Management v6.0.5 und alle nachfolgenden Releases, sofern nicht anderweitig in neuen Ausgaben angegeben.

Licensed Materials - Property of IBM.

© Copyright IBM Corporation 2012, 2013.

© Cúram Software Limited. 2011. Alle Rechte vorbehalten.

Inhaltsverzeichnis

© Copyright IBM Corp. 2012, 2013

Abbildungsverzeichnis

© Copyright IBM Corp. 2012, 2013

Tabellen

© Copyright IBM Corp. 2012, 2013

Kapitel 1. Einführung

1.1 Übersicht

Bei dem vorliegenden Dokument handelt es sich um das Referenzhandbuch zu Cúram Workflow, das eine detaillierte Erläuterung zu den Konzepten des Cúram-Workflow-Management-Systems (WMS) liefern soll. Es beschreibt, wie ein Prozess zum Erreichen bestimmter Ziele definiert werden sollte. Dazu werden die Workflowmetadaten sowie ihre Auswirkungen zur Laufzeit im Detail aufgeführt. Dieses Dokument ist nicht als Lerntext gedacht, sondern als umfassende Beschreibung aller in Cúram Workflow verfügbaren Funktionen.

1.2 Voraussetzungen

Dieses Dokument setzt gewisse Kenntnisse mit Workflowkonzepten sowie deren Umsetzung im Cúram-WMS voraus. Insbesondere wird vorausgesetzt, dass Sie wenigstens das Handbuch *Business Analyst Guides: Cúram Workflow Overview* gelesen haben.

1.3 Hinweise zur Verwendung dieses Handbuchs

Da dieses Dokument ein Referenzhandbuch ist, handelt es sich um die darin enthaltenen Kapitel um weitestgehend voneinander unabhängige Abschnitte. Ein Leser, der mit einem Konzept vertraut ist und seine Kenntnisse diesbezüglich vertiefen möchte, soll die Möglichkeit haben, das entsprechende Kapitel in diesem Handbuch zu finden und nur dieses Kapitel zu lesen. Obgleich nicht erwartet wird, dass dieses Dokument ganz durchgelesen wird, wurde es dennoch so strukturiert, dass es leicht verständlich und informativ ist.

Da sich einige Teile des Cúram-WMS stark aufeinander beziehen, spiegelt sich diese Tatsache auch in dieser Dokumentation wider. Es gibt zwei Formen dieser externen Referenzen: Voraussetzungen, die auf Informationen verweisen, die für das Verständnis eines aktuellen Abschnitts unerlässlich sind, und allgemeine Links, die auf verwandte (jedoch nicht erforderliche) Informationen verweisen.

1.4 Aufbau des Dokuments

Dieses Dokument ist in logische Abschnitte unterteilt, in denen jeweils ein Bereich des Cúram-WMS behandelt wird. In den folgenden Abschnitten werden diese logischen Abschnitte sowie die darin enthaltenen Kapitel beschrieben. Des Weiteren wird erläutert, welcher Bereich des Cúram-WMS in den jeweiligen Kapiteln abgedeckt wird.

1.4.1 Workflowprozesse

Im Abschnitt *Workflowprozesse* werden die Metadaten für eine Workflowprozessdefinition beschrieben. Zudem wird der Lebenszyklus einer Prozessdefinition dargelegt.

In Kapitel 2, „Erstellen eines Workflowprozesses", auf Seite 5 wird beschrieben, wie mithilfe des Cúram-Workflowsystems ein Workflowprozess erstellt und visualisiert werden kann. Außerdem wird erläutert, wie ein Prozess freigegeben wird und welche Auswirkungen dies auf die Versionierung für Prozessdefinitionen hat. Das Kapitel enthält zudem Informationen zum Importieren und Exportieren von Prozessdefinitionen sowie zur Lokalisierung von in einem Prozess enthalten Text. Weiter stehen Informationen zum Ausführen eines Workflowprozesses mithilfe der Cúram-Workflow-Engine zur Verfügung. Zudem werden die Methodenbibliothek sowie die Vorlagenbibliothek für Workflowdatenobjekte (siehe Kapitel 4, „Workflowdatenobjekte", auf Seite 17) näher beschrieben.

© Copyright IBM Corp. 2012, 2013

Kapitel 3, „Metadaten der Prozessdefinition", auf Seite 13 enthält Informationen zu den Metadaten für eine Workflowprozessdefinition. Dabei werden alle Metadatenfelder beschrieben und die Validierungen und Workflowdatenobjekte des Typs 'Context'für den ganzen Workflowprozess erläutert.

1.4.2 Datenfluss

Im Abschnitt *Datenfluss* des Dokuments wird beschrieben, wie Daten in einer Prozessinstanz gespeichert und bearbeitet werden. Insbesondere wird darauf eingegangen, wie Daten von außen (bei Prozesseinführung) und zwischen Aktivitäten und Übergängen innerhalb des Prozesses übermittelt werden.

In Kapitel 4, „Workflowdatenobjekte", auf Seite 17 sind die Objekte aufgeführt, die zur Verarbeitung der Daten in der Workflow-Engine verwendet werden. Zudem werden die Metadaten, aus denen sich Workflowdatenobjekte zusammensetzen, sowie deren zugehörigen Attribute ausführlich beschrieben. Außerdem wird auf Validierungen bezüglich der Erstellung und Bearbeitung von Workflowdatenobjekten eingegangen. Weiter enthält das Kapitel Informationen zu den Workflowdatenobjekten des Typs 'Context', die durch das Prozessdefinitionstool und die Workflow-Engine bereitgestellt werden.

In Kapitel 5, „Prozessumsetzung", auf Seite 25 wird erläutert, wie eine Prozessinstanz gestartet wird (d. h. die Durchführung der in der Prozessdefinition festgelegten Schritte). Es werden hierbei die Umsetzungsservice-API sowie die Metadaten der Umsetzungszuordnungen für die Umsetzung eines Prozesses beschrieben. Zudem werden zugehörige Validierungen und Codebeispiele genannt. Es ist darüber hinaus möglich, einen Prozess als Reaktion auf ein ausgelöstes Ereignis zu starten. Diese Möglichkeit ist in diesem Kapitel beschrieben. Das Kapitel enthält zudem eine detaillierte Beschreibung der Konfigurationsdaten für diese Aktion. Außerdem sind alle Validierungen aufgeführt, die beim Erstellen der Zuordnungen zwischen Ereignissen und Workflowprozessen durchgeführt werden.

1.4.3 Aktivitäten

Aktivitäten sind ein zentraler Bestandteil eines Workflowprozesses, da sie die Schritte für die Geschäftsverarbeitung für den Workflow darstellen. Es gibt verschiedene Aktivitätstypen, die vom Cúram-WMS unterstützt werden. Diese sind im Abschnitt *Aktivitäten* des vorliegenden Dokuments beschrieben. Da auch Benachrichtigungen für die verschiedenen Aktivitätstypen relevant sind, werden auch diese im Abschnitt zu den Aktivitäten erläutert.

Kapitel 6, „Basisaktivität", auf Seite 31 enthält Informationen zu den Metadaten, die für alle der unterstützten Aktivitätstypen im Cúram-Workflowsystem gelten. Zudem sind die Validierungen beschrieben, die beim Erstellen oder Ändern einer Aktivität durchgeführt werden. Des Weiteren werden einige der etwas einfacheren Aktivitätstypen erläutert, darunter die Weiterleitungsaktivität sowie die Start- und Endprozessaktivität.

Kapitel 7, „Automatisch", auf Seite 35 enthält Informationen zu den Metadaten für eine automatische Aktivität. Zudem werden die Eingabe- und Ausgabezuordnungen für die der automatischen Aktivität zugeordneten Methode erläutert. In diesem Kapitel werden auch die Validierungen beschrieben, die beim Erstellen oder Ändern der Metadaten für eine automatische Aktivität durchgeführt werden. Weiter enthält es Informationen zu den Workflowdatenobjekten `Context_Result` und `Context_Error`, die für die Verwendung in Übergängen von automatischen Aktivitäten zur Verfügung stehen.

Kapitel 8, „Event-Wait", auf Seite 49 enthält Informationen zu den Metadaten für eine Event-Wait-Aktivität. Dazu zählen die Liste der Ereignisse, die Fristdetails (einschließlich der Fristerinnerungen) für das Event-Wait sowie alle möglicherweise anzugebenen Ausgabezuordnungen. Außerdem werden die Validierungen beschrieben, die beim Erstellen oder Ändern von Event-Wait-Metadaten durchgeführt werden. Zudem enthält das Kapitel eine detaillierte Beschreibung der Laufzeitinformationen in Verbindung mit der Ausführung von Event-Wait-Aktivitäten durch die Workflow-Engine. Weiter enthält es Informationen zu den Workflowdatenobjekten `Context_Event` und `Context_Deadline`, die für die Verwendung in Übergängen von Event-Wait-Aktivitäten zur Verfügung stehen.

Kapitel 9, „Manuell", auf Seite 61 enthält Informationen zu den Metadaten für eine manuelle Aktivität. Dazu zählen Informationen zur manuellen Aufgabe, zur Zuteilungsstrategie, zu den Geschäftsobjektzuordnungen und zu dem der manuellen Aktivität zugeordneten Event-Wait. Außerdem werden die Validierungen beschrieben, die beim Erstellen oder Ändern der Metadaten für die manuelle Aktivität durchgeführt werden. Zudem enthält das Kapitel eine detaillierte Beschreibung der Laufzeitinformationen in Verbindung mit der Ausführung von manuellen Aktivitäten durch die Workflow-Engine. Abschließend wird das Workflowdatenobjekt `Context_Task` beschrieben, das in den verschiedenen Zuordnungen für eine manuelle Aktivität verwendet werden kann.

Kapitel 10, „Entscheidung", auf Seite 77 enthält Informationen zu den Metadaten für eine Entscheidungsaktivität. Dazu zählen Informationen zu Entscheidungsaufgaben (die den Informationen zur manuellen Aufgabe ähneln) und zu Multiple-Choice- und Freitextfragen. Zudem sind die Validierungen beschrieben, die beim Erstellen oder Ändern der Aufgabe oder Frage für eine Entscheidungsaktivität durchgeführt werden. Das Kapitel liefert zudem eine Beschreibung der Laufzeitinformationen, die nach dem Ausführen einer Entscheidungsaktivität durch die Workflow-Engine verfügbar sind. Außerdem wird das Workflowdatenobjekt `Context_Decision` näher beschrieben.

Kapitel 11, „Subflow", auf Seite 87 enthält Informationen zu den Metadaten für eine Subflowaktivität. Dazu zählen Einzelheiten zu dem Subflowprozess für die Subflowaktivität und zu allen Eingabezuordnungen, die für die Umsetzung dieses Subflowprozesses erforderlich sind. Zudem werden verschiedene Validierungen beschrieben, die beim Erstellen oder Ändern der Metadaten durchgeführt werden.

Kapitel 12, „Schleifenbeginn und Schleifenende", auf Seite 91 enthält Informationen zu den Metadaten für eine Schleifenbeginn- und eine Schleifenendaktivität. Es werden der Schleifentyp, die Schleifenbedingung und die Schleifenendaktivitätsreferenz einer Schleifenbeginnaktivität erläutert. Das Kapitel liefert zudem eine Beschreibung der Laufzeitinformationen, die beim Ausführen einer Schleife in einer Workflowprozessdefinition durch die Workflow-Engine verfügbar sind. Außerdem wird das Workflowdatenobjekt `Context_Loop` näher beschrieben.

Kapitel 13, „Parallel", auf Seite 95 enthält Informationen zu den Metadaten für eine parallele Aktivität. Parallele Aktivitäten umschließen vorhandene Aktivitätstypen, wie manuelle Aktivitäten (siehe Kapitel 9, „Manuell", auf Seite 61) und Entscheidungsaktivitäten (siehe Kapitel 10, „Entscheidung", auf Seite 77). Da die diesen Aktivitätstypen zugeordneten Metadaten gleich bleiben, werden sie in diesem Kapitel nicht noch einmal beschrieben. Außerdem werden die Validierungen aufgeführt, die beim Erstellen oder Ändern der Metadaten für die parallele Aktivität durchgeführt werden. Zudem enthält das Kapitel eine detaillierte Beschreibung der Laufzeitinformationen in Verbindung mit der Ausführung von parallelen Aktivitäten durch die Workflow-Engine. Abschließend wird das Workflowdatenobjekt `Context_Parallel` beschrieben, das in den verschiedenen Zuordnungen für eine parallele Aktivität verwendet werden kann.

Kapitel 14, „Aktivitätsbenachrichtigungen", auf Seite 101 enthält Informationen zu den Metadaten für eine Aktivitätsbenachrichtigung. Zu diesen Informationen zählen die Bereitstellungsmechanismen, der Betreff, der Benachrichtigungstext sowie die Zuteilungsstrategie und Aktionen für die Benachrichtigung. Zudem werden verschiedene Validierungen beschrieben, die beim Erstellen oder Ändern der Benachrichtigungsmetadaten durchgeführt werden. Das Kapitel liefert zudem eine Beschreibung der Laufzeitinformationen, die beim Erstellen einer Benachrichtigung durch die Workflow-Engine verfügbar sind. Außerdem enthält es eine Reihe von Informationen zur Implementierung, die in der Cúram-Anwendung benötigt werden, damit Benachrichtigung ordnungsgemäß bereitgestellt werden können.

1.4.4 Ablaufsteuerung

Ein Workflowprozess stellt modellhaft den Informationsfluss innerhalb einer Organisation dar und führt durch die Schritte, die von einer Person oder einer Computersoftware zum Erreichen eines Geschäftsziel durchgeführt werden. Im Abschnitt *Ablaufsteuerung* des Dokuments wird beschrieben, wie ein solcher Informationsfluss (zwischen Aktivitäten) im Cúram-WMS angegeben und verwaltet wird.

In Kapitel 15, „Übergänge", auf Seite 109 werden die Verknüpfungen zwischen Aktivitäten beschrieben. Außerdem enthält das Kapitel eine detaillierte Auflistung der Metadaten für Übergänge. Außerdem wird auf Validierungen bezüglich der Erstellung und Bearbeitung von Übergängen eingegangen. Zudem werden Laufzeitinformationen bezüglich der Verarbeitung von Übergängen durch die Workflow-Engine bereitgestellt.

In Kapitel 16, „Bedingungen", auf Seite 113 wird das Prozessdefinitionsmetadatenkonstrukt beschrieben, das eine Bedingung darstellt. Außerdem wird auf Validierungen bezüglich der Erstellung und Bearbeitung von Bedingungen eingegangen.

In Kapitel 17, „Splits/Joins (Verzweigungs-/Synchronisationspunkte)", auf Seite 119 sind die Metadaten für Splits und Joins von Aktivitäten aufgeführt sowie Informationen zu ihrer Verwendung und zu den verschiedenen verfügbaren Typen.

In Kapitel 18, „Workflowstruktur", auf Seite 121 wird die Struktur eines Workflowprozesses beschrieben, die auf den Aktivitäten im Prozess und den Übergängen zwischen diesen Aktivitäten basiert. Die beim Erstellen einer Prozessdefinition vorhandenen Einschränkungen, mit denen sichergestellt wird, dass die Definition eine gültige Blockstruktur aufweist, sind ebenfalls aufgeführt. Außerdem werden die Validierungen erläutert, die im Rahmen dieser Einschränkungen durchgeführt werden.

1.4.5 Entwicklung und Laufzeit

Im Abschnitt *Entwicklung und Laufzeit* dieses Dokuments werden die Besonderheiten der Entwicklungs- und Laufzeitumgebungen für Cúram-Workflows beschrieben. Dabei wird insbesondere auf das Ausführen und Konfigurieren von Workflows sowie die zugehörige Fehlerbehebung eingegangen.

Kapitel 19, „Workflow-Web-Services", auf Seite 127 enthält eine Beschreibung der Schritte, die für die Prozesseinführung über Web-Services erforderlich sind, indem ein Cúram-Workflowprozess als Web-Service bereitgestellt wird.

In Kapitel 20, „Dateipositionen", auf Seite 131 wird erläutert, wohin die verschiedenen Ausgaben von Dienstprogrammen wie dem Prozessdefinitionstool und anderer Administrationsbenutzerschnittstellen exportiert werden und wo die Versionskontrolle stattfindet. Zu diesem Ausgaben zählen Metadatendateien der Prozessdefinition sowie Quellendateien, die Ereignissen zugeordnet sind.

In Kapitel 21, „Konfiguration", auf Seite 133 werden die Anwendungseigenschaften für den Workflow beschrieben, darunter deren Namen, die Standardeinstellungen und ihr Verwendungszweck im Cúram-Workflowsystem.

Kapitel 22, „JMSLite", auf Seite 135 enthält Informationen zum Cúram-JMSLite-Server, der neben der RMI-Testumgebung in einer unterstützten integrierten Entwicklungsumgebung (Integrated) ausgeführt werden kann. Außerdem sind die erforderlichen Schritte zum Starten des JMSLite-Servers aufgeführt. Zudem wird im Detail erläutert, wie mithilfe von JMSLite eine Fehlerbehebung für Workflows durchgeführt werden kann.

1.4.6 Konfiguration und Anpassung des Posteingangs

Im Abschnitt *Konfiguration und Anpassung des Posteingangs* werden die Konfigurations- und Anpassungsoptionen beschrieben, die im Abschnitt "Posteingang und Aufgabenverwaltung" des Cúram-WMS verfügbar sind. Insbesondere wird erläutert, wie die Anzahl der Aufgaben konfiguriert wird, die in den verschiedenen Listen im Posteingang angezeigt werden, und wie die verschiedenen Aktionen für Posteingang und Aufgabenverwaltung angepasst werden, die im System verfügbar sind.

In Kapitel 23, „Posteingang und Aufgabenverwaltung", auf Seite 137 werden die verfügbaren Konfigurationsoptionen für den Posteingang beschrieben. Außerdem wird erläutert, wie die verfügbaren Funktionen für Posteingang und Aufgabenverwaltung über das Google Guice-Framework angepasst werden können.

Kapitel 2. Erstellen eines Workflowprozesses

2.1 Lebenszyklus einer Prozessdefinition

Die Prozessdefinition ist ein zentrales Konzept in Workflowsystemen; daher ist die Art ihrer Erstellung und Verwendung von großer Wichtigkeit für den gesamten Prozess. In diesem Kapitel werden die vom Cúram-Workflowsystem bereitgestellten Möglichkeiten und Funktionen zum Erstellen und Verwalten von Prozessdefinitionen beschrieben.

2.1.1 Prozesserstellung

Das Cúram-Workflowsystem stellt ein *Prozessdefinitionstool* (PDT) zum Erstellen und Verwalten von Prozessdefinitionen bereit, die dann von der Workflow-Engine interpretiert werden können. Im Rahmen der Erstellung von Prozessdefinitionen wird mithilfe des Prozessdefinitionstools das gewünschte Prozessverhalten in Bezug auf Aktivitäten und Übergänge beschrieben.

Im Umfang des Prozessdefinitionstools sind einige Dienstprogramme für die Prozesserstellung enthalten. Mithilfe des Prozessdefinitionstools kann eine Prozessdefinition während der Entwurfsphase visualisiert werden. Außerdem können Prozesse kopiert, importiert und exportiert werden.

2.1.2 Prozessvisualisierung

Das Prozessdefinitionstool umfasst ein schreibgeschütztes grafisches Dienstprogramm, mit dem Prozessadministratoren Prozesse beim Erstellen oder Ändern visualisieren können. Es können alle Aktivitäten und Übergänge in einer Prozessdefinition angezeigt werden. Außerdem bietet das Tool eine allgemeine Ansicht aller möglicher Wege durch den Workflowprozess während der Ausführung. Nachfolgend finden Sie ein Beispiel einer grafischen Darstellung einer Workflowprozessdefinition.

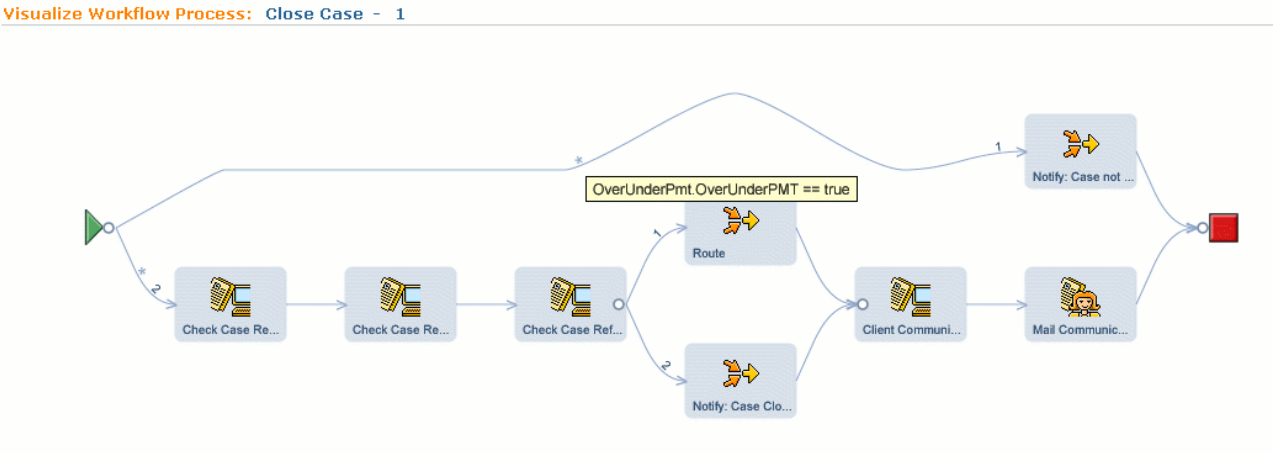

Abbildung 1. Visualisierung der Workflowprozessdefinition für 'Close Case' (Fall abschließen)

Der visualisierte Prozess besteht aus einer Reihe von Knoten in einer Grafik, die die Aktivitäten im Prozess darstellen. Die Knoten sind durch die in der Prozessdefinition festgelegten Übergänge miteinander verknüpft. Durch das Klicken auf eine Aktivität in der Grafik werden die Details der Aktivität im PDT angezeigt. Entsprechend werden beim Klicken auf einen Übergang zwischen Aktivitäten in der Grafik die Details zu dem Übergang im PDT aufgerufen.

Das Grafiktool zeigt die folgenden Informationen für jeden visualisierten Prozess an:

© Copyright IBM Corp. 2012, 2013

- Den Typ und Namen jeder Aktivität. Jeder Aktivitätstyp ist durch ein bestimmtes Symbol gekennzeichnet.
- Die für jede Aktivität definierten Benachrichtigungen (siehe Kapitel 14, „Aktivitätsbenachrichtigungen", auf Seite 101). Wenn eine Aktivität über eine zugeordnete Benachrichtigung verfügt, wird diese als Umschlag dargestellt, über den die entsprechende Seite mit der Aktivitätsbenachrichtigung aufgerufen werden kann.
- Der Split-/Join-Typ (siehe Kapitel 17, „Splits/Joins (Verzweigungs-/Synchronisationspunkte)", auf Seite 119) für jede Aktivität. Der Split- oder Join-Typ 'Auswahl' für eine Aktivität wird als Kreis dargestellt, während der Split- oder Join-Typ 'Parallel' als Quadrat angezeigt wird.
- Die Übergänge zwischen Aktivitäten. Wenn ein Übergang zwischen Aktivitäten über eine zugeordnete Übergangsbedingung (siehe Kapitel 16, „Bedingungen", auf Seite 113) verfügt, ist dies durch einen Stern gekennzeichnet. Die Details der Bedingung werden durch Bewegen des Cursor über diesen Stern angezeigt.
- Die Anordnung der einzelnen Split-Typen 'Auswahl' (siehe Kapitel 17, „Splits/Joins (Verzweigungs-/Synchronisationspunkte)", auf Seite 119) von einer Aktivität. Da die Anordnung eines Splits des Typs 'Auswahl' von einer Aktivität wichtig ist (der erste zulässige Übergang in der Liste wird verfolgt), wird die Abfolge der einzelnen Übergänge von einer Aktivität als Zahl für den entsprechenden Übergang angezeigt.

2.1.3 Freigeben eines Prozesses

Sobald eine Prozessdefinition erstellt wurde und verwendet werden kann, muss sie freigegeben werden, bevor sie von der Workflow-Engine ausgeführt werden kann (siehe 2.2, „Prozessausführung", auf Seite 8). Bei der Freigabe eines Prozesse mithilfe des PDTs wird dieser untersucht, um sicherzustellen, dass alle von der Workflow-Engine benötigten Informationen für die Ausführung des Prozesses vorhanden und in sich konsistent sind. Die Validierungen, die für die Freigabe eines Prozesses erforderlich sind, sind in den verschiedenen Abschnitten zu den Metadaten in diesem Dokument beschrieben.

Nur Prozesse, die entsprechend umfassend validiert wurden, können freigegeben und somit für die Workflow-Engine verfügbar gemacht werden. Sobald eine Prozessdefinition freigegeben wurde, ist sie schreibgeschützt und kann nur noch durch Erstellen einer neuen Version mit dem Prozessdefinitionstool bearbeitet werden.

2.1.4 Prozessversionen (Prozessbearbeitung)

Für einen freigegebenen Prozess können mit der Zeit Änderungen erforderlich werden. Da ein freigegebener Prozess jedoch schreibgeschützt ist, muss eine neue Version erstellt werden, bevor Änderungen an dem Prozess vorgenommen werden können. Beim Versuch, einen freigegebenen Prozess im PDT zu bearbeiten, wird automatisch eine neue nicht freigegebene Version dieses Prozesses erstellt.

Es kann zu jedem Zeitpunkt immer nur eine nicht freigegebene Version eines Prozesses existieren. Wenn der Administrator einen freigegebenen Prozess bearbeiten möchte, müssen alle vorhandenen nicht freigegebenen Versionen zunächst freigegeben oder gelöscht werden.

2.1.5 Importieren, Exportieren und Kopieren von Prozessen

Mit der Import- und Exportfunktion können Prozessentwickler Prozessdefinitionen wie gewünscht verschieben. Beispielsweise kann eine Prozessdefinition im Entwicklungssystem erstellt und erst dann in ein Produktionssystem verschoben werden, wenn sie erfolgreich getestet wurde.

Beim Export eines Prozesses werden die Prozessmetadaten in das Dateisystem exportiert. Diese Metadaten können dann mithilfe der Option zum Importieren von Prozessen des Prozessdefinitionstools importiert werden. Ein auf diese Weise importierter Prozess erhält die höchste verfügbare Versionsnummer und wird beim Import ungeachtet seines Status "Freigegeben" in einen nicht freigegebenen Status versetzt. Dadurch wird sichergestellt, dass importierte Prozessdefinitionen denselben Freigabevalidierungen unterzo-

gen werden wie andere lokal entwickelte Definitionen. Beim Import ist eine Überschreibungsoption verfügbar, damit alle vorhandenen nicht freigegebenen Versionen des Prozesses mit der importierten Version überschrieben werden können.

Es kann Situationen geben, in denen sich eine Prozessdefinition nur leicht von einer anderen Definition im Workflowsystem unterscheidet. Es steht eine Option zum Kopieren zur Verfügung, mit der bei Bedarf ein vorhandener Prozess zu einem neuen Prozess kopiert werden kann. Der neue Prozess wird beim Kopieren mit der Version 1 immer in einen nicht freigegebenen Status versetzt, unabhängig vom Status des ursprünglichen Prozesses.

2.1.5.1 Validierungen

- Eine Prozessdefinition kann nicht importiert werden, wenn eine nicht freigegebene Version eines Prozesses bereits mit demselben Namen vorhanden ist und die Überschreibungsoption nicht ausgewählt wurde.
- Eine Prozessdefinition kann nicht importiert werden, wenn kein Name für diesen Prozess angegeben wurde.
- Eine Prozessdefinition kann nicht importiert werden, wenn bereits ein Prozess mit demselben Namen und einer anderen Prozesskennung vorhanden ist. Diese Validierung stellt sicher, dass eine importierte Definition eine vorhandene Prozessdefinition unabdingbar überschreiben kann, es sei denn, die Prozesskennungen stimmen überein.
- Beim Kopieren eines vorhandenen Prozesses muss der Name des neuen Prozesses innerhalb des Workflowsystems eindeutig sein.
- Die Länge des Namens der zu importierenden Workflowprozessdefinition darf die maximal zulässige Länge für einen solchen Namen nicht überschreiten. Die zulässige Länge beträgt 254 Zeichen.
- Die Länge des Namens eines Workflowdatenobjekts, das in der zu importierenden Workflowprozessdefinition enthalten ist, darf die maximal zulässige Länge für einen solchen Namen nicht überschreiten. Die zulässige Länge beträgt 75 Zeichen.
- Die Länge des Namens eines Workflowdatenobjekt-Attributs, das in der zu importierenden Workflowprozessdefinition enthalten ist, darf die maximal zulässige Länge für einen solchen Namen nicht überschreiten. Die zulässige Länge beträgt 75 Zeichen.
- Alle Codetabellenwerte, die in der zu importierenden Workflowprozessdefinition enthalten sind, müssen gültige Werte sein (d. h., die Codetabelle muss vorhanden sein und den angegebenen Code enthalten).
- Text für die Standardländereinstellung muss für alle angegebenen lokalisierbaren Textzeichenfolgen in der zu importierenden Prozessdefinition vorhanden sein.
- Die Kennungen für Aktivitäten, Übergänge, Übergangsbedingungsausdrücke, Schleifenbedingungsausdrücke, Ereignisse und Erinnerungen müssen in der zu importierenden Workflowprozessdefinition eindeutig sein.

2.1.6 Lokalisierung

Workflowprozessdefinitionen enthalten Metadatentext, der in vielen verschiedenen Sprachen für unterschiedliche Benutzer zur Verfügung stehen muss. Beim Ausführen einer manuellen Aktivität wird beispielsweise eine Aufgabe erstellt, die einen zugehörigen Betreff hat. Mit dem Prozessdefinitionstool kann der Prozessentwickler diese Betreffzeichenfolge für jede von der Anwendung unterstützte Ländereinstellung lokalisieren.

Lokalisierbare Zeichenfolgen können in einer Prozessdefinition durch die in 6.2.1, „Lokalisierter Text", auf Seite 32 aufgeführten Metadaten angegeben werden. Alle lokalisierbaren Textzeichenfolgen, die in einer Prozessdefinition angegeben wurden, müssen einen Eintrag für die standardmäßig verwendete Serverländereinstellung enthalten. Diese Serverländereinstellung wird in der Cúram-Anwendung durch die Eigenschaft 'curam.environment.default.locale' angegeben. Standardmäßig verwendet das PDT diese Länderein-

stellung beim Hinzufügen von lokalisierten Zeichenfolgen zu einer Prozessdefinition. Alle anderen erforderlichen Ländereinstellungen müssen mithilfe der bereitgestellten Lokalisierungsoption hinzugefügt werden.

Nachfolgend sind die lokalisierbaren Textzeichenfolgen aufgeführt, die in einer Prozessdefinition angegeben werden können.

- Prozessbeschreibung
- Anzeigename des Workflowdatenobjekts
- Beschreibung des Workflowdatenobjekts
- Anzeigename des Workflowdatenobjekt-Attributs
- Aktivitätsname
- Aktivitätsbeschreibung
- Aufgabe der manuellen Aktivität, Nachricht
- Aufgabenaktion der manuellen Aktivität, Nachricht
- Aufgabe der parallelen manuellen Aktivität, Nachricht
- Aufgabenaktion der parallelen manuellen Aktivität, Nachricht
- Aktion der Entscheidungsaktivität, Nachricht
- Frage der Entscheidungsaktivität, Nachricht
- Sekundäre Aktion der Entscheidungsaktivität, Nachricht
- Antwort der Entscheidungsaktivität, Anzeigewert
- Aktion der parallelen Entscheidungsaktivität, Nachricht
- Frage der parallelen Entscheidungsaktivität, Nachricht
- Sekundäre Aktion der parallelen Entscheidungsaktivität, Nachricht
- Antwort der parallelen Entscheidungsaktivität, Anzeigewert
- Benachrichtigungsbetreff der Aktivität, Nachricht
- Benachrichtigungstext der Aktivität, Nachricht
- Benachrichtigungsaktion der Aktivität, Nachricht
- Betreff der Erinnerungsbenachrichtigung, Nachricht
- Text der Erinnerungsbenachrichtigung, Nachricht
- Aktion der Erinnerungsbenachrichtigung, Nachricht

Die API `LocalizableStringResolver` stellt Routinen bereit, die die verschiedenen lokalisierbaren Zeichenfolgen für Aufgaben und Benachrichtigungen auflöst und zurückgibt, die in einer Workflowprozessdefinition für die Ländereinstellung des aktuellen Benutzers vorhanden sind. Wenn eine Textzeichenfolge für die aktuelle Benutzerländereinstellung nicht lokalisiert wurde, wird stattdessen der Text für die Standardserverländereinstellung zurückgegeben.

2.2 Prozessausführung

Eine Workflowprozessdefinition beschreibt die Aufgaben und den Ablauf eines Geschäftsprozesses so, dass sie vom Cúram-Workflow-Management-System verstanden werden. Um die in der angegebenen Prozessdefinition beschriebene Arbeit durchzuführen, muss von der Workflow-Engine eine entsprechende Prozessinstanz erstellt und ausgeführt werden. Die dafür erforderlichen Mechanismen werden im nachfolgenden Abschnitt beschrieben. Eine Prozessinstanz kann als Laufzeitdaten für eine umgesetzte Workflowprozessdefinition angesehen werden.

2.2.1 Grundlegendes Verhalten der Workflow-Engine

Das Cúram-Workflow-Management-System umfasst eine Workflow-Engine, die die Laufzeitausführungsumgebung für eine Prozessinstanz bereitstellt. Es sind verschiedene Mechanismen zum Umsetzen eines

Workflowprozesses verfügbar. Diese werden in 19.2.1, „Prozessumsetzung", auf Seite 127 erläutert. Wenn ein Prozess umgesetzt wird, untersucht die Workflow-Engine die entsprechende Datenbanktabelle und erstellt mithilfe der *neuesten freigegebenen Version* der angegebenen Prozessdefinition die auszuführende Prozessinstanz.

Bei der Ausführung der einzelnen Aktivitäten wird jeweils ein zugehöriger Datensatz für die Aktivitätsinstanz von der Workflow-Engine erstellt und verwaltet. Dieser Datensatz enthält die Laufzeitdaten für eine Aktivitätsinstanz im umgesetzten Workflow. Im weiteren Workflowverlauf bewertet die Engine die Übergänge (siehe Kapitel 15, „Übergänge", auf Seite 109) für die verschiedenen Aktivitäten, um zu entscheiden, welcher Weg durch den Prozess genommen werden soll. Dazu zählt die Bestimmung der Typen von Splits und Joins (siehe Kapitel 17, „Splits/Joins (Verzweigungs-/Synchronisationspunkte)", auf Seite 119), über die die Aktivität verfügt, sowie das Ausführen aller Bedingungen (siehe Kapitel 16, „Bedingungen", auf Seite 113), die für die verschiedenen Übergänge im Prozess gelten. Übergangsinstanzdatensätze (die die Laufzeitdaten für einen Workflowübergang enthalten) für alle Übergänge im Workflowprozess werden ebenfalls von der Engine erstellt und verwaltet.

2.2.2 Ausführen mehrerer Versionen

Das Ändern und Freigeben einer neuen Version eines Prozesses hat keinerlei Auswirkungen auf aktuell ausgeführte Instanzen dieses Prozesses. Ein Prozess wird in der Workflow-Engine mit der Version, mit der er gestartet wurde, bis zum Ende ausgeführt, unabhängig aller möglicherweise freigegebenen Nachfolgeversionen.

2.2.3 Verwaltung von Prozessinstanzen

Ein Workflowadministrator kann die Ausführung einer aktiven Prozessinstanz über die Schnittstelle von Cúram Workflow Administration beeinflussen. Dafür stehen die folgenden Funktionen zur Verfügung:

Aussetzen einer Prozessinstanz
Jede gleichzeitig ausgeführte Prozessinstanz kann ausgesetzt werden. Wenn dieser Fall eintritt, sorgt die Workflow-Engine dafür, dass alle Aktivitätsinstanzen mit dem Status *In Bearbeitung* innerhalb dieser Prozessinstanz beendet werden können. Die nächste Gruppe von Aktivitäten, die für diese Prozessinstanz ausgeführt werden muss, wird dann zwar von der Workflow-Engine gestartet, jedoch sofort wieder ausgesetzt. Alle mit der ausgesetzten Prozessinstanz verknüpften synchronen Subflowprozesse (siehe Kapitel 11, „Subflow", auf Seite 87) werden auch von der Workflow-Engine ausgesetzt.

Fortsetzen einer Prozessinstanz
Jede Workflowprozessinstanz, die ausgesetzt wurde, kann fortgesetzt werden. Dabei werden die Aktivitätsinstanzen, die zuvor für die entsprechende Prozessinstanz ausgesetzt wurden, von der Workflow-Engine neu gestartet. Zudem werden auch alle ausgesetzten synchronen Subflowprozesse (siehe Kapitel 11, „Subflow", auf Seite 87), die dieser Prozessinstanz zugeordnet sind, von der Workflow-Engine fortgesetzt.

Abbrechen einer Prozessinstanz
Jede gleichzeitig ausgeführte oder ausgesetzte Prozessinstanz kann abgebrochen werden. Alle Aktivitäten mit dem Status *In Bearbeitung* in einer abgebrochenen Prozessinstanz werden beendet. Wenn der Prozess manuelle Aktivitäten oder Entscheidungsaktivitäten mit dem Status *In Bearbeitung* aufweist, werden beim Abbrechen der Prozessinstanz die zugehörigen Aufgaben von der Workflow-Engine geschlossen. Es werden keine einer abgebrochenen Prozessinstanz zugeordneten neuen Aktivitäten von der Workflow-Engine gestartet. Zudem werden alle synchronen Subflowprozesse (siehe Kapitel 11, „Subflow", auf Seite 87) abgebrochen, die der Prozessinstanz zugeordnet sind. Eine abgebrochene Prozessinstanz kann nicht fortgesetzt werden.

2.3 Bibliothek für Methodenverweise

Es gibt einige Situationen im Cúram-Workflow-ManagementSystem, bei denen eine Interaktion mit der Cúram-Anwendung durch Aufrufen verschiedener Methoden für Geschäftsprozesse und Entitäten erforderlich ist (siehe 7.3, „Cúram-Geschäftsmethoden", auf Seite 35 für ein Beispiel für eine solche Interaktion). Alle GPO- (GPO, Geschäftsprozessobjekt) oder Entitätsmethoden in der Anwendung können mithilfe der Workflow-Engine aufgerufen werden. Es gibt jedoch viel zu viele solcher Methoden, als dass sie einem Prozessdesigner in einer akzeptablen Art und Weise für die Verwendung in Prozessdefinitionen bereitgestellt werden könnten. Der Zweck dieser Bibliothek besteht darin, dass ein Administrator Methoden, die höchstwahrscheinlich in Prozessdefinitionen verwendet werden, zu einer Liste hinzufügt, die dann von Prozessdesignern leichter verwaltet und verwendet werden kann. Dabei ist es nicht erforderlich, im Vorfeld alle Methoden hinzuzufügen, die möglicherweise zukünftig verwendet werden. Bei Bedarf können der Bibliothek immer wieder neue Methoden hinzugefügt werden.

2.3.1 Referenzieren von Cúram-Methoden

GPO- (GPO, Geschäftsprozessobjekt) oder Entitätsmethoden müssen der Bibliothek für Methodenverweise hinzugefügt werden, bevor sie in einer Prozessdefinition referenziert werden können. Der Methodentyp, der beim Hinzufügen der Methode zur Bibliothek definiert wird, bestimmt, wo in einer Prozessdefinition die Methode zur Verfügung gestellt wird.

Beachten Sie, dass durch das Entfernen eines Methodenverweises aus der Bibliothek für Methodenverweise dieser nicht aus den Prozessdefinitionen gelöscht wird, die ihn referenzieren. Solange es sich bei einer Methode um eine gültige Cúram-Anwendungsmethode handelt, sind auch alle sie referenzierenden Prozessdefinitionen gültig.

2.3.2 Methodentypen

Der Bibliothek für Methodenverweise muss mit einer der drei definierten Methodentypen eine GPO- oder eine Entitätsmethode hinzugefügt werden. Eine Methode kann mehr als einem Methodentyp zugeordnet werden; dabei muss sie jedoch jedes Mal mit einem anderen Methodentyp erneut hinzugefügt werden. Nachfolgend sind die verschiedenen Methodentypen in der Methodenverweisbibliothek gemeinsam mit den Einschränkungen für ihre Verwendung aufgeführt.

Allgemein
Methoden mit dem Typ *Allgemein* sind nur als Anwendungsmethoden verfügbar, die aus automatischen Aktivitäten aufgerufen werden können (siehe 7.3, „Cúram-Geschäftsmethoden", auf Seite 35). Das Prozessdefinitionstool beschränkt den Zugriff auf Methoden ausschließlich dieses Typs, wenn eine Methode aus einer automatischen Aktivität abgerufen werden soll.

Zuteilung
Methoden in der Bibliothek mit dem Typ *Zuteilung* können nur als Zuteilungsstrategiefunktionen in Zusammenhang mit manuellen Aktivitäten, Entscheidungsaktivitäten, parallelen Aktivitäten und Aktivitätsbenachrichtigungen verwendet werden. (Siehe 9.4, „Zuteilungsstrategie", auf Seite 67.) Alle Methoden, die den Typ 'Zuteilung' aufweisen, müssen den Rückgabetyp `curam.util.workflow.struct.AllocationTargetList` haben.

Frist Methoden des Typs *Frist* können nur als Fristhandlermethoden für Event-Wait-Aktivitäten, manuelle Aktivitäten, Entscheidungsaktivitäten und parallele Aktivitäten referenziert werden. (Siehe 8.4, „Frist", auf Seite 52.)

2.4 WDO-Vorlagen

Daten werden in der Workflow-Engine als Workflowdatenobjekte (siehe Kapitel 4, „Workflowdatenobjekte", auf Seite 17) verwaltet und weitergeleitet. Es ist jedoch wahrscheinlich, dass einige der Workflowdatenobjekte in vielen Prozessdefinitionen zum Einsatz kommen. Daher wäre es günstig, wenn sie aus einem Pool importiert werden könnten anstatt in jedem einzelnen Prozess neu erstellt werden zu müssen. Dazu dient die Bibliothek mit den Vorlagen für Workflowdatenobjekte (WDO, Workflowdatenobjekte).

2.4.1 Metadaten

```
<wdo is-list-wdo="false" initialize-attributes="false">
    <wdo-name>TaskCreateDetails</wdo-name>
    <display-name>
        <localized-text>
            <locale language="en">TaskCreateDetailsName</locale>
        </localized-text>
    </display-name>
    <description>
        <localized-text>
            <locale language="en">The Task Create Details WDO
              Template</locale>
        </localized-text>
    </description>
    <attributes>
        <attribute>
            <attribute-name>subject</attribute-name>
            <display-name>
                <localized-text>
                    <locale language="en">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="en">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>
```

Die Metadaten, die für Vorlagen der Workflowdatenobjekte definiert sind, entsprechen den Metadaten, die für Workflowdatenobjekte definiert sind. Eine umfassende Beschreibung zu diesen Metadaten finden Sie in Kapitel 4, „Workflowdatenobjekte", auf Seite 17. Die Vorlagenbibliothek der Workflowdatenobjekte ist in der Datenbanktabelle 'WDOTemplateLibrary' gespeichert.

Beachten Sie, dass das Element `initialize-attributes` eines Workflowdatenobjekts und die Elemente `required-at-enactment`, `process-output` und `constant-value` eines Attributs des Workflowdatenobjekts in den Vorlagen der Workflowdatenobjekte nicht bearbeitet werden können und in den zugehörigen Metadaten automatisch mit ihren Standardwerten initialisiert werden.

2.4.2 Import und Synchronisation

Die in der Vorlagenbibliothek für Workflowdatenobjekte definierten Vorlagen stehen bei der Erstellung von Prozessdefinitionen zur Verfügung. Beim Importieren einer Vorlage eines Workflowdatenobjekts aus der Bibliothek werden das Workflowdatenobjekt und alle zugehörigen Attribute zu der aktuellen Prozessdefinition hinzugefügt.

Sobald eine Vorlage des Workflowdatenobjekts in eine Prozessdefinition importiert wurde, kann sie zu einem beliebigen Zeitpunkt mit dem entsprechenden Eintrag in der Vorlagenbibliothek für Workflowdatenobjekte synchronisiert werden. Durch das Synchronisieren der Vorlage für eine Prozessdefinition werden der Name und der Anzeigename des Workflowdatenobjekts von der Vorlagenbibliothek synchronisiert. Außerdem werden gleichzeitig alle neuen Attributeinträge, die im Vorlagenbibliothekseintrag vorhanden sind, automatisch zu dem Workflowdatenobjekt in der Prozessdefinition hinzugefügt. Der Benutzer kann bei der Synchronisation optional vorhandene Attribute im Workflowdatenobjekt mit denen aus der Vorlagenbibliothek überschreiben. Es muss beachtet werden, dass durch das Überschreiben vorhandener Attribute die Prozessdefinition ungültig gemacht werden kann und möglicherweise Aktualisierungen an den Stellen erforderlich werden, an denen alte Attributwerte verwendet werden.

2.4.3 Validierungen

- Ein Workflowdatenobjekt kann nicht aus einer Vorlage importiert werden, wenn in der zugehörigen Workflowprozessdefinition bereits ein Objekt mit demselben Namen vorhanden ist.

Kapitel 3. Metadaten der Prozessdefinition

3.1 Übersicht

Der Prozess ist das übergeordnete Konzept in einer Prozessdefinition. Es enthält primär Informationen zum Angeben und Beschreiben der Prozessdefinition. Zu diesen Informationen zählen die Kennung und die Version der Prozessdefinition sowie deren Name und eine kurze Beschreibung. Zudem enthält es eine Beschreibung der Fehlerzuteilungsstrategie, die für einen Prozess angegeben werden kann. In den folgenden Abschnitten werden diese allgemeinen Informationen beschrieben.

3.2 Metadaten

```
<workflow-process id="100" process-version="2"
                    language-version="1.0"
                    released="false" category="PC5"
                    createdBy="testuser"
                    creationDate="20050812T135800">
    <name>ApprovePlannedItem</name>
    <description>
      <localized-text>
        <locale language="en">This workflow process may be
        enacted to approve a planned item.</locale>
      </localized-text>
    </description>
    <documentation>Refer to the approve planned
                  item documentation.
    </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

Dies ist der übergeordnete Tag aller Prozessdefinitionsmetadaten.

id Hierbei handelt es sich um eine 64-Bit-Kennung, die vom Cúram-Schlüsselserver bereitgestellt wird, wenn ein Prozess im Prozessdefinitionstool erstellt wird. Die Prozesskennung muss im Cúram-Workflowsystem eindeutig sein. Der Grund dafür ist, dass die Workflow-Engine anhand der Prozesskennung in Verbindung mit der Prozessversionsnummer Prozessdefinitionseinträge für den Lesezugriff auf die Datenbank voneinander unterscheiden kann.

process-version

Diese Zahl steht für die Version einer Workflowprozessdefinition. Ein Datensatz zur Workflowprozessdefinition wird eindeutig durch seine Kennung und Versionsnummer gekennzeichnet. Eine Prozessdefinition kann mehrere freigegebene Versionen sowie eine Version zur Bearbeitung aufweisen. Sobald eine Prozessdefinition freigegeben wurde, wird eine neue Version erstellt und die Definition kann nicht mehr bearbeitet werden. Für alle nachfolgenden Aktualisierungen muss eine neue Version erstellt werden. Diese Versi-

© Copyright IBM Corp. 2012, 2013

on wird erst dann aktiv, wenn sie freigegeben wird. Wenn ein Prozess umgesetzt wird, wird die höchste freigegebene Versionsnummer verwendet. Prozessinstanzen mit einer festgelegten Versionsnummer behalten diese Nummer so lange bei, bis die Version fertiggestellt ist.

language-version
Bei den Prozessdefinitionsmetadaten handelt es sich um die Cúram-Workflowsprache. Beim Hinzufügen von neuen Funktionen und Erweiterungen kann sich diese Sprache ändern. Mithilfe der Versionsnummer kann die Workflow-Engine ältere Sprachversionen anders als neuere ausführen. Es können aber auch Upgrade-Tools verwendet werden, um alte Prozessdefinitionen in neue Sprachversionen zu konvertieren.

released
Dieses Attribut steht für ein boolesches Flag, das anzeigt, ob eine Prozessdefinition freigegeben wurde oder nicht. Nur Prozessdefinitionen, die freigegeben wurden, können umgesetzt oder als Unterprozesse in einer Subflowaktivität ausgewählt werden (siehe Kapitel 11, „Subflow", auf Seite 87).

category
Eine Prozessdefinition muss in eine Kategorie eingefügt werden. Die Kategorie muss im Prozessdefinitionstool ausgewählt werden und stammt aus der Codetabelle `ProcessCategory`. Dieses Attribut kann im Rahmen der Suchfunktion für die Prozessdefinition verwendet werden und hat keine funktionale Wirkung auf den Prozess in der Workflow-Engine.

createdBy
Dieses Attribut stellt den Namen des Benutzers dar, der die Workflowprozessdefinition erstellt hat. Es kann im Rahmen der Suchfunktion für die Prozessdefinition verwendet werden und hat keine funktionale Wirkung auf den Prozess in der Workflow-Engine.

creationDate
Dieses Attribut gibt das Datum und die Zeit der Erstellung der Workflowprozessdefinition an. Es kann im Rahmen der Suchfunktion für die Prozessdefinition verwendet werden und hat keine funktionale Wirkung auf den Prozess in der Workflow-Engine.

name Anhand des Namens der Prozessdefinition kann ein Prozess zur Umsetzung ermittelt werden. Der Umsetzungsservice (die API, über die ein Prozess in Code umgesetzt wird) bestimmt den umzusetzenden Prozess anhand seines Namens. Daher muss dieser Name innerhalb des Workflowsystems eindeutig sein und kann nach der Prozesserstellung nicht mehr geändert werden. Da es sich bei dem Prozessnamen effektiv um eine Konstante handelt, ist er im Gegensatz zu einem Aktivitätsnamen nicht lokalisierbar.

description
Ein Prozess kann außerdem eine kurze optionale Beschreibung aufweisen, die den Verwendungszweck angibt. Diese kann für die Benutzer hilfreich sein, die die Prozessdefinition zukünftig bearbeiten. Hierbei handelt es sich um ein lokalisierbares Textfeld im selben Format wie alle lokalisierbaren Felder in einer Prozessdefinition (siehe 6.2.1, „Lokalisierter Text", auf Seite 32).

documentation
Ein Prozess kann auch einen Link zu einer Dokumentation aufweisen, in der der Prozess noch näher beschrieben wird. Hierbei handelt es sich um ein Feld im Freitextformat, in dem der Entwickler den Namen eines für den Workflowprozess relevanten Dokuments eingeben oder einen Link zu einem solchen Dokument einfügen kann.

web-service
Dieses optionale Element beschreibt die Web-Service-Details eines Workflowprozesses. Ein Prozess kann durch Angabe dieses Metadatenwertes als Web-Service markiert werden. Der Wert zeigt an, dass der Prozess als Web-Service verfügbar gemacht werden sollte. Dadurch kann der Prozess als Teil eines mithilfe von BPEL (Business Process Execution Language) koordinierten

Prozess fungieren, was bedeutet, dass der Prozess über einen BPEL-Prozess aufgerufen werden kann. Einzelheiten zu dieser Funktion finden Sie in Kapitel 19, „Workflow-Web-Services", auf Seite 127.

expose
Dieses Attribut steht für ein boolesches Flag, das anzeigt, ob die Prozessdefinition als Web-Service verfügbar gemacht werden sollte oder nicht. Eine Workflowprozessdefinition wird standardmäßig nicht als Web-Service festgelegt.

callback-service
Dies ist ein optionales Element, da nicht für alle Aufrufe von einem BPEL-Prozess ein Rückruf erforderlich ist. Der Wert ist ein vollständig qualifizierter Name einer Klasse als Erweiterung der Klasse `org.apache.axis.client.Service` (die Teil des Service (der Axis-API) des Apache Axis-Projekts ist). Die Klasse `org.apache.axis.client.Service` wird von der Cúram-Web-Services-Connector-Funktion für ausgehende Web-Services generiert.

failure-allocation-strategy
Für einen Prozess kann auch eine optionale Fehlerzuteilungsstrategie angegeben sein. Beim Zuteilen einer Aufgabe (die einer manuellen Aktivität, siehe Kapitel 9, „Manuell", auf Seite 61, oder einer Entscheidungsaktivität, sieheKapitel 10, „Entscheidung", auf Seite 77, zugeordnet ist) ruft die Workflow-Engine die zugehörige Zuteilungsstrategie auf, um die Liste der Zuteilungsziele abzurufen. Wenn im Rahmen dieses Aufrufs keine Zuteilungsziele zurückgegeben werden, prüft die Workflow-Engine, ob eine Fehlerzuteilungsstrategie vorhanden ist und versucht dann mithilfe dieser Strategie, die Aufgabe zuzuteilen. Da die Zuteilungsstrategie des Typs *TARGET* direkt ein Zuteilungsziel angibt, ist es nie nötig, auf die Fehlerzuteilungsstrategie zurückzugreifen. Bei der Fehlerzuteilungsstrategie handelt es sich um eine prozessweite Strategie, die (wenn sie angegeben ist) bei Bedarf für alle manuellen Aktivitäten und Entscheidungsaktivitäten im Prozess verwendet wird.

allocation-strategy
Dieses Element beschreibt die für den Prozess verwendete Fehlerzuteilungsstrategie. Die Fehlerzuteilungsstrategie muss vom Typ *TARGET* sein. Wenn der geschäftliche Resolver die Aufgabe keinem Benutzer, keinem Organisationsobjekt (z. B. einer Organisationseinheit, einer Position oder einer Tätigkeit) oder keinem Gruppenpostfach zuweisen kann, die das angegebene Zuteilungsziel verwenden, wird die Aufgabe dem Standard-Gruppenpostfach zugewiesen. Das Kennungsattribut stellt die Kennung des Zuteilungsziels dar, das als Fehlerzuteilungsstrategie verwendet wird.

3.3 Validierungen

- Ein Workflowprozess muss einen eindeutigen Prozessnamen aufweisen. Dies bedeutet, dass ein Prozess nicht erstellt werden kann, wenn der Prozessname leer ist oder wenn bereits ein Prozess mit gleichem Namen vorhanden ist.
- Ein Workflowprozess muss eine Kategorie angeben.
- Eine freigegebene Version eines Workflowprozesses kann nach ihrer Umsetzung nicht gelöscht werden. Dies ist notwendig, da selbst bei Vorhandensein einer neueren Version eines Prozesses Prozessinstanzen, die noch verarbeitet werden, wenn die neue Version verfügbar wird, mit der Version beendet werden, mit der sie gestartet wurden. Prozessdefinitionen sind außerdem notwendige Verlaufsdatensätze, die zum Erstellen von Auditinformationen verwendet werden.
- Eine freigegebene Version eines Workflowprozesses kann nicht gelöscht werden, wenn sie von einer Subflowaktivität in einer freigegebenen Version eines anderen Prozesses referenziert wird, bei dem diese freigegebene Version die aktuelle freigegebene Version ist.
- Wenn für den Workflowprozess eine Fehlerzuteilungsstrategie angegeben wurde, muss sie den Typ *TARGET* aufweisen.
- Der Name der Rückrufserviceklasse kann nicht angegeben werden, wenn der Workflowprozess nicht als Web-Service festgelegt wurde.

- Der Name der Rückrufserviceklasse muss eine Klasse widerspiegeln, die im Klassenpfad der Anwendung vorhanden ist.
- Der Name der Rückrufserviceklasse muss eine Klasse darstellen, die die Klasse `org.apache.axis.client.Service` erweitert.

3.4 Beschreibung von Workflowdatenobjekten des Typs 'Context'

Während der Laufzeit einer Prozessinstanz müssen für die Aktivitäten und Übergänge einige bestimmte allgemeine Systemlaufzeitinformationen zur Workflow-Engine zur Verfügung gestellt werden. Details zum Workflowdatenobjekt `Context_RuntimeInformation`, das diese Informationen bereitstellt, finden Sie in 4.4, „Liste der Workflowdatenobjekte des Typs 'Context'", auf Seite 21.

Kapitel 4. Workflowdatenobjekte

4.1 Übersicht

Die Daten werden in der Workflow-Engine als Workflowdatenobjekte und Listen-Workflowdatenobjekte gepflegt und weitergeleitet. Es handelt sich hierbei um logische Objekte, die in der Prozessdefinition angegeben sind. Sie haben einen Namen und verfügen über eine Liste von Attributen verschiedener Typen, denen Daten zugewiesen werden können. Vom ihrem Konzept her ähneln sie Objekten in Programmiersprachen, obwohl ihre Darstellung im Workflowsystem natürlich eine ganz andere ist. Werte der Workflowdatenobjekte können bei der Prozessumsetzung oder basierend auf der Ausgabe unterschiedlicher Aktivitätstypen geschrieben werden.

Instanzen von Workflowdatenobjekten und Listen-Workflowdatenobjekten sind vorhanden, sobald der Prozess umgesetzt wird. Sie existieren dann so lange, bis der Prozess beendet ist. So können diese Objekte während der gesamten Lebensdauer der Prozessinstanz in den Aktivitäten und Übergängen des Prozesses verwendet werden. Der Prozessdesigner muss daher sicherstellen, dass die Attribute der Workflowdatenobjekte vor ihrer Verwendung mit Daten aufgefüllt werden. Weisen die Attribute vor ihrer Verwendung keine Daten auf, führt dies zur Laufzeit zu Fehlern.

© Copyright IBM Corp. 2012, 2013

4.2 Metadaten

```
<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="en">Task Create Details</locale>
          </localized-text>
        </display-name>
        <description>
          <localized-text>
            <locale language="en">This workflow data object
            contains the attributes required for the
            manual creation of a task.</locale>
          </localized-text>
        </description>
        <attributes>
          <attribute>
            <attribute-name>subject</attribute-name>
            <display-name>
              <localized-text>
                <locale language="en">Task subject</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="en">Participant Role ID</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="en">Deadline date</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="en">Deadline Duration</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>
```

wdos Dies ist optional (da eine Workflowprozessdefinition keine Workflowdatenobjekte enthalten muss) und enthält Informationen zu allen Workflowdatenobjekten, die für die Workflowprozessdefinition angegeben wurden.

wdo Enthält die Details eines der Workflowdatenobjekte, das für die Workflowprozessdefinition angegeben wurde. Dazu zählen allgemeine Informationen zum eigentlichen Objekt sowie Details zu den einzelnen Attributen des Objekts. Die Metadaten, die ein Workflowdatenobjekt und seine zugehörigen Attribute beschreiben, sind nachfolgend aufgeführt:

is-list-wdo
Enthält einen booleschen Wert, der anzeigt, ob das angegebene Workflowdatenobjekt ein Listen-Workflowdatenobjekt ist oder nicht. Ist der Wert auf `true` festgelegt, fungiert das angegebene Workflowdatenobjekt als Liste und kann somit im Rahmen des Workflows Daten in Listenform bereitstellen.

initialize-attributes
Enthält einen booleschen Wert, der anzeigt, ob die mit dem Workflowdatenobjekt verknüpften Attribute bei der ersten Verwendung des Workflowdatenobjekts initialisiert werden sollen. Die verwendeten Standardwerte entsprechen den Werten, wie sie in einer Cúram-Struktur angegeben werden würden.

wdo-name
Enthält den Namen des Workflowdatenobjekts.

display-name
Enthält den Anzeigename des Workflowdatenobjekts. Dieser Name stellt eine Kurzbeschreibung des Workflowdatenobjekts dar und wird im gesamten Prozessdefinitionstool angezeigt. Es handelt sich um eine lokalisierbare Zeichenfolge, die keine Parameter enthält. Weitere Informationen zu lokalisierbarem Text und den zugehörigen Metadaten finden Sie in 6.2.1, „Lokalisierter Text", auf Seite 32.

description
Enthält eine detailliertere Beschreibung des Workflowdatenobjekts. Auch hier handelt es sich um eine lokalisierbare Zeichenfolge ohne Parameter. Weitere Informationen zu lokalisierbarem Text und den zugehörigen Metadaten finden Sie in 6.2.1, „Lokalisierter Text", auf Seite 32.

attributes
Enthält die Details aller Attribute, die mit dem Workflowdatenobjekt verknüpft sind.

attribute
Enthält die Details eines der Attribute, das mit dem Workflowdatenobjekt verknüpft ist. Aus den nachfolgend beschriebenen Metadaten setzt sich ein Workflowdatenobjekt-Attribut zusammen:

attribute-name
Enthält den Namen des Workflowdatenobjekt-Attributs.

display-name
Gibt den Anzeigename des Workflowdatenobjekt-Attributs an. Dieser Name stellt eine Kurzbeschreibung des Workflowdatenobjekt-Attributsdar. Es handelt sich um eine lokalisierbare Zeichenfolge, die keine Parameter enthält. Weitere Informationen zu lokalisierbarem Text und den zugehörigen Metadaten finden Sie in 6.2.1, „Lokalisierter Text", auf Seite 32.

type Jedes definierte Workflowdatenobjekt-Attribut muss einen Typ angeben, bei dem es sich um eine gültige Cúram-Basisdomäne handeln muss. Beim Erstellen eines Workflowdatenobjekt-Attributs im Prozessdefinitionstool wird dieser Typ aus der Codetabelle `DomainType` ausgewählt. Aus dieser Codetabelle sollte die Liste der Typen abgefragt werden, die für Workflowdatenobjekt-Attribute zur Verfügung stehen. Der Typ eines Workflowdatenobjekt-Attributs wird verwendet, um sicherzustellen, dass die in einem Workflowprozess enthaltenen Datenzuordnungen kompatibel sind und keine Fehler zur Laufzeit verursachen. Ein Beispiel dafür wäre, dass, wenn ein Parameterfeld für eine Geschäftsprozessob-

jektmethode den Typ 'STRING' aufweist, das Workflowdatenobjekt-Attribut, das zum Zuordnen der Daten in dieses Feld verwendet wird, auch vom Typ 'STRING' sein muss.

required-at-enactment
Umsetzungszuordnungen stellen die mindestens erforderliche Datenmenge dar, die für die Umsetzung des Workflows benötigt wird. Sie müssen einen Eintrag für jedes Workflowdatenobjekt-Attribut enthalten, das mit dem Wert `true` als erforderlich für die Umsetzung festgelegt ist. Wenn hingegen der Wert auf `false` (den Standardwert) gesetzt ist, bedeutet dies, dass das Workflowdatenobjekt-Attribut nicht für die Umsetzung des zugehörigen Prozesses erforderlich ist. Diese Umsetzungszuordnungen werden mithilfe des Prozessdefinitionstools erstellt. Dabei wird jedes definierte Workflowdatenobjekt-Attribut untersucht und eine Zuordnung für die Attribute erstellt, die mit dem Wert `true` als erforderlich für die Umsetzung festgelegt sind. Wenn eine freigegebene Workflowprozessdefinition als Subflowprozess in einer Subflowaktivität ausgewählt wurde (siehe Kapitel 11, „Subflow", auf Seite 87), müssen alle im Subflowprozess als erforderlich für die Umsetzung markierten Workflowdatenobjekte zugeordnet werden, bevor diese übergeordnete Prozessdefinition freigegeben werden kann.

process-output
Ein Workflowprozess kann durch Angabe eines Metadatenwertes als Web-Service markiert werden, der anzeigt, dass der Prozess als Web-Service verfügbar gemacht werden sollte. Dadurch kann der Prozess als Teil eines mithilfe von BPEL (Business Process Execution Language) koordinierten Prozess fungieren, was bedeutet, dass der Prozess über einen BPEL-Prozess entweder synchron oder asynchron aufgerufen werden kann. Es kann auch notwenig sein, die Daten aus einem Workflowprozess zurück in den BPEL-Prozess zu speisen, aus denen sie abgerufen wurden. Ist dieses optionale Element auf `true` gesetzt, gibt es an, dass die Daten bei Beendigung des Cúram-Workflowprozesses aus diesem Workflowdatenobjekt-Attribut zurück zum aufrufenden BPEL-Prozess geleitet werden sollten. Die Standardwert für dieses Element ist `false`.

constant-value
Dieses optionale Element gibt an, ob das Workflowdatenobjekt-Attribut einen konstanten Wert darstellt. An verschiedenen Stellen in einer Workflowprozessdefinition werden Workflowdatenobjekt-Attribute in Eingabezuordnungen verwendet (d. h. in Zuordnungen der Zuteilungsfunktion, in Zuordnungen der Fristfunktion usw.). Unter gewissen Umständen ist es erforderlich, in einigen dieser Zuordnungen Konstanten zu verwenden. Durch das Bereitstellen eines konstanten Wertes können Workflowdatenobjekt-Attribut dieses Typs für diesen Zweck verwendet werden. Ein Workflowdatenobjekt-Attribut kann nicht als für die Umsetzung erforderlich markiert (Flag `true`) sein und gleichzeitig einen konstanten Wert enthalten. Daten, die als Umsetzungsdaten übergeben werden, werden als dynamische Daten angesehen, die veränderbar sind. Die Daten, die in einem konstanten Workflowdatenobjekt-Attribut angegeben sind, sind für diesen Zweck nicht geeignet, da ihre Werte bereits bekannt sind.

initial-value
Dieses Element zeigt an, ob das Workflowdatenobjekt-Attribut einen Anfangswert aufweist. Dieses Funktion kann in Situationen hilfreich sein, in denen ein Workflowdatenobjekt-Attribut im Workflow verwendet wird, bevor es von einer automatischen Aktivität oder in ähnlicher Form mit Daten aufgefüllt wurde (d. h. um zu verhindern, dass eine automatische Aktivität zum Auffüllen von Workflowdatenobjekt-Attributen verwendet wird, um sicherzustellen, dass diese Attribute nicht null sind, wenn sie zu einem späteren Zeitpunkt im Workflow als ein Bestandteil von Übergangsbedingungen verwendet werden). Wenn dieses Element aufgefüllt wurde, wird das Workflowdatenobjekt-Attribut bei seiner erstmaligen Verwendung mit dem angegebenen Wert initialisiert. Der Anfangswert eines Workflowdatenobjekt-Attributs kann später durch verschiedene im Workflowprozess vorhandene Ausgabezuordnungen überschrieben werden. Für ein Workflowdatenobjekt-Attribut können nicht gleichzeitig ein konstanter Wert und ein Anfangswert angegeben sein.

4.3 Validierungen

- Ein Workflowprozess darf nur ein Workflowdatenobjekt des Typs `Context_RuntimeInformation` enthalten.
- Der Name eines Workflowdatenobjekts muss bezüglich der zugehörigen Workflowprozessdefinition eindeutig sein.
- Der Name eines Workflowdatenobjekts muss eine gültige Java™-Kennung sein.
- Ein benutzerdefinierte Name eines Workflowdatenobjekts darf nicht das Präfix `Context_` aufweisen, da es sich hierbei um ein reserviertes Präfix im Cúram-Workflowsystem handelt.
- Alle in der Workflowprozessdefinition angegebenen Workflowdatenobjekte müssen mindestens ein zugeordnetes Attribut aufweisen.
- Der Name des Workflowdatenobjekt-Attributs muss eine gültige Java-Kennung sein.
- Ein Attribut des Workflowdatenobjekts kann nicht mit dem Namen `"value"` erstellt werden. Hierbei handelt es sich um einen reservierten Attributnamen im Cúram-Workflowsystem.
- Der Typ eines Workflowdatenobjekt-Attributs muss eine gültige Cúram-Basisdomäne sein und in der Codetabelle `DomainType` enthalten sein.
- Ein Workflowdatenobjekt-Attribut kann nicht gleichzeitig als für die Umsetzung erforderlich und als konstanter Wert markiert sein.
- Für ein Workflowdatenobjekt-Attribut können nicht gleichzeitig ein konstanter Wert und ein Anfangswert angegeben sein.
- Wenn ein Workflowdatenobjekt-Attribut als Konstante markiert wurde, muss ein konstanter Wert angegeben werden. Im Gegenzug sollte für ein Attribut, dass nicht als Konstante markiert wurde, kein solcher Wert bereitgestellt werden.
- Wenn das Workflowdatenobjekt-Attribut als Konstante markiert wurde, kann ein leerer Wert für dieses Attribut nur dann angegeben werden, wenn es sich bei dem Attribut um eine Zeichenfolge handelt (Typ 'STRING').
- Wenn das Workflowdatenobjekt-Attribut mit einem Anfangswert angegeben wurde, kann ein leerer Anfangswert für dieses Attribut nur dann angegeben werden, wenn es sich bei dem Attribut um eine Zeichenfolge handelt (Typ 'STRING').
- Wenn das Workflowdatenobjekt-Attribut als Konstante markiert wurde, muss der als diese Konstante angegebene Wert mit dem Typ des zugehörigen Attributs kompatibel sein.
- Wenn das Workflowdatenobjekt-Attribut mit einem Anfangswert angegeben wurde, muss der als dieser Anfangswert angegebene Wert mit dem Typ des zugehörigen Attributs kompatibel sein.
- Das Prozessausgabe-Flag kann für ein angegebenes Workflowdatenobjekt-Attribut nur dann auf den Wert "true" gesetzt werden, wenn der zugehörige Workflowprozess als Web-Service verfügbar gemacht wurde.

4.4 Liste der Workflowdatenobjekte des Typs 'Context'

Workflowdatenobjekte des Typs 'Context' sind Objekte, die nicht explizit in den Metadaten der Workflowprozessdefinition definiert sind, sondern die mithilfe des Prozessdefinitionstools und der Workflow-Engine an verschiedenen Stellen während der Ausführung eines Prozesses bereitgestellt werden. Nachfolgend finden Sie eine kurze Beschreibung dieser Workflowdatenobjekte sowie Verknüpfungen zu weiterführenden Informationen zu den Objekten.

Workflowdatenobjekt 'Context_RuntimeInformation'

Das Workflowdatenobjekt 'Context_RuntimeInformation' ist ein Objekt, das von der Workflow-Engine bereitgestellt und verwaltet wird. Es enthält Informationen, die während des Lebenszyklus einer Workflowprozessinstanz relevant sind, sowie die entsprechenden Attribute. Diese Attribute sind folgende:

- `processInstanceID`: Die vom System generierte Kennung der Prozessinstanz (die mithilfe des Workflowschlüsselsatzes vom Cúram-Schlüsselserver abgerufen wird).

- `enactingUser`: Der Benutzername des Benutzers, dessen Aktionen in der Anwendung zur Umsetzung des Workflowprozesses geführt haben.
- `enactmentTime`: Der Zeitpunkt (Datum und Uhrzeit), zu dem der Prozess umgesetzt wurde.

Workflowdatenobjekt 'Context_Result'
Ein Übergang von einer automatischen Aktivität sollte den Rückgabewert der aufgerufenen Methode direkt in seiner Bedingung verwenden können, ohne dabei auf Zuordnungen zu Workflowdatenobjekt-Attributen zurückgreifen zu müssen. Aufgrund des transaktionsorientierten Modells der Workflow-Engine müssen diese Daten jedoch außerhalb der Transaktion des Aufrufs der GPO-Methode beibehalten werden. Um dies zu erreichen, wird zur Laufzeit eine Workflowdatenobjektdefinition erstellt, wenn der Rückgabewert in Bedingungen für einen ausgehenden Übergang verwendet wird. Diese Rückgabewertdefinitionen müssen nicht beibehalten werden, da sie jederzeit bei Bedarf in der Workflow-Engine abgeleitet werden können. Die tatsächlichen Workflowdatenobjektdaten werden beibehalten und erst dann gelöscht, wenn die Übergängen von den fraglichen Aktivitätsinstanzen bewertet wurden. Weitere Informationen zum Workflowdatenobjekt 'Context_Result' finden Sie in 7.6, „Beschreibung der Workflowdatenobjekte des Typs 'Context'", auf Seite 46.

Workflowdatenobjekt 'Context_Event'
Das Workflowdatenobjekt 'Context_Event' kann in Datenelement- oder Funktionsbedingungen (siehe Kapitel 16, „Bedingungen", auf Seite 113) für einen Übergang von einer Aktivität mit einem Event-Wait verwendet werden. Es stellt gewisse Informationen (z. B. die Ereignisklasse und den Ereignistyp des ausgelösten Ereignisses, den Zeitpunkt, zu dem das Ereignis ausgelöst wurde usw.) bereit, die im ausgelösten Ereignis enthalten sind, um die Aktivitätsinstanz erfolgreich zu beenden. Diese Informationen können dann zum Modellieren des Weges für die angegebene Aktivität verwendet werden. Weitere Informationen zum Workflowdatenobjekt 'Context_Event' finden Sie in 8.5.4, „Beschreibung von Workflowdatenobjekten des Typs 'Context'", auf Seite 57.

Workflowdatenobjekt 'Context_Decision'
Das Workflowdatenobjekt 'Context_Decision' kann in Datenelement- oder Funktionsbedingungen (siehe Kapitel 16, „Bedingungen", auf Seite 113) für einen Übergang von einer Entscheidungsaktivität verwendet werden. Die verfügbaren Attribute hängen von dem für die Entscheidungsaktivität definierten Antwortformat ab. Weitere Informationen zum Workflowdatenobjekt 'Context_Decision' finden Sie in 10.4.4, „Beschreibung von Workflowdatenobjekten des Typs 'Context'", auf Seite 85.

Workflowdatenobjekt 'Context_Task'
Das Workflowdatenobjekt 'Context_Task' kann in verschiedenen Zuordnungen (z. B. Eingabezuordnungen der Zuteilungsfunktion, Eingabezuordnungen der Fristfunktion, Aktionslinkparameter einer manuellen Aktivität) verwendet werden, die einer Aufgabe für eine manuelle Aktivität zugewiesen sind. Dieses Workflowdatenobjekt stellt die Kennung der Aufgabe zur Verfügung, die als Ergebnis der Ausführung der zugehörigen Aktivität erstellt wurde. Weitere Informationen zum Workflowdatenobjekt 'Context_Task' finden Sie in 9.3.5, „Beschreibung von Workflowdatenobjekten des Typs 'Context'", auf Seite 67.

Workflowdatenobjekt 'Context_Loop'
Das Workflowdatenobjekt 'Context_Loop' kann beim Erstellen der mit einer Schleifenbeginnaktivität verknüpften Schleifenbedingung verwendet werden. Außerdem kann es für jede beliebige Aktivität in einer Schleife zum Erstellen von Bedingungen für ausgehende Übergänge sowie bei der Angabe von Eingabezuordnungen zum Definieren von Textparametern und Aktionslinkparametern für einige Aktivitäten und Funktionen in einer Schleife verwendet werden. Dieses Workflowdatenobjekt gibt für solche Zuordnungen die Häufigkeit an, mit der eine Schleife durchlaufen wurde. Weitere Informationen zum Workflowdatenobjekt 'Context_Loop' finden Sie in 12.5, „Beschreibung von Workflowdatenobjekten des Typs 'Context'", auf Seite 93.

Workflowdatenobjekt 'Context_Deadline'
Das Workflowdatenobjekt 'Context_Deadline' kann beim Erstellen von Datenelement- oder Funktionsbedingungen (siehe Kapitel 16, „Bedingungen", auf Seite 113) für einen Übergang von einer Aktivität mit einem fristbasierten Event-Wait verwendet werden. Mithilfe dieses Objekts kann ein

Entwickler verschiedene Ausführungspfade aus einer Aktivität heraus modellieren, die eine Frist enthält. Diese Pfade hängen davon ab, wann die Frist abgelaufen ist. Weitere Informationen zum Workflowdatenobjekt 'Context_Deadline' finden Sie in 8.4.6, „Beschreibung von Workflowdatenobjekten des Typs 'Context'", auf Seite 55.

Workflowdatenobjekt 'Context_Parallel'
Das Workflowdatenobjekt 'Context_Parallel' kann in verschiedenen Zuordnungen verwendet werden, die einer manuellen parallelen Aktivität (z. B. Textparameter für den Aufgabenbetreff und für die Aufgabenaktion, Zuordnungen für die Zuteilungsstrategie usw.) und einer parallelen Entscheidungsaktivität (z. B. Textparameter für die Entscheidungsaktion, für die sekundäre Aktion, für die Frage usw.) zugeordnet sind. Das Objekt stellt den Index des Elements aus dem Listen-Workflowdatenobjekt der parallelen Aktivität bereit, das zum Erstellen der angegebenen Instanz der umschlossenen Aktivität verwendet wird. Weitere Informationen zum Workflowdatenobjekt 'Context_Parallel' finden Sie in 13.3.6, „Beschreibung von Workflowdatenobjekten des Typs 'Context'", auf Seite 99.

Workflowdatenobjekt 'Context_Error'
Das Workflowdatenobjekt 'Context_Error' kann in Datenelement- oder Funktionsbedingungen (siehe Kapitel 16, „Bedingungen", auf Seite 113) für einen Übergang von einer automatischen Aktivität verwendet werden. Mithilfe dieses Objekts kann ein Prozessentwickler einen Ausführungspfad aus einer automatischen Aktivität heraus modellieren, d. h. einen Übergang, der verfolgt wird, wenn die automatische Aktivität aufgrund einer nicht verarbeiteten Ausnahme fehlschlägt. Weitere Informationen zum Workflowdatenobjekt 'Context_Error' finden Sie in 7.6, „Beschreibung der Workflowdatenobjekte des Typs 'Context'", auf Seite 46.

4.5 Laufzeitinformationen

Instanzen von Workflowdatenobjekten und Listen-Workflowdatenobjekten sind vorhanden, sobald ein Workflowprozess umgesetzt wird. Sie existieren dann so lange, bis der Prozess beendet ist. Diese Workflowdatenobjektinstanzen können so in den Aktivitäten (z. B. zum Weiterleiten von Daten an die GPO-Methode) und Übergängen (z. B. zum Bereitstellen von Daten für die Bewertung von Übergangsbedingungen) während der gesamten Lebensdauer dieser Prozessinstanz verwendet werden.

Das Attribut `enactingUser` des Workflowdatenobjekts `Context_RuntimeInformation` gibt den Benutzernamen des Benutzers an, dessen Aktionen in der Anwendung zur Umsetzung des Workflowprozesses geführt haben. Es führt *nicht* dazu, dass derselbe Wert der Transaktion zugeordnet wird, wenn eine GPO-Methode nachfolgend in der Workflowprozessinstanz aufgerufen wird. Der Grund dafür ist die Demarkation der Transaktion in der Workflow-Engine, wenn automatische Aktivitäten (z. B. GPO-Methoden) im Anwendungsserver aufgerufen werden. Da dieser Aufruf asynchron erfolgt und sichergestellt werden muss, dass der Aufruf an den Anwendungscode innerhalb der eigenen Transaktion erfolgt, wird die GPO-Methode von der Workflow-Engine (Benutzer 'SYSTEM') aufgerufen und nicht von dem Benutzer, der den Workflowprozess ursprünglich umgesetzt hat. Tatsächlich weiß die Person, die den Workflow umgesetzt hat, wahrscheinlich nicht einmal, dass sie diese GPO-Methode aufgerufen hat.

In ähnlicher Weise sollte beachtet werden, dass der ausführende Benutzer einer Workflowprozessinstanz nicht in eine der Subflowprozessinstanzen übertragen wird, die eventuell vom übergeordneten Prozess aufgerufen wurden. Wenn der ausführende Benutzer der übergeordneten Prozessinstanz in einer der Subflowprozessinstanzen benötigt wird, sollte er mithilfe eines Workflowdatenobjekt-Attributs in der Eingabezuordnung für diesen Subflowprozess explizit übergegeben werden.

Bei der Aktualisierung von Daten für eine Workflowdatenobjekt-Attributinstanz während der Ausführung paralleler automatischer Aktivitäten in einer Workflowprozessinstanz ist Vorsicht geboten. Wenn solche automatischen Aktivitäten dieselbe GPO-Methode aufrufen und diese Methode versucht, die Daten für das exakt gleiche Workflowdatenobjekt-Attribut zu aktualisieren, kann dies zu einer Sperre von Datenbanksätzen führen. Der Workflowprozessdesigner sollte das Auftreten einer solchen Situation vermeiden, indem er die Workflowprozessdefinition so gestaltet, dass sichergestellt wird, dass parallel ausgeführte automatische Aktivitäten nicht dasselbe Workflowdatenobjekt-Attribut aktualisieren.

Kapitel 5. Prozessumsetzung

5.1 Übersicht

Eine Prozessdefinition definiert die Struktur eines Geschäftsprozesses. Um die Verarbeitung dieser Prozessdefinition zu starten, muss eine Instanz des Prozesses erstellt werden. Das Starten einer Prozessinstanz wird als *Prozessumsetzung* bezeichnet. Für die meisten Prozessdefinitionen ist ein Mindestdatensatz erforderlich, der primär zum Angeben bestimmter Geschäftsobjekte für die Prozessinstanz verwendet wird. Alle Umsetzungsmechanismen müssen die Eingabedaten in irgendeiner Form akzeptieren können, um einen bestimmten Prozess starten zu können. Diese Eingabedaten werden als *Umsetzungsdaten* für einen Prozess bezeichnet.

Derzeitig werden vier Umsetzungsmechanismen von Cúram Workflow unterstützt:

- Umsetzung aus Code
- Umsetzung aus einem Ereignis
- Umsetzung als Subflow
- Umsetzung über einen Web-Service

Die beiden ersten Mechanismen werden in diesem Kapitel beschrieben. Informationen zum Subflow-Umsetzungsmechanismus finden Sie in Kapitel 11, „Subflow", auf Seite 87. Eine Beschreibung des Web-Service-Umsetzungsmechanismus enthält Kapitel 19, „Workflow-Web-Services", auf Seite 127.

5.2 Codeumsetzung (Umsetzungsservice-API)

Der direkte Weg zum Umsetzen eines Prozesses besteht darin, eine Stelle in der Anwendung zu ermitteln, an der eine Prozessinstanz gestartet werden muss. Anschließend muss an der entsprechenden Stelle Code eingefügt werden, um die Umsetzungsservice-API aufzurufen. Diese Anwendungsprogrammierschnittstelle (API) ermöglicht Entwicklern, den Namen des zu startenden Prozesses festzulegen und die vom Prozess benötigten Daten bereitzustellen.

Die Umsetzung eines Prozesses auf diese Weise ist zwar einfach und intuitiv, hat aber den Nachteil, dass sie in der Anwendungslogik fest programmiert ist. Daher bewirken Änderungen wie das Entfernen der Umsetzung, das Umgestalten des Prozessstarts oder auch nur geringfügige Anpassungen an den erforderlichen Umsetzungsdaten, dass der Code bearbeitet und die der Anwendung erneut bereitgestellt werden muss.

© Copyright IBM Corp. 2012, 2013

5.2.1 Metadaten

```
<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
Enthält eine Liste von Zuordnungen, die als Anfangsdaten für die Umsetzung der zugehörigen Prozessinstanz verwendet werden können. Für eine Prozessdefinition müssen keine Umsetzungszuordnungen definiert sein, damit sie umgesetzt werden kann.

mapping
Der Tag `mapping` stellt ein von einem Cúram-Strukturattribut bereitgestelltes Datenelement dar, das für die Umsetzung der zugehörigen Prozessinstanz verwendet werden soll.

source-attribute
Stellt ein Cúram-Strukturattribut dar, dass zum Auffüllen der Umsetzungsdaten für den Prozess verwendet werden soll und das in einer Umsetzungszuordnung erforderlich ist.

struct-name
Der Name einer Cúram-Struktur, die ein für die Umsetzung des Workflowprozesses benötigtes Attribut enthält. Aggregierte Strukturen und Listenstrukturen können zudem zum Übergeben von Umsetzungsdaten an einen Workflowprozess verwendet werden, wie im Metadatenausschnitt oben dargestellt.

name Der Name des Attributs einer Cúram-Struktur, das für die Umsetzung des zugehörigen Workflowprozesses erforderlich ist. Wenn ein Feld aus einer aggregierten Struktur oder eine Listenstruktur verwendet wird, stellt dieser Name den vollständig qualifizierten Namen dieses Feldes dar. In einem solchen Fall besteht der Name neben dem eigentlichen Feldnamen aus dem Rollennamen aus der Zuordnung zwischen der übergeordneten und der untergeordneten Struktur. Dies ist im Metadatenabschnitt oben dargestellt.

target-attribute
Dieser Tag stellt ein Workflowdatenobjekt-Attribut dar, das mit den Umsetzungsdaten für den Prozess aufgefüllt werden soll und das in einer Umsetzungszuordnung erforderlich ist.

wdo-name
Der Name eines Cúram-Workflowdatenobjekts, das das zuzuordnende Zielattribut enthält. (Siehe Kapitel 4, „Workflowdatenobjekte", auf Seite 17.)

name Der Name eines Cúram-Workflowdatenobjekt-Attributs, das als erforderlich für die Umsetzung markiert ist. Der Wert des entsprechenden Quellenattributs der Cúram-Struktur wird diesem Attribut bei der Umsetzung des Prozesses zugeordnet.

5.2.2 Validierungen

- Das in einer Umsetzungszuordnung als Quellenattribut verwendete Cúram-Strukturattribut muss ein gültiges Attribut sein und den richtigen Typ für das zugehörige Zielattribut des Workflowdatenobjekts aufweisen.
- Das Zielattribut des Workflowdatenobjekts in einer Umsetzungszuordnung muss ein gültiges Attribut und als erforderlich für die Umsetzung markiert sein.
- Wenn das Zielattribut der Umsetzungszuordnung aus einem Listen-Workflowdatenobjekt stammt, muss das Quellenattribut ein Feld aus einer Listenstruktur sein.

5.2.3 Code

```
// Create the list we will pass to the enactment service.
    final List enactmentStructs = new ArrayList();

    final TaskCreateDetails taskCreateDetails =
      new TaskCreateDetails();

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

    enactmentStructs.add(taskCreateDetailsStruct);

    // An aggregated struct.
    GroupMemberDetails groupMemberDetails
      = new GroupMemberDetails();

    groupMemberDetails.dtls.memberName = "Test User";

    enactmentStructs.add(groupMemberDetails);

    // A list struct.
    ChildDetailsList childDetailsList
      = new ChildDetailsList();

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

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

    enactmentStructs.add(childDetailsList);


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

- Die API `EnactmentService` wird bereitgestellt, um die Umsetzung des Workflowprozesses über den Anwendungscode zu ermöglichen. Die Liste der für die Methode `startProcess()` bereitgestellten Cúram-Strukturen muss ausreichen, um die Umsetzungszuordnungen des zugehörigen Prozesses vollständig mit Daten aufzufüllen. Beachten Sie, dass die Umsetzung eines Prozesses auf diese Weise asynchron ist und der Prozess gestartet wird, sobald die aktuelle Anwendungstransaktion abgeschlossen ist.

- Die Methode `startProcessInV3CompatibilityMode` wird nur für die Verwendung der Basisanwendungsaufgaben-API bereitgestellt. Die direkte Verwendung dieser Methode in angepasstem Code wird nicht unterstützt und kann zukünftige Aktualisierungen beeinträchtigen.

5.3 Ereignisumsetzung

Es ist möglich, einen Prozess als Reaktion auf ein ausgelöstes Ereignis zu starten. Dazu müssen bestimmte Konfigurationsdaten festgelegt werden (entweder über eine Administrationsschnittstelle oder als vorkonfigurierte Datenbankeinträge). Durch die Konfiguration wird festgelegt, welche Prozesse als Reaktion auf ein bestimmtes ausgelöstes Ereignis gestartet werden sollen. Die Zuordnung der Ereignisdaten zu den vom Prozess benötigten Umsetzungsdaten kann ebenfalls auf diese Weise konfiguriert werden.

Die Ereigniskonfiguration für die Prozessumsetzung wird in der Datenbank gespeichert und es wird eine Benutzeroberfläche bereitgestellt, über die diese Daten bearbeitet werden können. Eine auf diese Weise erstellte Prozessumsetzung kann zur Laufzeit aktiviert, inaktiviert, geändert und sogar entfernt werden. Der Nachteil dieses Ansatzes besteht jedoch darin, dass nur Prozessdefinitionen, die eine entsprechend kleine Menge an Umsetzungsdaten benötigen, auf diese Weise umgesetzt werden können, da Ereignisse nur über eine begrenzte Datenmenge verfügen.

Es wird ein Ereignishandler für die Prozessumsetzung von Cúram bereitgestellt, der automatisch registriert wird, um in der Anwendung ausgelöste Ereignisse zu überwachen. Wenn ein Prozess für die Umsetzung von einem Ereignis konfiguriert wurde, werden die Daten aus dem Ereignis den Umsetzungsdaten des Prozesses zugeordnet und der Prozess wird gestartet.

5.3.1 Konfigurationsdaten

Für die Aktivierung eines Ereignisses zur Umsetzung eines Prozesses ist das Konfigurieren einer Ereignis/Prozess-Zuordnung erforderlich. Jedes Ereignis, das in der Anwendung ausgelöst wird, prüft, ob irgendwelche Prozesse zugeordnet wurden und umgesetzt werden müssen. Es wird immer die neueste freigegebene Version eines Prozesses für ein zugehöriges Ereignis umgesetzt.

Die Registrierung eines Ereignisses zum Auslösen eines Prozesses wird als Datensatz in der Tabelle 'ProcEnactmentEvt' gespeichert. Der Ereignishandler für die Prozessumsetzung durchsucht die zwischengespeicherte Abbildung dieser Tabelle nach übereinstimmenden Einträgen, wenn ein Ereignis in der Anwendung ausgelöst wird, und setzt alle entsprechenden Prozesse um. In der folgenden Tabelle sind die Daten beschrieben, die zum Auffüllen der Tabelle 'ProcEnactmentEvt' erforderlich sind.

Tabelle 1. Beschreibung der Tabelle 'ProcEnactmentEvt'

Entitätsfeldname	**Feldbeschreibung**
procStartEventID	Die eindeutige Kennung der Ereignis/Prozess-Zuordnung.
eventClass	Die Ereignisklasse des Ereignisses, die für die Umsetzung des Workflowprozesses angegeben wurde.
eventType	Der Ereignistyp des Ereignisses, der für die Umsetzung des Workflowprozesses angegeben wurde.
processToStart	Wenn ein Ereignis mit der oben beschriebenen Ereignisklasse und dem Typ ausgelöst wird, wird die neueste freigegebene Version des Prozesses umgesetzt, die durch diesen Namen angegeben wird.

Tabelle 1. Beschreibung der Tabelle 'ProcEnactmentEvt' (Forts.)

Entitätsfeldname	Feldbeschreibung
enabled	Dieses boolesche Flag zeigt an, ob die Ereignis/Prozess-Zuordnung aktiviert ist. Dadurch kann die Umsetzung eines Workflowprozesses durch ein angegebenes Ereignis zur Laufzeit aktiviert/inaktiviert werden.

In der Tabelle 'ProcEnactEvtData' werden die Daten gespeichert, die einem umzusetzenden Workflowprozess aus einem Geschäftsereignis zugeordnet werden, wenn dieses angegebene Ereignis ausgelöst wird. In der folgenden Tabelle sind die Daten beschrieben, die zum Auffüllen der Tabelle 'ProcEnactEvtData' erforderlich sind.

Tabelle 2. Beschreibung der Tabelle 'ProcEnactEvtData'

Entitätsfeldname	Feldbeschreibung
procEventMappingID	Die eindeutige Kennung der Zuordnung zwischen Prozessumsetzungs- und Ereignisdaten.
procStartEventID	Die eindeutige Kennung der Ereignis/Prozess-Zuordnung. Dieses Feld ist der eindeutige Schlüssel in der zugehörigen Tabelle 'ProcEnactmentEvt' und wird verwendet, um alle Daten zuzuordnen, die zur Umsetzung des Workflowprozesses beim Auslösen eines angegebenen Ereignisses benötigt werden.
eventField	Dieses Element zeigt an, welches der drei Felder eines Ereignisses zum Auffüllen des Attributs des Workflowdatenobjekts verwendet wird. Die Werte für dieses Feld stammen aus der Codetabelle `EventField` und sind nachfolgend näher beschrieben.
wdoAttribute	Der vollständig qualifizierte Name eines Workflowdatenobjekt-Attributs, das bei der Umsetzung eines Prozesses mit Daten aus einem bestimmten Feld aufgefüllt wird. Diese Tabelle enthält einen Eintrag für jedes Workflowdatenobjekt-Attribut, das in dem Prozess, der vom ausgelösten Ereignis umgesetzt wird, als erforderlich für die Umsetzung markiert wurde.

Es gibt drei Felder eines Ereignisses, die als Umsetzungszuordnungen verwendet werden können. Diese sind in der Codetabelle `EventField` aufgeführt und nachfolgend beschrieben.

primary event data
: Eine eindeutige Kennung, die sich auf die Ereignisklasse bezieht, von der aus das Ereignis ausgelöst wird. Wenn beispielsweise der für ein Ereignis angegebene Geschäftsobjekttyp dem Typ 'Case' (Fall) entspricht, kann es sich bei den Ereignisdaten um die Fallkennung handeln.

secondary event data
: Dies kann ein beliebiger numerischer Wert sein, der sich auf Ereignisse bezieht, die eine Zuordnung zwischen zwei Einträge darstellen müssen.

raised by user
: Der Cúram-Benutzername des Benutzers, der das Ereignis ausgelöst hat.

5.3.2 Validierungen

- Die verfügbaren Daten aus einem Ereignis müssen ausreichen, um die Umsetzungsdaten für die zugehörige Prozessdefinition vollständig aufzufüllen.
- Wenn ein Prozess bereits für eine ereignisbasierte Umsetzung konfiguriert wurde, müssen nachfolgende Änderungen an den Prozessumsetzungsdaten den bereits vorhandenen Ereignisdatenzuordnungen Rechnung tragen.
- Wenn ein Prozess so konfiguriert wurde, dass er aus einem Ereignis umgesetzt wird, darf die neueste Version des Prozesses nicht gelöscht werden, wenn für die Vorgängerversion der neuesten Version die Umsetzungsdaten nicht vollständig aus dem Ereignis aufgefüllt werden können.

Kapitel 6. Basisaktivität

6.1 Übersicht

Alle von Cúram Workflow unterstützten Aktivitäten weisen einige grundlegende Gemeinsamkeiten auf. Mithilfe dieser Informationen können sie eindeutig von der Workflow-Engine identifiziert werden und können somit sowohl in Text- als auch in Graphform im Prozessdefinitionstool dargestellt werden. Jede Aktivität verfügt über einen Namen und eine optionale Beschreibung, die beide lokalisierbar sind. Dadurch können in verschiedenen Administrationsbenutzerschnittstellen die Informationen in der gewünschten Ländereinstellung angezeigt werden.

Durch diese Einheitlichkeit auf Basisebene können Aktivitäten von der Workflow-Engine ermittelt und ausgeführt werden, ohne dass dafür der Typ der Aktivität bekannt sein muss. Jeder Aktivitätstyp kennt seine eigenen Metadaten und verhält sich bei der Ausführung entsprechend. Somit können bei Bedarf neue Aktivitätstypen hinzugefügt werden, ohne dass das Kernverhalten der Workflow-Engine beeinflusst wird.

6.2 Metadaten

```
<automatic-activity id="1" category="AC1">
    <name>
      <localized-text>
        <locale language="en">ApproveCase</locale>
      </localized-text>
    </name>
    <description>
      <localized-text>
        <locale language="en">This automatic activity
        will be executed to approve a case.</locale>
      </localized-text>
    </description>

    ...

  </automatic-activity>
```

id Hierbei handelt es sich um eine 64-Bit-Kennung, die vom Cúram-Schlüsselserver bereitgestellt wird, wenn Aktivitäten im Prozessdefinitionstool erstellt werden. Die Aktivitätskennung muss innerhalb einer Prozessdefinition eindeutig sein. Eine globale Eindeutigkeit in allen Prozessdefinitionen im System ist jedoch nicht erforderlich.

category
Eine Aktivität kann optional in eine Kategorie eingefügt werden. Die Kategorie muss im Prozessdefinitionstool ausgewählt werden und stammt aus der Codetabelle `ActivityCategory`. Dieses Attribut kann für die aktivitätsbasierte Suche verwendet werden und hat keine funktionale Wirkung auf die Aktivität.

name Anhand ihres Namens kann eine Aktivität zu Anzeigezwecken angegeben werden. Im Gegensatz dazu wird die Kennung einer Aktivität verwendet, um die Aktivität zu ermitteln, die durch die Workflow-Engine ausgeführt werden soll.

description
Eine Aktivität kann außerdem eine kurze optionale Beschreibung aufweisen, die den Verwendungszweck angibt. Diese kann für die Benutzer hilfreich sein, die die Prozessdefinition zukünftig bearbeiten.

© Copyright IBM Corp. 2012, 2013

6.2.1 Lokalisierter Text

Wie im XML-Fragment oben dargestellt sind der Aktivitätsname und die Beschreibung nicht nur reine Textfelder, sondern werden anhand eines Elements des Typs `localized-text` definiert. Dabei handelt es sich um ein allgemeines Element, das im Rahmen der Prozessdefinition in den gesamten Metadaten immer an der Stelle verwendet wird, an der Text lokalisierbar ist.

Ein gültiges Element des Typs `localized-text` muss mindestens ein untergeordnetes Element des Typs `locale` aufweisen. Dadurch wird sichergestellt, dass für ein bestimmtes Feld immer Text zum Anzeigen vorhanden ist. Im Prozessdefinitionstool wird jeder lokalisierbare Text, der in den meisten Benutzerschnittstellenbildschirmen eingegeben wird, in der standardmäßig verwendeten Serverländereinstellung gespeichert, wie sie von der Anwendungseigenschaft 'curam.environment.default.locale' angegeben ist.

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

locale Enthält den Text für die durch die Attribute `language` und `country` festgelegte Ländereinstellung. Hinweise: Eine Ländereinstellung wird sowohl durch die Sprache als auch das Land eindeutig angegeben. Das bedeutet, dass *en*, *en_US* und *en_GB* jeweils für unterschiedliche Ländereinstellungen stehen.

language
Dieser Tag ist obligatorisch und stellt den aus zwei Buchstaben bestehenden ISO-Sprachencode dar.

country
Dieser Tag ist optional und ist der aus zwei Buchstaben bestehende ISO-Ländercode.

6.3 Validierungen

- Der Aktivitätsname ist obligatorisch und muss innerhalb einer angegebenen Prozessdefinition für einen Workflow eindeutig sein. Bei dem Aktivitätsname handelt es sich jedoch auch um eine lokalisierbare Zeichenfolge. Eine Validierung stellt daher zudem sicher, dass ein definierter Aktivitätsname auch für jede angegebene Ländereinstellung eindeutig ist.
- Eine Aktivität muss einen zulässigen Aktivitätstyp aufweisen. In der Praxis genügt diese Regel sich selbst, da es nicht möglich ist, Aktivitäten zu erstellen, ohne dabei einen angemessenen Aktivitätstyp im Prozessdefinitionstool auszuwählen. Selbst beim manuellen Erstellen von Prozessdefinitionen in einem Texteditor entsprechen die Namen der Aktivitätstypen den Namen der Metadatenelementen und machen es somit unmöglich, ein gültiges Markup zu erstellen, das einen nicht vorhandenen Aktivitätstyp darstellt.

6.4 Basisaktivitätstypen

Einige Aktivitätstypen, nämlich Startprozess- und Endprozessaktivitäten, weisen ausschließlich die Metadaten auf, die für alle Aktivitätstypen gelten (verfügen also nicht über zusätzliche Metadaten). Ihr Verhalten ist intuitiv und wird daher an dieser Stelle nicht näher erläutert. Allen anderen Aktivitätstypen ist jeweils ein eigenes Kapitel gewidmet.

6.4.1 Weiterleitungsaktivität

Eine Weiterleitungsaktivität ist eine Aktivität, mit der keinerlei Geschäftsfunktion ausgeführt wird. Sie kann als Nullaktivität angesehen werden, da ihre Ausführung sich weder auf die Anwendungsdaten noch auf den Geschäftsprozess auswirkt.

Der vorrangige Zweck der Weiterleitungsaktivität besteht darin, die Ablaufsteuerung zu unterstützen. Weiterleitungsaktivitäten werden oftmals als Verzweigungs- (Split) und Synchronisationspunkte (Join) verwendet. Sie erweisen sich zudem als hilfreich, wenn die von einem Geschäftsprozess benötigten Aktivitäten nicht von sich aus eine gültige Blockstruktur bilden, die die Workflow-Engine ausführen kann.

Da allen Aktivitätstypen Benachrichtigungen zugeordnet sein können (siehe hierzu Kapitel 14, „Aktivitätsbenachrichtigungen", auf Seite 101), können Weiterleitungsaktivitäten verwendet werden, um die Wirkung einer reinen Benachrichtigung zu erzielen, die mit keiner anderen Funktionalität verknüpft ist.

6.4.2 Start-/Endprozessaktivität

Die Start- und Endprozessaktivitäten stellen Datenpunkte für den Start und für das Ende eines Prozesses bereit. Diese Punkte sind Ankerpunkte, an die andere Aktivitäten mithilfe von Übergängen angehängt werden können. Dadurch entsteht eine Reihe von Schritten vom Start bis zum Ende des Prozesses. In einer gültigen Prozessdefinition sollte das Traversieren aller Übergänge zwischen den von der Startprozessaktivität ausgehenden Aktivitäten zur Endprozessaktivität führen. Beachten Sie, dass in einer Prozessinstanz, die ausgeführt wird, nicht unbedingt alle Wege verfolgt werden. Wenn beispielsweise ein Split (siehe Kapitel 17, „Splits/Joins (Verzweigungs-/Synchronisationspunkte)", auf Seite 119) auftritt, werden je nach Bewertung der Übergangsbedingungen möglicherweise nur einige Wege verfolgt. Die einfachste Prozessdefinition ist demnach die, die nur diese beiden Aktivitäten und einen Übergang von der Start- zur Endprozessaktivität enthält.

Jede Prozessdefinition muss genau eine Start- und eine Endprozessaktivität aufweisen. Bei der Definition eines Prozesses mithilfe des Prozessdefinitionstools werden diese beiden Aktivitäten automatisch bei der Prozesserstellung generiert und müssen (bzw. können) nicht vom Benutzer erstellt werden.

Die Start- und Endprozessaktivitäten bilden den äußersten Block einer gültigen Prozessdefinition in Blockform, wie sie im Rahmen von Cúram Workflow erforderlich ist.

Kapitel 7. Automatisch

7.1 Voraussetzungen

- Die Basisdetails, die für alle von Cúram Workflow unterstützten Aktivitätstypen gelten, sind in Kapitel 6, „Basisaktivität", auf Seite 31 beschrieben und gelten für die nachfolgend beschriebene automatische Aktivität.

7.2 Übersicht

Eine automatische Aktivität ist eine vollständig automatisierte Aktivität in einem Workflowprozess. Unter normalen Umständen ist für das Abschließen eines solchen Schritts kein manueller Eingriff erforderlich. Bei einem Schritt mit einer automatischen Aktivität wird in der Anwendung eine Methode zum Durchführen eines Teils der im Rahmen des gesamten Geschäftsprozesses erforderlichen Verarbeitung aufgerufen. Automatische Aktivitäten werden für gewöhnlich in den folgenden Bereichen verwendet: Durchführen von Berechnungen, Aktualisieren von Entitäten in der Anwendung und Einspeisen von Daten in die Workflow-Engine.

7.3 Cúram-Geschäftsmethoden

Ein Großteil der Verarbeitung für eine automatische Aktivität wird im aufgerufenen Anwendungscode durchgeführt. Automatische Aktivitäten funktionieren durch das Aufrufen von Cúram-Geschäftsmethoden. Dabei werden sowohl die GPO- (Geschäftsprozessobjekt) als auch die Entitätsmethode unterstützt. Technisch gesehen handelt es sich hierbei um öffentliche Methoden für Cúram-Geschäftsprozessobjekte und -Entitäten. Ein wichtiger Teil der automatischen Aktivitätsdefinition sind die aufzurufende Methode und die weiterzugebenen Parameter. Weitere Informationen hierzu finden Sie in den nachfolgenden Abschnitten.

7.3.1 Metadaten

```
<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

Enthält die Details der Cúram-Geschäftsmethode, die aufgerufen wird, wenn die zugehörige automatische Aktivität ausgeführt wird. Zu diesen Details gehören der Name der Schnittstelle und die zugehörige Methode sowie alle Eingabe- und Rückgabezuordnungen, die mit der aufgerufenen Methode verbunden sind. Die Eingabe- und Ausgabezuordnungen werden in den folgenden Abschnitten beschrieben. Die obligatorischen Attribute einer Geschäftsprozessobjektzuordnung (GPO-Zuordnung) sind nachfolgend aufgeführt.

interface-name

Gibt den vollständig qualifizierten Namen der Cúram-Schnittstelle an, die die der automatischen Aktivität zugeordnete Methode enthält.

© Copyright IBM Corp. 2012, 2013

method-name
Steht für die Methode der angegebenen Cúram-Schnittstelle, die aufgerufen wird, wenn die automatische Aktivität ausgeführt wird.

7.3.2 Validierungen

- Sowohl der Name der Schnittstelle als auch der Name der Methode müssen für die Zuordnung der Methode für das Geschäftsprozessobjekt für die automatische Aktivität angegeben werden.
- Der angegebene Schnittstellenname muss eine gültige Klasse sein und die Klasse muss im Klassenpfad der Cúram-Anwendung vorhanden sein.
- Der Methodenname muss ein gültiger Methodenname sein und in der angegebenen Schnittstelle vorhanden sein.

7.3.3 Code

Wie bereits zuvor erwähnt, können alle gültigen öffentlichen Cúram-Geschäftsmethoden (GPO oder Entität) einer automatischen Aktivität in einem Workflowprozess zugeordnet und daher beim Ausführen dieser Aktivität aufgerufen werden. Normalerweise führt das Fehlgeschlagen einer solchen Methode beim Ausführen einer automatischen Aktivität dazu, dass die Strategie zur Handhabung von Workflow-Fehlern aufgerufen wird. Dies kann beispielsweise dazu führen, dass mehrmals versucht wird, die der fehlgeschlagenen Methode zugeordnete Aktivität erneut auszuführen. Basierend auf dieser Tatsache sollten für Methoden, die automatischen Aktivitäten zugeordnet sind, möglichst keine Ausnahmen auftreten. Wenn die modellierte Ausnahmefunktion verwendet wird und eine GPO-Methode auch nach mehrmaligen Ausführen eine Ausnahme auslöst, werden alle Übergänge von der automatischen Aktivität bewertet, die das Workflowdatenobjekt `Context_Error` enthalten. Für die Übergänge, die mit "true" bewertet werden, werden die entsprechenden Wege verfolgt. So kann nach dem Fehlschlagen einer GPO-Methode durch eine weitere Verarbeitung der Prozess nachgebessert werden.

7.4 Eingabezuordnungen

Es muss einen Weg geben, die Parameter, die von einer Methode zum Aufrufen der Methode in der Workflow-Engine benötigt werden, abzurufen. Die Workflow-Engine verfügt über einen eigenen Datenpool in Form von Workflowdatenobjekten (siehe Kapitel 4, „Workflowdatenobjekte", auf Seite 17). Mithilfe von Eingabezuordnungen kann angegeben werden, welche Attribute der Workflowdatenobjekte zum Auffüllen der Werte der entsprechenden Methodenparameter verwendet werden, wenn die Methode aufgerufen wird. Eingabezuordnungen können optional ausgewählt werden, wenn Strukturfelder als Methodenparameter angegeben wurden. Parameter als primitive Basistypen müssen jedoch zugeordnet werden.

7.4.1 Metadaten

Die folgenden Metadaten gelten allgemein für alle drei Typen von Eingabezuordnungen für Parameter (Basistyp, Struktur und aggregierte Strukturen) und werden daher nicht noch einmal beschrieben.

formal-parameters
Enthält die Liste der formalen Parameter, wie in der Geschäftsmethodensignatur der automatischen Aktivität definiert.

formal-parameter
Enthält die Details einer Eingabezuordnung für einen formalen Parameter, wie in der zugehörigen Geschäftsmethodensignatur definiert. In diesem Fall ist für jeden in der zugehörigen Geschäftsmethode definierten Parameter ein Zuordnungseintrag vorhanden.

index Gibt die Position des formalen Parameters in der Liste der formalen Parameter an, die für die angegebene Methode definiert sind. Es handelt sich um einen nullbasierten Index.

7.4.1.1 Eingabezuordnungen für Basisparameter

Basisparameter stellen die einfachste Art der Eingabezuordnungen bereit. In diesem Fall werden Eingabezuordnungen für alle formalen Basisparameter in einer Geschäftsmethode erstellt, die der automatischen

Aktivität zugeordnet ist. Ein Basisparameter in einer Cúram-Geschäftsmethode stellt eine Domänendefinition dar (Informationen zu Domänendefinitionen finden Sie im Handbuch *Cúram Modeling Reference Guide*).

```
<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
Enthält die Details einer Eingabezuordnung für Basistypen. Eine Basistypzuordnung gibt an, dass das Feld, zu dem die Zuordnung erfolgt, primitiv ist (im Gegensatz zu den Struktur- und verschachtelten Strukturzuordnungen, die weiter unten beschrieben sind). Eine Basistypzuordnung enthält das folgende obligatorische Attribut:

type Beschreibt den Typ des primitiven Feldes, zu dem die Zuordnung erfolgt. Bei einer Basistypzuordnung ist dies die Art der Domänendefinition, die als formaler Parameter in der Methode angegeben ist.

wdo-attribute
Enthält die Details des Attributs des Workflowdatenobjekts (siehe Kapitel 4, „Workflowdatenobjekte", auf Seite 17), das die Daten enthält, die zum Auffüllen des zugehörigen Basistypparameters verwendet werden, wenn die Geschäftsmethode der automatischen Aktivität aufgerufen wird. Die obligatorischen Attribute sind nachfolgend beschrieben:

wdo-name
Beschreibt den Namen des Workflowdatenobjekts, das in der Eingabezuordnung verwendet wird.

name Beschreibt den Namen des Attributs eines bestimmten Workflowdatenobjekts, das in der Eingabezuordnung verwendet wird.

7.4.1.2 Eingabezuordnungen für Strukturparameter

Strukturen können als Parameter für Methoden für Geschäftsprozessobjekte festgelegt werden. In diesem Abschnitt werden die Metadaten der Eingabezuordnungen für solche Parameter beschrieben.

```
<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>
```

struct Enthält die Details zu einer Struktureingabezuordnung, einschließlich des Typs der Struktur sowie der Zuordnungen für die einzelnen in dieser Struktur definierten Felder. Eine Struktureingabezuordnung enthält die folgenden obligatorischen Attribute:

type Beschreibt den Typ der Struktur, die als formaler Parameter in der Methode angegeben ist. Das Attribut wird als vollständig qualifizierter Name der als formaler Parameter definierten Struktur dargestellt.

field Enthält die Details der Eingabezuordnung für eines der Felder, die im Strukturparameter definiert sind. Ein Feld enthält die Details der Eingabezuordnung für den primitiven Basistyp dieses Feldes sowie das folgende obligatorische Attribut:

name Beschreibt den Namen des Feldes, wie in der als formaler Parameter definierten Struktur definiert.

base-type
Enthält die Details einer Eingabezuordnung für Basistypen für das angegebene Feld. Eine Basistypzuordnung enthält das folgende obligatorische Attribut:

type Beschreibt den Typ des primitiven Feldes, zu dem die Zuordnung erfolgt.

wdo-attribute
Enthält die Details des Attributs des Workflowdatenobjekts (siehe Kapitel 4, „Workflowdatenobjekte", auf Seite 17), das die Daten enthält, die zum Auffüllen des zugehörigen Basistypfeldes verwendet werden, wenn die Methode aufgerufen wird. Dieses Element ist nicht vorhanden, wenn der Benutzer für diesen Methodenparameter keine Eingabezuordnung festgelegt hat. Wenn das Element angegeben ist, enthält es die folgenden obligatorischen Attribute:

wdo-name
Beschreibt den Namen des Workflowdatenobjekts, das in der Eingabezuordnung verwendet wird.

name Beschreibt den Namen des Attributs eines bestimmten Workflowdatenobjekts, das in der Eingabezuordnung verwendet wird.

7.4.1.3 Eingabezuordnungen für aggregierte Strukturparameter

Aggregierte Strukturen (Informationen zur Aggregation von Strukturen finden Sie im Handbuch *Cúram Modeling Reference Guide*) können als Parameter für Geschäftsmethoden angegeben werden. In diesem Fall ähneln die Metadaten denen der bereits weiter oben beschriebenen formalen Strukturparameter (siehe 7.4.1.2, „Eingabezuordnungen für Strukturparameter", auf Seite 37). Der einzige Unterschied hierbei besteht jedoch darin, dass ein Feld im definierten Strukturparameter auf eine andere darunterliegende Struktur aufgelöst wird (und nicht auf einen primitiven Typ, wie im Beispiel für die Strukturzuordnung gezeigt). In diesem Szenario ist der Feldname nicht der Name des zugeordneten Feldes, das dem Strukturparameter zugeordnet ist, sondern der Name der Rolle, die in der Zuordnung zwischen der angegebenen Struktur und der Struktur, die aggregiert wird, enthalten ist. Die folgenden Metadatenausschnitte sind ein Beispiel für solche Eingabezuordnungen. Die Metadatenelemente wurden bereits im Abschnitt zu den Eingabezuordnungen für Strukturen beschrieben.

```
<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>
```

7.4.1.4 Eingabezuordnungen für Parameter von Listenstrukturen

Es können nun auch Eingabezuordnungen für Parameter von Listenstrukturen angegeben werden. In diesem Fall ähneln die Metadaten denen der bereits weiter oben beschriebenen formalen Aggregatparameter (siehe 7.4.1.3, „Eingabezuordnungen für aggregierte Strukturparameter"). Der in den Metadaten für einen Listenstrukturparameter angegebene Typ ist der Name der Listenstruktur. Der Name des ersten Feldes gibt den Namen der Rolle an, die in der Zuordnung zwischen der festgelegten Listenstruktur und der untergeordneten Struktur, die aggregiert wird, enthalten ist. Normalerweise erfolgt dann eine Auflösung

von diesem Feld auf eine weitere Struktur (die untergeordnete Struktur innerhalb der Listenstruktur). Das in einer solchen Zuordnung angegebene Workflowdatenobjekt ist ein Listen-Workflowdatenobjekt. Die folgenden Metadatenausschnitte sind ein Beispiel für solche Eingabezuordnungen. Die Metadatenelemente wurden bereits im Abschnitt zu den Eingabezuordnungen für Strukturen beschrieben.

```
<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>
```

7.4.1.5 Eingabezuordnungen und indexierte Elemente aus Listen-Workflowdatenobjekten

Bei Aktivitäten in Schleifen kann ein Element aus einem Listen-Workflowdatenobjekt in einer Eingabezuordnung zum Auffüllen eines Formalparameterfeldes verwendet werden. Wenn diese Art der Eingabezuordnung verwendet wird, wird bei jedem Iterieren der Schleife mit der Aktivität das Formalparameterfeld mit dem nächsten Wert aus dem Listen-Workflowdatenobjekt aufgefüllt. Dies wird an dieser Stelle hervorgehoben, da sich die Metadatensyntax für eine solche Zuordnung leicht von der Syntax der anderen Eingabezuordnungstypen unterscheidet. Der nachfolgende Metadatenausschnitt ist ein Beispiel für solche Eingabezuordnungen. Der Name des zum Auffüllen des Formalparameterfeldes verwendete Name des Listen-Workflowdatenobjekts wird durch die Syntax `[Context_Loop.loopCount]` qualifiziert. Diese wird von der Workflow-Engine zur Laufzeit verwendet, um zu bestimmen, welche Iteration der Schleife ausgeführt wird und welches Element daher aus dem Listen-Workflowdatenobjekt zum Abrufen der Daten verwendet wird, mit denen das Formalparameterfeld aufgefüllt wird.

```
<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>
```

7.4.2 Validierungen

- Die in den Eingabezuordnungen angegebenen Attribute des Workflowdatenobjekts müssen gültig sein. Die Kriterien, die ein gültiges Attribut des Workflowdatenobjekts definieren, sind im Abschnitt 4.3, „Validierungen", auf Seite 21 aufgeführt.
- Der Typ des formalen Parameters, zu dem die Zuordnung hergestellt wird, und der Typ des Workflowdatenobjekt-Attributs, das in dieser Eingabezuordnung verwendet wird, müssen kompatibel sein. Wenn es sich beispielsweise bei der erstellten Eingabezuordnung um ein Strukturfeld handelt, das den Typ 'STRING' aufweist, muss das Attribut des Workflowdatenobjekts für diese Zuordnung auch vom Typ 'STRING' sein.
- Das Workflowdatenobjekt `Context_Task` kann nur in einer Eingabezuordnung verwendet werden, wenn es sich bei der zugehörigen Aktivität um eine manuelle oder eine Entscheidungsaktivität handelt.
- Das Workflowdatenobjekt `Context_Loop` kann nur in einer Eingabezuordnung verwendet werden, wenn die zugehörige Aktivität in einer Schleife enthalten ist.
- Es wird eine Validierungswarnung angezeigt, wenn keiner der in der Methode für das Geschäftsprozessobjekt definierten Strukturparameter eine zugehörige Eingabezuordnung enthält.
- Alle primitiven Basistypformalparameter, die in der GPO-Methode definiert sind, müssen eine zugehörige Eingabezuordnung enthalten.
- Wenn es sich bei dem zuzuordnenden Formalparameterfeld um einen Basistypparameter handelt, kann kein Attribut aus einem Listen-Workflowdatenobjekt verwendet werden.
- Wenn das zuzuordnende Formalparameterfeld aus einer Listenstruktur stammt, muss es einem Attribut aus einem Listen-Workflowdatenobjekt zugeordnet werden.
- Wenn das indexierte Element aus einem Listen-Workflowdatenobjekt (z. B. `ClaimantDependent[Context_Loop.loopCount]`) in einer Eingabezuordnung verwendet wird, dann muss es sich bei dem zugehörigen Workflowdatenobjekt um ein Listen-Workflowdatenobjekt handeln und die Aktivität mit den Eingabezuordnungen muss in einer Schleife enthalten sein.

7.4.3 Laufzeitinformationen

Die Werte der Attribute für das Workflowdatenobjekt, die in den Eingabeparameterzuordnungen definiert sind, werden der angegebenen Methode als Eingabedaten bereitgestellt, bevor die Methode beim Ausführen der zugehörigen automatischen Aktivität aufgerufen wird.

7.5 Ausgabezuordnungen

Bei Workflowdatenobjekten (siehe Kapitel 4, „Workflowdatenobjekte", auf Seite 17) handelt es sich um den Datenspeicher der Workflow-Engine. Einige der Attribute in den angegebenen Workflowdatenobjekten werden beim Start des Prozesses mit Daten aufgefüllt. Es ist jedoch hilfreich, die Werte der Attribute der Workflowdatenobjekte während der Ausführung des Workflowprozesses zu aktualisieren oder festzulegen. Um diese Vorgehensweise zu unterstützen, können einige Aktivitätstypen Daten zurück in die Workflow-Engine speisen. Dies ist insbesondere für automatische Aktivitäten von Vorteil, da die Geschäftsmethoden, die die Aktivitäten aufrufen, auf Daten in jeder Entität in der Anwendung zugreifen und diese Daten für die Verwendung in nachfolgenden Aktivitäten im Workflowprozess zurückgeben könnten. Diese Rückgabezuordnungen von einer Methode für das Geschäftsprozessobjekt, die einer automatischen Aktivität zugeordnet sind, sind optional.

7.5.1 Metadaten

Ähnlich wie Eingabezuordnungen (siehe 7.4, „Eingabezuordnungen", auf Seite 36) werden Ausgabezuordnungen für primitive Rückgabetypen sowie für Rückgabetypen für Strukturen, verschachtelte (aggregierte) Strukturen und für Listenstrukturen unterstützt. Wenn es sich bei dem Rückgabetyp um einen primitiven Typ handelt, kann ein Rückgabezuordnungseintrag angegeben werden. Wenn der Rückgabetyp eine Struktur, eine aggregierte Struktur oder eine Listenstruktur ist, können Rückgabezuordnungen für mindestens ein Feld in der angegebenen Struktur erstellt werden. Die folgenden Metadatenausschnitte zeigen beispielhaft solche Zuordnungen:

7.5.1.1 Primitiver Rückgabetyp

```
<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>
```

7.5.1.2 Rückgabetyp für Strukturen

```
<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>
```

7.5.1.3 Rückgabetyp für aggregierte Strukturen

```
<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>
```

7.5.1.4 Rückgabetyp für Listenstrukturen

```
<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 Enthält die Details für die Ausgabezuordnungen, die für die Geschäftsmethode angegeben sind, die der automatischen Aktivität zugeordnet ist. Für einen primitiven Rückgabetyp ist ein Eintrag der Basistypmetadaten vorhanden, wie im Beispiel oben dargestellt (siehe 7.5.1.1, „Primitiver Rückgabetyp", auf Seite 42). Für eine Struktur, eine aggregierte Struktur und für Rückgabetypen für Listenstrukturen wird der Metadaten-Tag 'struct' angegeben und enthält Felder, deren Basistypen anhand von Attributen von Workflowdatenobjekten zugeordnet werden.

struct Enthält die Details der Strukturausgabezuordnung. Eine Strukturausgabezuordnung enthält das folgende obligatorische Attribut:

field Enthält die Details der Ausgabezuordnung für eines der Felder, die im Strukturrückgabetyp definiert sind. Ein Feld enthält die Details der Ausgabezuordnung für den primitiven Basistyp dieses Feldes sowie das folgende obligatorische Attribut:

- **name** Beschreibt den Namen des Feldes, wie in der als Rückgabetyp definierten Struktur definiert. Bei Rückgabetypen für nicht aggregierte Strukturen definiert dies einfach den Namen des Feldes in der angegebenen Rückgabestruktur, das zugeordnet wird. Bei Rückgabetypen für aggregierte Strukturen und Listenstrukturen gibt der Feldname den Namen der Rolle an, die in der Zuordnung zwischen der festgelegten Struktur und der Struktur, die aggregiert wird, enthalten ist.

base-type
Enthält die Details einer Ausgabezuordnung für Basistypen für das angegebene Feld oder einen primitiven Rückgabetyp.

wdo-attribute
Enthält die Details des Attributs des Workflowdatenobjekts (siehe Kapitel 4, „Workflowdatenobjekte", auf Seite 17), dem die Daten des zugehörigen Feldes mit dem Rückgabetyp zugeordnet werden und in dem sie gespeichert werden. Die obligatorischen Attribute sind nachfolgend beschrieben:

wdo-name
Beschreibt den Namen des Workflowdatenobjekts, das in der Ausgabezuordnung verwendet wird.

name Beschreibt den Namen des Attributs des Workflowdatenobjekts, das in der Ausgabezuordnung verwendet wird.

7.5.2 Validierungen

- Es sind keine doppelten Ausgabeparameterzuordnungen zulässig. Anders gesagt kann ein Attribut des Workflowdatenobjekts nur einmal in einer Liste mit Ausgabeparameterzuordnungen angegeben werden.
- Alle in den Ausgabezuordnungen angegebenen Attribute des Workflowdatenobjekts müssen bezüglich der zugehörigen Workflowprozessdefinition gültige Attribute sein.
- Der Typ des Rückgabefeldes, von dem aus die Zuordnung erfolgt, und der Typ des Workflowdatenobjekt-Attributs, zu dem die Zuordnung hergestellt wird, müssen kompatibel sein.
- Ausgabezuordnungen können nicht für Attribute des Workflowdatenobjekts erstellt werden, die als konstante Attribute markiert wurden. Konstante Attribute des Workflowdatenobjekts stellen Daten dar, die für die Dauer der Prozessinstanz konstant bleiben sollten (siehe 4.2, „Metadaten", auf Seite 18). Wenn diese Attribute in Ausgabezuordnungen verwendet werden dürften, würden diese Daten mit den Daten aus den Ausgabezuordnungen überschrieben werden.
- Wenn es sich bei der Rückgabestruktur um eine Listenrückgabestruktur handelt, dann muss das in der Rückgabezuordnung verwendete Workflowdatenobjekt ein Listen-Workflowdatenobjekt sein.

7.5.3 Laufzeitinformationen

Die Werte der Rückgabetypfelder, die in den Ausgabeparameterzuordnungen definiert sind, werden nach dem Ausführen der zugehörigen automatischen Aktivität mithilfe des angegebenen Workflowdatenobjekt-Attributs beibehalten.

7.6 Beschreibung der Workflowdatenobjekte des Typs 'Context'

Es gibt zwei Workflowdatenobjekte des Typs 'Context', die beim Erstellen von Datenelement- und Funktionsbedingungen für Übergänge von einer automatischen Aktivität verfügbar sind. Diese werden nachfolgend beschrieben.

Workflowdatenobjekt 'Context_Result'
Das Workflowdatenobjekt 'Context_Result' kann in Datenelement- oder Funktionsbedingungen (siehe Kapitel 16, „Bedingungen", auf Seite 113) für einen Übergang von einer automatischen Aktivität verwendet werden. Dadurch kann der Rückgabewerte der aufgerufenen Methode in den besagten Bedingungen verwendet werden. Die Konventionen für die für das Workflowdatenobjekt 'Context_Result' verfügbaren Attribute sind wie folgt:

- Wenn der Rückgabetyp ein Basistyp ist, lautet das verfügbare Attribut `value` (d. h. `Context_Result.value`).
- Wenn der Rückgabewert eine Struktur ist, dann sind die verfügbaren Attribute für 'Context_Result' alle Felder, die in der Strukturrückgabeklasse vorhanden sind (d. h. `Context_Result.description` usw.).

- Wenn der Rückgabewert eine verschachtelte (aggregierte) Struktur ist, dann sind die verfügbaren Attributwerte für 'Context_Result' die Felder in der übergeordneten Struktur (d. h. `Context_Result.description` usw.) und zudem die vollständig qualifizierten Namen der Felder in den verschachtelten Strukturen (d. h. `Context_Result.dtls:concernRoleID` usw.). Unabhängig von der Tiefe der Verschachtelung der Struktur der Rückgabewerte gibt es nur ein verfügbares Workflowdatenobjekt des Typs 'Context_Result', bei dem die Namen der verschachtelten Strukturen ein Teil der Attributnamen sind. Das Trennzeichen zwischen einer verschachtelten Struktur und ihren Feldern ist ein Doppelpunkt (wie im obigen Bespiel dargestellt).
- Wenn der Rückgabetyp eine Listenstruktur ist, ist das Workflowdatenobjekt `Context_Result` nicht verfügbar.

Workflowdatenobjekt 'Context_Error'

Eine durch eine automatische Aktivität aufgerufene GPO-Methode kann manchmal fehlschlagen (d. h. eine Ausnahme auslösen, die dazu führt, dass für eine Aktivitätstransaktion ein Rollback durchgeführt wird). Wenn dies passiert, kann es hilfreich sein, nach dem Fehlschlagen Folgeaktionen modellieren zu können. Mithilfe des Workflowdatenobjekts `Context_Error` ist diese Art der "Fehlerpfad"-Modellierung möglich. Das Objekt kann in Datenelement- oder Funktionsbedingungen (siehe Kapitel 16, „Bedingungen”, auf Seite 113) für einen Übergang von einer automatischen Aktivität verwendet werden. Das Workflowdatenobjekt `Context_Error` verfügt über das Attribut `exceptionOccurred`, das nachfolgend beschrieben wird:

- Das Attribut `exceptionOccurred` ist ein boolescher Wert, der angibt, wenn die einer automatischen Aktivität zugeordnete GPO-Methode fehlgeschlagen ist. Die Standardeinstellung lautet "false". Der Wert wird auf "true" gesetzt, wenn die zugehörige GPO-Methode fehlschlägt.

Wenn zur Laufzeit die in einer automatischen Aktivität aufgerufene GPO-Methode fehlschlägt (und selbst nach einer vorgegebenen Anzahl von Versuchen weiterhin fehlschlägt), setzt die Workflow-Engine das Attribut `exceptionOccurred` für `Context_Error` auf "true". Alle Übergänge, die das Workflowdatenobjekt `Context_Error` verwenden, werden dann bewertet und weiterfolgt, wenn sie auf den Wert "true" aufgelöst werden. Dadurch kann eine Workflowprozessinstanz auf dem vorgegebenen Fehlerpfad weitergeführt werden, obwohl die automatische Aktivität fehlgeschlagen ist.

Falls die aufgerufene GPO-Methode fehlschlägt und *keine* Übergänge mit dem Workflowdatenobjekt `Context_Error` verwendet werden, wird die Aktivität angehalten und in der Administrationskonsole für Nachrichten des Typs 'Fehlgeschlagen' wird ein entsprechender Eintrag erstellt.

Anmerkung: Das Workflowdatenobjekt `Context_Error` berücksichtigt dabei nicht den *Grund* des Fehlers, sondern nur die Tatsache, ob ein Fehler aufgetreten ist oder nicht.

Kapitel 8. Event-Wait

8.1 Voraussetzungen

- Die Basisdetails, die für alle von Cúram Workflow unterstützten Aktivitätstypen gelten, sind in Kapitel 6, „Basisaktivität", auf Seite 31 beschrieben und gelten für die nachfolgend beschriebene Event-Wait-Aktivität.

8.2 Übersicht

Die Cúram-Anwendung kann Ereignisse zu verschiedenen Zeitpunkten auslösen und registrierte Listener darüber informieren, was passiert ist. Für ein angegebenes Ereignis können mehrere unterschiedliche Ereignislistener registriert sein. Diese Ereignislistener sind Anwendungsfunktionen, die die Schnittstelle `curam.util.events.impl.EventHandler` implementieren. Wenn ein angegebenes Ereignis ausgelöst wird, ruft die Workflow-Engine die zugehörige Ereignishandlerfunktion auf (weitere Informationen zu Ereignissen und zum Ereignishandler finden Sie im Handbuch *Cúram Server Developers Guide*).

Im Rahmen des Workflows wird diese Funktion mithilfe von Event-Wait-Aktivitäten etwas anders verwendet. Eine Event-Wait-Aktivität unterbricht die Ausführung eines bestimmten Teils einer Prozessinstanz so lange, bis ein bestimmtes Ereignis aufgetreten ist.

8.3 Liste der Ereignisse

Die Aussage, dass eine Event-Wait-Aktivität einen Workflowprozess so lange anhält, bis ein bestimmtes Ereignis ausgelöst wird, ist nicht ganz richtig. Ein Event-Wait kann tatsächlich eine beliebige Anzahl von Ereignissen angeben, auf die gewartet werden soll. Wenn angegeben wurde, dass nicht auf alle diese Ereignisse gewartet werden soll, die zum Abschließen der Aktivität ausgelöst werden müssen, schließt das erste Ereignis, das einem der angegebenen Event-Waits entspricht, die Aktivität ab und setzt den Workflow fort. In diesem Szenario hat der Umstand, ob der Rest der Ereignisse jemals ausgelöst wird, keinerlei Auswirkungen auf den Prozess. Es kann zudem festgelegt werden, dass alle Event-Waits den zugehörigen ausgelösten Ereignissen entsprechen müssen, bevor die Aktivität abgeschlossen und die Bearbeitung des Workflowprozesses fortgesetzt werden kann.

© Copyright IBM Corp. 2012, 2013

8.3.1 Metadaten

```
<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
Enthält die Details zu dem der angegebenen Aktivität zugeordneten Event-Wait. Dazu zählen Informationen zu allen Ereignissen für das Event-Wait.

wait-on-all-events
Der Wert dieses Flags zeigt der Workflow-Engine an, dass sie darauf warten soll, bis Ereignisse für alle der angegebenen Event-Waits ausgelöst wurden, bevor sie die zugehörige Aktivität abschließt. Wenn der Wert auf "false" gesetzt ist, führt das erste Ereignis, das einem der angegebenen Event-Waits entspricht, zum Abschluss der zugehörigen Aktivität und zur weiteren Bearbeitung des Workflows. Ist der Wert auf "true" gesetzt, muss für jedes der angegebenen Event-Waits ein Ereignis ausgelöst werden, bevor die Aktivität abgeschlossen und der Workflow fortgesetzt wird.

events Enthält die Details aller Ereignisse, auf die die angegebene Aktivität wartet.

event Enthält die Details für ein bestimmtes Ereignis, auf das die Aktivität wartet. Die Ereignisdetails enthalten die folgenden obligatorischen Attribute:

event-class
Stellt die Klasse des Geschäftsereignisses dar, auf das dieser Prozess wartet.

event-type
Stellt den Typ des Geschäftsereignisses dar, auf das dieser Prozess wartet. Die Kombination aus 'event-class' und 'event-type' bezeichnet das erforderliche Geschäftsereignis.

identifier
Stellt die eindeutige Kennung dieses Ereignisses dar. Die Kennung muss nur in der Liste der Ereignisse für diese Aktivität eindeutig sein.

event-match-attribute
Stellt das Workflowdatenobjekt-Attribut dar (siehe Kapitel 4, „Workflowdatenobjekte”, auf Seite 17

17), das für den Abgleich mit der erforderlichen Instanz des angegebenen Ereignisses verwendet wird. Beispiel: Im ersten in den oben aufgeführten Metadaten angegebenen Ereignis bezieht sich das Workflowdatenobjekt-Attribut auf die Aufgabenkennung, die dem Schließen einer bestimmten Aufgabe zugeordnet ist. Wenn dieses Ereignis ausgelöst wird, verwendet die Workflow-Engine die Daten im Ereignisübereinstimmungsattribut zur eindeutigen Identifizierung der Aufgabe, die geschlossen werden soll.

multiple-occurring-event
Gibt an, dass dieses Ereignis ein mehrfach auftretendes Ereignis darstellt. Wenn also diese Metadaten für ein Ereignis angegeben werden, erstellt die Workflow-Engine beim Ausführen der Aktivität einen Event-Wait-Datensatz für jedes Element im Listen-Workflowdatenobjekt, das als das mehrfach auftretende Ereignis festgelegt wurde. Dies ermöglicht der Workflow-Engine, auf ein mehrfaches Auftreten desselben Ereignisses zu warten.

Wenn ein Ereignis als mehrfach auftretendes Ereignis verwendet wird, muss ein Attribut des verknüpften Listen-Workflowdatenobjekts als Ereignisübereinstimmungsdaten für das Ereignis verwendet werden. Dadurch wird sichergestellt, dass alle von der Workflow-Engine für das mehrfach auftretende Ereignis generierten Ereignisse jeweils eindeutig sind.

list-wdo-name
Beschreibt den Namen des Listen-Workflowdatenobjekts, das als mehrfach auftretendes Ereignis verwendet werden soll.

8.3.2 Validierungen

- Es muss mindestens ein Ereignis für die mit einer Event-Wait-Aktivität verknüpften Event-Wait-Informationen definiert werden.
- Die für die einzelnen Geschäftsereignisse angegebene Ereignisklasse und der Typ müssen in den entsprechenden Ereignisdatenbanktabellen gültige Einträge sein.
- Ein Ereignis und das zugehörige Ereignisübereinstimmungsattribut können jeweils nur einmal in einer Event-Wait-Aktivität definiert werden. Das bedeutet, dass dieselbe Ereignisklasse, derselbe Ereignistyp und dasselbe Ereignisübereinstimmungsattribut nur einmal als ein bestimmtes Ereignis, auf das für eine Event-Wait-Aktivität gewartet wird, verwendet werden können.
- Das dem Ereignisübereinstimmungsattribut für ein Ereignis zugeordnete Workflowdatenobjekt-Attribut muss ein gültiges Attribut sein. Da es als eindeutige Kennung im Ereignisübereinstimmungsmechanismus verwendet wird, muss es den Typ 'LONG' aufweisen, um den in Cúram verwendeten 64-Bit-Kennungen Rechnung zu tragen.
- Das Workflowdatenobjekt 'Context_Task' kann nur als Workflowdatenobjekt-Attribut für die Ereignisübereinstimmung verwendet werden, wenn es sich bei der Aktivität entweder um eine manuelle oder eine parallele manuelle Aktivität handelt und das Ereignis kein mehrfach auftretendes Ereignis ist.
- Wenn ein indexiertes Element aus einem Listen-Workflowdatenobjekt (z. B. 'ParentList[Context_Loop-.loopCount]') als Ereignisübereinstimmungsdaten verwendet wird, muss das Workflowdatenobjekt ein Listen-Workflowdatenobjekt und die Aktivität mit der Ereigniszuordnung muss in einer Schleife enthalten sein.
- Wenn ein indexiertes Element aus dem parallelen Listen-Workflowdatenobjekt als Ereignisübereinstimmungsdaten verwendet wird, muss die Aktivität mit der Zuordnung eine parallele Aktivität sein (z. B. 'ParallelListWDO[Context_Parallel.occurrenceCount]'). Das anhand des Workflowdatenobjekts 'Context-_Parallel' indexierte Workflowdatenobjekt muss ein Listen-Workflowdatenobjekt für eine parallele Aktivität sein.
- Wenn das Listen-Workflowdatenobjekt des mehrfach auftretenden Ereignisses nicht für das Ereignis angegeben wurde und die Aktivität mit der Ereigniszuordnung keine parallele Aktivität ist, kann ein Attribut aus einem Listen-Workflowdatenobjekt nicht als Ereignisübereinstimmungsdaten für dieses Ereignis verwendet werden.
- Wenn das Listen-Workflowdatenobjekt des mehrfach auftretenden Ereignisses für das Ereignis angegeben wurde, muss ein Attribut aus diesem Listen-Workflowdatenobjekt als Ereignisübereinstimmungsdaten für dieses Ereignis verwendet werden.

- Das als mehrfach auftretende Ereignis zugeordnete Workflowdatenobjekt-Attribut muss ein gültiges Attribut sein. Zudem muss es ein Listen-Workflowdatenobjekt sein.

8.3.3 Code

Es wird ein Worfklowereignishandler für die Prozessumsetzung von Cúram bereitgestellt, der automatisch registriert wird, um in der Anwendung ausgelöste Ereignisse zu überwachen. In einem Workflowprozess können für eine bestimmte Aktivitätsinstanz mehrere Event-Waits registriert werden. Wenn das Flag *waitOnAllEvents* für die angegebenen Event-Wait-Daten auf den Wert "false" gesetzt ist, muss zum Abschließen dieser Aktivitätsinstanz nur eines dieser Event-Waits zugeordnet werden. Der Workflowereignishandler verarbeitet dieses Ereignis, indem er die angegebene Aktivitätsinstanz abschließt und den Prozess durch Starten der nächsten Gruppe von Aktivitäten im Prozess weiterführt. Alle anderen Event-Wait-Datensätze, die für die abgeschlossene Aktivitätsinstanz registriert wurden, werden dann entfernt. Wenn Ausgabezuordnungen (siehe 8.5, „Ausgabezuordnungen", auf Seite 55) für das Event-Wait angegebenen wurden, werden diese von der Workflow-Engine beibehalten und können in nachfolgenden Aktivitäten und Übergängen im Prozess verwendet werden.

Wenn das Flag *waitOnAllEvents* auf "true" gesetzt ist, müssen alle für die Aktivitätsinstanz angegebenen Event-Waits ausgelösten Ereignissen zugeordnet sein, damit die Aktivität abgeschlossen und der Workflow fortgesetzt werden kann. Für jedes ausgelöste Ereignis, das einem zugehörigen Event-Wait für die Aktivitätsinstanz zugeordnet ist, verarbeitet der Workflowereignishandler das Ereignis, indem er den entsprechenden Event-Wait-Datensatz löscht und alle Ausgabezuordnungen (siehe 8.5, „Ausgabezuordnungen", auf Seite 55) beibehält, die für das Event-Wait angegeben wurden. Diese Verarbeitung wird so lange weitergeführt, bis alle zugehörigen Event-Waits ausgelösten Ereignissen zugeordnet wurden. Erst dann kann der Workflowereignishandler die angegebene Aktivitätsinstanz abschließen und den Prozess durch Starten der nächsten Gruppe von Aktivitäten im Prozess fortsetzen.

8.3.4 Laufzeitinformationen

Durch ein in der Anwendung ausgelöstes Ereignis kann die Bearbeitung einer Prozessinstanz nur fortgesetzt werden, wenn das Ereignis mit dem Ereignis übereinstimmt, auf das gewartet wird, und das für das Event-Wait angegebene Ereignisübereinstimmungsattribut den primären Ereignisdaten des Ereignisses entspricht.

8.4 Frist

Ein Event-Wait hält einen Workflowprozess an, anstatt ein Ereignis auszulösen. In vielen Fällen ist es jedoch nicht wünschenswert, einen Prozess auf unbestimmte Zeit anzuhalten. Es kann eine Ereigniskette auftreten, was bedeutet, dass das Ereignis, auf das der Prozess wartet, möglicherweise gar nicht ausgelöst wird. Beispiel: Das Ereignis kann zufällig vor dem Erreichen der Event-Wait-Aktivität ausgelöst werden. Um ein solches Risiko zu minimieren, kann optional eine Frist für ein auszulösendes Ereignis festgelegt werden, nach deren Ablauf ein Fristhandler aufgerufen wird.

8.4.1 Voraussetzungen

- Bei für eine Event-Wait-Frist angegebenen Fristhandlermethoden handelt es sich um Cúram-Geschäftsprozessobjektmethoden. Die Eingabezuordnungen für die formalen Parameter dieser Methoden und deren zugehörigen Metadaten sind in Kapitel 7, „Automatisch", auf Seite 35 näher beschrieben. Dieses Kapitel sollte daher für eine Beschreibung dieser Zuordnungen hinzugezogen werden.

8.4.2 Metadaten

```
<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
: Dieser Tag stellt ein boolesches Flag dar, das anzeigt, ob die Aktivität bei Ablauf der Fristdauer abgeschlossen werden sollte. Der Standardwert für dieses Flag ist "false".

duration
: Zeigt die Zeit an, die vergehen kann, bevor der Fristhandler aufgerufen wird. Diese Dauer kann in einem der unten aufgeführten Formate dargestellt werden, das nachfolgend für das Berechnen der Frist (Datum und Uhrzeit) für das Event-Wait verwendet wird:

  seconds
  : Die Anzahl der Sekunden, die vergehen können, bevor der Fristhandler aufgerufen wird.

mapped-duration
Das Attribut eines Workflowdatenobjekts, das als Anzahl der Sekunden zugeordnet werden kann, die vergehen können, bevor der Fristhandler aufgerufen wird.

deadline-handler
Gibt die Methode an, die nach Ablauf der Fristdauer aufgerufen wird. Für einen Fristhandler müssen die folgenden Metadaten angegeben werden:

interface-name
Stellt den vollständig qualifizierten Namen der Schnittstellenklasse des Fristhandlers dar.

method-name
Gibt die erforderliche Methode in der Fristhandlerschnittstelle an, die bei Ablauf der Frist aufgerufen werden muss.

formal-parameters
Enthält eine Liste der Methodenparameter des Fristhandlers sowie zugehörige Workflowdatenobjekt-Attribute, die diesen Parametern beim Aufruf des Fristhandlers zugeordnet werden. Detaillierte Informationen zu Methodenparameterzuordnungen finden Sie im Abschnitt 7.4, „Eingabezuordnungen", auf Seite 36.

deadline-output-mappings
Enthält die Fristausgabedaten, die optional Workflowdatenobjekt-Attributen zugeordnet werden können. Diese Daten zeigen an, ob die Fristdauer abgelaufen ist oder nicht sowie das Datum und die Uhrzeit für den Ablauf der Fristdauer.

8.4.3 Validierungen

- Wenn ein Fristhandler angegeben ist, muss er eine gültige Cúram-Geschäftsmethode referenzieren, die im Klassenpfad der Anwendung vorhanden ist.
- Die in den Eingabezuordnungen angegebenen Attribute des Workflowdatenobjekts müssen gültig sein. Die Kriterien, die ein gültiges Attribut des Workflowdatenobjekts definieren, sind im Abschnitt 4.3, „Validierungen", auf Seite 21 aufgeführt.
- Der Typ des formalen Parameters, zu dem die Zuordnung hergestellt wird, und der Typ des Workflowdatenobjekt-Attributs, das in dieser Eingabezuordnung verwendet wird, müssen kompatibel sein. Wenn es sich beispielsweise bei der erstellten Eingabezuordnung um ein Strukturfeld handelt, das den Typ 'STRING' aufweist, muss das Attribut des Workflowdatenobjekts für diese Zuordnung auch vom Typ 'STRING' sein.
- Wenn das indexierte Element aus einem Listen-Workflowdatenobjekt (z. B. `ClaimantDependent[Context_Loop.loopCount]`) in einer Eingabezuordnung verwendet wird, dann muss es sich bei dem zugehörigen Workflowdatenobjekt um ein Listen-Workflowdatenobjekt handeln und die Aktivität mit den Eingabezuordnungen muss in einer Schleife enthalten sein.
- Wenn das Workflowdatenobjekt `Context_Parallel` in einer Eingabezuordnung verwendet wird, muss die Aktivität mit der Eingabezuordnung eine Aktivität des Typs `Parallel` sein.
- Wenn ein indexiertes Element aus dem parallelen Listen-Workflowdatenobjekt in einer Eingabezuordnung verwendet wird, muss die Aktivität mit der Zuordnung eine parallele Aktivität sein (d. h. 'ParallelListWDO[Context_Parallel.occurrenceCount]'). Das anhand des Workflowdatenobjekts 'Context_Parallel' indexierte Workflowdatenobjekt muss ein Listen-Workflowdatenobjekt für eine parallele Aktivität sein.
- Die Fristdauer kann entweder in Sekunden oder mithilfe einer Attributzuordnung des Workflowdatenobjekts angegeben werden. Beides zusammen ist nicht möglich.
- Wenn die Fristdauer mithilfe eines Attributs des Workflowdatenobjekts angegeben wurde, muss es sich bei dem Attribut um ein gültiges Attribut des Typs 'INTEGER' handeln.
- Wenn für eine Aktivität eine Frist festgelegt wurde, muss eine Fristhandlerfunktion angegeben und/oder das Flag für vollständige Aktivitäten auf "true" gesetzt werden. Geschieht dies nicht, würde der Workflow nicht weiter verarbeitet werden, wenn die Frist abläuft.

- Wenn der Wert für den Fristablauf der Fristausgabezuordnungen einem Attribut des Workflowdatenobjekts zugeordnet wurde, muss das Attribut ein gültiges Attribut des Typs 'BOOLEAN' sein.
- Wenn der Wert für die Ablaufzeit der Frist der Fristausgabezuordnungen einem Attribut des Workflowdatenobjekts zugeordnet wurde, muss das Attribut ein gültiges Attribut des Typs 'DATETIME' sein.
- Das Flag für die vollständige Aktivität kann nicht auf `true` gesetzt werden, wenn es sich bei der Aktivität mit der Frist um eine parallele Aktivität handelt. Dies ist auf die Tatsache zurückzuführen, dass parallele Aktivitäten keine modellierten Fristen unterstützen.

8.4.4 Code

- Alle der Fristhandlermethode zugeordneten Rückgabeparameter werden nicht in der Workflow-Engine verwendet und sind daher nicht relevant.
- Die API-Funktion `DeadlineScanner.scanDeadlines()` des Workflowfristenscanners kann zum Scannen von Event-Wait-Fristen verwendet werden, die bereits abgelaufen sind. Event-Waits, deren Ablaufdauer überschritten ist, werden verarbeitet und ihre zugeordnete Handlerfunktion aufgerufen oder die zugehörige Aktivität abgeschlossen.

8.4.5 Laufzeitinformationen

Wenn die Workflow-Engine eine Aktivität ausführt, die Fristmetadaten enthält, erstellt sie Datum und Uhrzeit der Frist wie folgt:

- Wenn die Dauer in Sekunden angegeben wurde, dann gilt für die Berechnung Folgendes: Aktuelles Datum + Sekunden (wie in den Metadaten definiert) = Frist (Datum/Zeit).
- Wenn die Dauer als Workflowdatenobjekt-Attribut definiert wurde, dann gilt für die Berechnung Folgendes: Aktuelles Datum + Wert (wie im Workflowdatenobjekt-Attribut definiert) = Frist (Datum/Zeit).

Fristen, die abgelaufen sind, werden durch das Aufrufen des Batch-Jobs `ScanTaskDeadlines` verarbeitet. Dieser Batch-Job ruft wiederum die weiter oben beschriebene Fristscanner-API des Workflows auf, die eine Liste aller abgelaufenen Fristen abruft und diese verarbeitet. Wenn für die Frist eine Fristhandlermethode angegeben wurde, werden die Werte der Workflowdatenobjekt-Attribute, die in den Parameterzuordnungen definiert sind, als Eingabeparameter für die Fristhandlermethode bereitgestellt, die dann aufgerufen wird. Wenn das Flag für die vollständige Aktivität auf "true" gesetzt ist, dann wird die zugehörige Aktivität abgeschlossen. Alle möglichweise angegebenen Fristausgabezuordnungen (Fristablauf und Ablaufzeit der Frist) werden beibehalten. Die Attribute des Workflowdatenobjekts `Context_Deadline` werden während dieser Verarbeitung auch beibehalten, damit sie in Übergängen verwendet werden können, die von der Aktivität mit der Frist ausgehen.

8.4.6 Beschreibung von Workflowdatenobjekten des Typs 'Context'

Das Workflowdatenobjekt 'Context_Deadline' kann in Datenelement- oder Funktionsbedingungen (siehe Kapitel 16, „Bedingungen”, auf Seite 113) für einen Übergang von einer Aktivität mit einem fristbasierten Event-Wait verwendet werden. Zu den verfügbaren Attributen des Workflowdatenobjekts 'Context_Deadline' zählen:

Context_Deadline.durationExpired
Stellt einen booleschen Wert dar, der anzeigt, ob die der Aktivität zugeordnete Frist abgelaufen ist.

Context_Deadline.expiryTime
Ein Attribut, das den Zeitpunkt (Datum/Uhrzeit) angibt, an dem die Fristdauer abläuft.

8.5 Ausgabezuordnungen

Das ausgelöste Ereignis enthält einige Informationen, die möglicherweise wieder der Workflow-Engine zugeordnet werden sollten. Das Ereignis verfügt sowohl über primäre als auch über sekundäre Ereignisdaten. Mit den primären Ereignisdaten wird vor allem das Ereignis abgeglichen; es macht daher wenig Sinn, diese Daten wieder dem Prozess zuzuführen. Die sekundären Ereignisdaten hingegen sind der

Workflow-Engine möglicherweise nicht bekannt und könnten ihr demnach zugeordnet werden. Da zudem eine Event-Wait-Aktivität auf eine beliebige Anzahl von Ereignissen warten kann, kann das eigentlich ausgelöste Ereignis von Interesse sein und demnach auch der Workflow-Engine zugeordnet werden. Auch der Cúram-Benutzer, der das Ereignis auslöst, kann interessant sein und auch wieder der Workflow-Engine zugeordnet werden.

Wenn eine Aktivitätsinstanz warten sollte, bis alle ihre zugehörigen Event-Waits zugewiesen wurden, werden alle Ereignisausgabezuordnungen, die für die Aktivitätszuordnung vorhanden sind, bei jedem Auslösen eines Ereignisses verarbeitet, das einem der Event-Waits zugeordnet werden kann.

8.5.1 Metadaten

```
<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
Enthält die Daten, die optional der Workflow-Engine von dem Ereignis zugeordnet werden können, das ausgelöst wurde.

event-type
Enthält das Geschäftsereignis, das ausgelöst wurde und auf das die Aktivitätsinstanz gewartet hat.

output-data
Enthält die sekundären Ereignisdaten, die mit der Workflow-Engine verknüpft werden sollen.

raised-by
Enthält den Benutzernamen des Cúram-Benutzers, der dafür verantwortlich ist, dass das Ereignis ausgelöst wurde.

time-raised
Enthält das Datum und die Uhrzeit der Ereignisauslösung.

8.5.2 Validierungen

- Falls angegeben, muss die Ereignisausgabezuordnung für den Ereignistyp ein gültiges Workflowdatenobjekt-Attribut des Typs 'STRING' sein.
- Falls angegeben, muss die Ereignisausgabezuordnung für den Namen des auslösenden Benutzers ein gültiges Workflowdatenobjekt-Attribut des Typs 'STRING' sein.
- Falls angegeben, muss die Ereignisausgabezuordnung für die Ausgabendaten ein gültiges Workflowdatenobjekt-Attribut des Typs 'STRING' sein.
- Falls angegeben, muss die Ausgabezuordnung für die Auslösezeit ein gültiges Workflowdatenobjekt-Attribut des Typs 'DATETIME' sein.

8.5.3 Laufzeitinformationen

Wenn ein Ereignis in der Anwendung ausgelöst wird, auf das eine Aktivitätsinstanz wartet, werden alle Workflowdatenobjekt-Attribute, die in den für das Event-Wait definierten Ereignisausgabezuordnungen enthalten sind, mit den entsprechenden Daten aus dem Ereignis aufgefüllt und beibehalten.

8.5.4 Beschreibung von Workflowdatenobjekten des Typs 'Context'

Das Workflowdatenobjekt 'Context_Event' kann in Datenelement- oder Funktionsbedingungen (siehe Kapitel 16, „Bedingungen", auf Seite 113) für einen Übergang von einer Aktivität mit einem Event-Wait verwendet werden. Zu den verfügbaren Attributen des Workflowdatenobjekts 'Context_Event' zählen:

Context_Event.raisedByUserName
Der Cúram-Benutzername des Benutzers, der das Ereignis ausgelöst hat.

Context_Event.timeRaised
Der Zeitpunkt, zu dem das Ereignis ausgelöst wurde.

Context_Event.fullyQualifiedEventType
Der vollständig qualifizierte Name (sowohl Ereignisklasse als auch Ereignistyp) des Geschäftsereignisses, das ausgelöst wurde.

Context_Event.outputData
Die sekundären Ereignisdaten, die dem ausgelösten Ereignis zugeordnet sind.

8.6 Erinnerungen

Eine Erinnerung kann für alle Fristen angegeben werden, die einer manuellen Aktivität, einer Entscheidungsaktivität, einer Even-Wait-Aktivität, einer parallelen manuellen Aktivität oder einer parallelen Entscheidungsaktivität zugeordnet sind. Es können beliebig viele Erinnerungen angegeben werden. Erinnerungen verwenden die Benachrichtigungsmetadaten, die im Kapitel zu Aktivitätsbenachrichtigungen beschrieben sind (siehe Kapitel 14, „Aktivitätsbenachrichtigungen", auf Seite 101). Dies bedeutet, dass typische Merkmale einer Benachrichtigung, wie Betreff, Haupttext, Zuteilungsstrategie und Aktionen, für eine Erinnerung angegeben werden können.

8.6.1 Metadaten

```
<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
Dieser Tag ist optional und umfasst alle Tags des Typs 'reminder' für die Frist.

reminder
Enthält alle Erinnerungsmetadaten für die Frist, einschließlich der zugehörigen Benachrichtigungsmetadaten.

delivery-offset
Bezieht sich auf einen Wert aus der Codetabelle `ReminderDeliveryOffset`, der den Versatz für die Tags 'seconds' oder 'mapped-delivery-time' anzeigt. Im Fall einer Frist handelt es sich hierbei um den Versatz von der Ablaufzeit der Frist. Dies ist der derzeit einzig unterstützte Versatz.

delivery-time
Enthält entweder den Wert für den Tag 'seconds' oder den Tag 'mapped-delivery-time', je nachdem, welcher angegeben wurde.

seconds
Dieser Tag gibt an, wie viele Sekunden vor der Ablaufzeit der Frist die Erinnerung gesendet werden soll.

mapped-delivery-time
Dieser Tag stellt ein Workflowdatenobjekt dar, dass angibt, wie viele Sekunden vor dem Ablauf der Frist die Erinnerung gesendet wird.

8.6.2 Validierungen

- Es kann keine Erinnerung erstellt werden, wenn eine Frist nicht der entsprechenden Aktivität zugeordnet wurde. Eine Erinnerung kann auch nicht erstellt werden, wenn zwar eine Frist vorhanden ist, der Fristhandler aber nicht aktiviert wurde oder die Kennung für die vollständige Aktivität auf "false" gesetzt ist.
- Jede Erinnerung verfügt über eine Kennung. Diese Kennung muss für die Frist eindeutig sein, auf deren Grundlage die Zuordnung erfolgt.
- Für eine Erinnerung muss entweder der Tag 'mapped-delivery-time' oder der Tag 'seconds' angegeben sein.
- Ist der Tag 'seconds' angegeben, muss der zugehörige Wert vor der Ablaufzeit der Frist liegen.
- Das vom Tag 'mapped-delivery-time' referenzierte Attribut des Workflowdatenobjekts muss den Typ 'INTEGER' aufweisen.
- Alle vorhandenen Validierungen für Aktivitätsbenachrichtigungen (siehe Kapitel 14, „Aktivitätsbenachrichtigungen”, auf Seite 101) gelten für die mit Erinnerungen verknüpften Benachrichtigungsmetadaten.

8.6.3 Code

Die API-Funktion `DeadlineScanner.scanDeadlines()` des Workflowfristenscanners enthält einen Aufruf für die Funktion `deliverReminders()`, die alle Erinnerungen, die ihre Bereitstellungszeit erreicht haben, verarbeitet und sendet.

8.6.4 Laufzeitinformationen

Wenn eine Aktivität, die Erinnerungen enthält, ausgeführt wurde, werden die Erinnerungen in der Entität `Reminders` gespeichert. Der Zeitrahmen, in dem eine Erinnerung gesendet werden soll, wird wie folgt berechnet:

- Die Bereitstellungsdauer für die Erinnerung wird in Sekunden abgerufen. Dies kann direkt in Sekunden oder in Form eines Workflowdatenobjekt-Attributs angegeben werden.
- Die Dauer für die der Erinnerung zugeordneten Frist wird in Sekunden abgerufen. Dies kann direkt in Sekunden oder in Form eines Workflowdatenobjekt-Attributs angegeben werden.
- Wenn die Bereitstellungsdauer für die Erinnerung als positive Zahl angegeben wird und diese Zahl kleiner ist als die Fristdauer (Bereitstellungszeiten für Erinnerungen dürfen aus offensichtlichen Gründen nicht nach dem Fristdatum liegen), dann wird die Zeit für die Bereitstellung der Erinnerungsbenachrichtigung als die Fristdauer (die Bereitstellungsdauer der Erinnerung) berechnet. Diese Dauer in

Sekunden wird dann in einen Wert für Datum/Uhrzeit konvertiert und anschließend dem Zeitpunkt (Datum/Uhrzeit) hinzugefügt, auf dessen Grundlage die Erinnerung erstellt wird. Dieser Wert wird dann im Erinnerungsdatensatz als der Zeitpunkt gespeichert, zu dem die Erinnerungsbenachrichtigung gesendet werden soll.

Erinnerungen, die für Fristen konfiguriert wurden, werden durch Aufrufen des Batch-Jobs `ScanTaskDeadlines` verarbeitet und gesendet. Dieser Batch-Job ruft die Funktion `DeadlineScanner.scanDeadlines()` auf, die einen Scan nach fälligen Erinnerungen durchführt und die entsprechenden Erinnerungsbenachrichtigungen (mithilfe der Zuteilungsstrategie für Erinnerungsbenachrichtigungen, um die Benutzer zu ermitteln, an die die Benachrichtigungen gesendet werden sollen) sendet. Die Erinnerungen, die gesendet wurden, werden aus der Entität `Reminders` entfernt, um sicherzustellen, dass sie nicht noch einmal gesendet werden. Wenn die Aktivität abgeschlossen wird, werden alle Erinnerungen, die für die Aktivität konfiguriert, aber nicht gesendet wurden, entfernt.

Kapitel 9. Manuell

9.1 Voraussetzungen

- Die Basisdetails, die für alle von Cúram Workflow unterstützten Aktivitätstypen gelten, sind in Kapitel 6, „Basisaktivität", auf Seite 31 beschrieben und gelten für die nachfolgend beschriebene manuelle Aktivität.

9.2 Übersicht

In allen automatisierten Geschäftsprozesses ist ein menschliches Eingreifen unabdingbar. Es müssen Entscheidungen getroffen, zusätzliche Daten bereitgestellt oder ganz alltägliche Aufgaben wie das Anrufen eines Kunden erledigt werden. In Cúram Workflow werden solche Schritte in einem Prozess mithilfe von manuellen Aktivitäten modelliert. Eine manuelle Aktivität gibt an, an welcher Stelle im Geschäftsprozess ein menschliches Eingreifen erforderlich ist. Sie legt zudem die Informationen fest, die der Benutzer im Rahmen der Benachrichtigung über eine auszuführende Aufgabe erhält, und gibt die Auswahl der Bearbeiter an, denen die Arbeit zugewiesen wird.

9.3 Aufgabendetails

Um einen Benutzer zu benachrichtigen, dass er im Rahmen eines automatisierten Geschäftsprozesses eine bestimmte Arbeit leisten muss, wird ihm eine entsprechende Aufgabe zugewiesen. Eine Aufgabe ist eine Nachricht, die im Posteingang des Benutzers angezeigt wird. Dort wird angegeben, welche Arbeit der Benutzer zu verrichten hat. Einer Aufgabe kann zudem eine Liste mit Aktionen zugeordnet sein. Aktionen sind Links zu Cúram-Anwendungsseiten, auf denen die für die Aufgabe erforderliche Arbeit durchgeführt werden kann.

© Copyright IBM Corp. 2012, 2013

9.3.1 Metadaten

```
<manual-activity id="1">
      ...
      <task>
        <message>
          <message-text>
            <localized-text>
              <locale language="en">The following
                case %1n for %1s must be approved</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">
                    Case Home Page for case: %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">
                    Confirm Person Details for
                      person: %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 Enthält alle Details einer Aufgabe, einschließlich die Nachricht und die Details der zugehörigen Aktionen. Die mit einer Aufgabe verknüpften Metadaten sind nachfolgend aufgeführt.

message
Enthält die Details der parametrisierten Nachricht. Beim Ausführen einer manuellen Aktivität wird eine Aufgabe erstellt. Wenn ein Benutzer sich seine Aufgaben im Posteingang ansieht, gibt diese Nachricht den Betreff der entsprechenden Aufgabe an.

message-text
Enthält die Details des Nachrichtentextes. Der Betrefftext kann ersetzbare Zeichenfolgen (%k) enthalten, die durch die verknüpften Textparameter ersetzt werden. Ein Textparameter ist eine Zuordnung zu einem Workflowdatenobjekt-Attribut. Parameter 'k' in der Liste ersetzt '%k' in der Textzeichenfolge, wobei 'k' die Reihenfolge der Parameter in der Liste ist. '%k' kann innerhalb der Zeichenfolge wiederholt werden. Daher darf jedes Workflowdatenobjekt-Attribut nur einmal zugeordnet werden. Optional kann auch ein Format für die ersetzbare Zeichenfolge angegeben werden, indem nach der Zeichenfolge ein Buchstabe eingefügt wird, z. B. "%1d", wobei "d" den Wert als Datum kennzeichnet.

Tabelle 3. Datenkonvertierung für Betrefftext

Formatierungsbuchstabe	Formatieren als
s	Zeichenfolge
n	Zahl
d	Datum
z	Datum/Uhrzeit
t	Uhrzeit

localized-text
Enthält die Details des lokalisierbaren Aufgabennachrichtentextes. Weitere Informationen zu lokalisierbarem Text und den zugehörigen Metadaten finden Sie im Abschnitt 6.2.1, „Lokalisierter Text", auf Seite 32.

message-parameters
Einer Aufgabennachricht können Parameter zugeordnet sein. Dieser Tag enthält die Details der Workflowdatenobjekt-Attributparameter, die zum Ersetzen der Platzhalter im zugehörigen Text verwendet werden. Informationen zu Workflowdatenobjekten und Workflowdatenobjekt-Attributen finden Sie in Kapitel 4, „Workflowdatenobjekte", auf Seite 17.

actions
Enthält die Details aller Aktionen, die der Aufgabe der manuellen Aktivität zugeordnet sind. Diese Aktionen sind Links zu Cúram-Anwendungsseiten, auf denen die für die Aufgabe erforderliche Arbeit durchgeführt werden kann.

action Enthält die Definition eines Hyperlinks zu einer Cúram-Seite, auf der eine Aufgabe ausgeführt werden kann. Die der Aufgabenaktion zugeordneten Felder sind nachfolgend beschrieben:

page-id
Gibt die Kennung der Cúram-Zielseite an, auf der ein Benutzer die erforderliche Aktion ausführen kann.

principal-action
Aktionen müssen als primäre oder als sekundäre Aktionen definiert werden. Primäre Aktionen enthalten für gewöhnlich die Links zu den Cúram-Seiten, auf denen ein Benutzer die tatsächlich erforderliche Arbeit ausführen kann. Sekundäre Aktionen enthalten Links zu unterstützenden Informationen, auf die der Benutzer beim Durchführen seiner zugewiesenen Aufgabe zurückgreifen kann.

open-modal
Die mit einer Aufgabenaktion verknüpften Seiten können so angegeben werden, dass sie

in einem modalen Dialogfeld geöffnet werden. Wenn dieser Indikator auf "true" gesetzt ist, wird die vom Aktionslink angegebene Seite in einem modalen Dialogfeld geöffnet. Ist er auf "no" (Standardeinstellung) gesetzt, entscheidet die Clientinfrastruktur in gleicher Art und Weise wie für jeden anderen Link in der Anwendung, wie der Link geöffnet wird (d. h., wenn die Seite Teil einer Registerkartenkonfiguration ist, wird die entsprechende Registerkarte geöffnet; wenn nicht, wird ganz einfach die Startseite des Aktionslinks im Inhaltsbereich der aktuellen Registerkarte mit der Seite ersetzt).

message
Enthält die Details der parametrisierten Nachricht, die der durchzuführenden Aktion zugeordnet ist, einschließlich des Nachrichtentextes und der optionalen Parameter, die dem Text zugeordnet sein können.

link-parameter
Die Links zu den Cúram-Seiten, auf denen die tatsächliche Arbeit für die Aufgabe durchgeführt wird, müssen eine Seitenkennung (siehe weiter oben) und optionale Seitenparameter enthalten. Diese Seitenparameter werden durch die nachfolgenden Metadaten beschrieben und stellen ein Name/Wert-Paar dar, wobei das Attribut 'name' der Name eines Linkparameters (der Name des Seitenparameters auf der zugehörigen Cúram-Clientseite) ist und der Wert von einem Workflowdatenobjekt-Attribut bereitgestellt wird. Das dem Linkparameter zugeordnete Feld ist nachfolgend beschrieben:

name Der Name des Linkparameters.

multiple-occurring-action
Gibt an, dass diese Aktion eine mehrfach auftretende Aktion darstellt. Wenn also diese Metadaten für eine Aktion angegeben werden, erstellt die Workflow-Engine beim Ausführen der Aktivität einen Datensatz der Aktion für jedes Element im Listen-Workflowdatenobjekt, das als die mehrfach auftretende Aktion festgelegt wurde.

Wenn eine Aktion als mehrfach auftretende Aktion verwendet wird, muss ein Attribut des verknüpften Listen-Workflowdatenobjekts als Linkparameter für die Aktion verwendet werden.

list-wdo-name
Der Name des Listen-Workflowdatenobjekts, das mit der mehrfach auftretenden Aktion verwendet wird.

wdo-attribute
Der Wert, der im Aktionslinkparameter verwendet wird, wird durch die in diesen Metadaten angegebene Workflowdatenobjekt-Attributzuordnung bereitgestellt.

task-priority
Eine Aufgabe kann optional eine Priorität enthalten. Die Details dazu sind in diesen Metadaten enthalten. Die Priorität einer Aufgabe kann in einem der folgenden Formate angegeben werden:

priority
In diesem Fall wird die Priorität im Prozessdefinitionstool ausgewählt und stammt aus der Codetabelle `TaskPriority`.

mapped-priority
Die Priorität einer manuellen Aufgabe kann mithilfe eines Workflowdatenobjekt-Attributs zugeordnet werden. Der folgende Metadatenausschnitt zeigt beispielhaft, wie dies erreicht werden kann:

```
<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
Stellt ein boolesches Flag dar, das anzeigt, ob die der Aufgabe für die manuelle Aktivität zugeordnete Frist (siehe 8.4, „Frist", auf Seite 52) überschrieben werden darf. Wenn der Wert für dieses Flag auf `true` (der Standardwert ist `false`) gesetzt wird, heißt das, dass das Fristintervall geändert werden kann, nachdem die Aufgabe von der Workflow-Engine erstellt wurde.

allow-task-forwarding
Stellt ein boolesches Flag dar, das anzeigt, ob die Aufgabe, die im Rahmen der Ausführung der zugehörigen manuellen Aktivität erstellt wurde, an einen anderen Benutzer weitergeleitet werden kann. Wenn eine Aufgabe erstellt wird, wird sie einem Sachbearbeiter zur Bearbeitung zugeordnet. Wenn dieses Flag auf `true` (den Standardwert) gesetzt wird, kann der Sachbearbeiter die Aufgabe an einen anderen Bearbeiter weiterleiten, der die erforderliche Arbeit ausführen soll.

administration-sid
Mit diesem Feld kann eine Verwaltungssicherheitskennung für eine manuelle Aufgabe angegeben werden. Dadurch kann ein Benutzer in einer Gruppe, die der angegebenen Sicherheitskennung zugeordnet ist, die Aufgabendetails ändern, obwohl die Aufgabe möglicherweise durch einen anderen Benutzer in der Anwendung reserviert ist.

initial-comment
Hiermit kann eine Zuordnung des Erstkommentars für die manuelle Aktivität angegeben werden. Mithilfe des Wertes des Workflowdatenobjekt-Attributs, der in dieser Zuordnung verwendet wird, kann ein Datensatz in die Tabelle `TaskHistory` eingefügt werden, wenn die zugehörige manuelle Aktivität ausgeführt wird.

9.3.2 Validierungen

- Für eine Aufgabe einer manuellen Aktivität muss ein Betreff definiert werden. Dabei handelt es sich um eine lokalisierbare Zeichenfolge in der Prozessdefinition, es muss jedoch ein Eintrag für die Standardländereinstellung des Servers vorhanden sein.
- Alle Workflowdatenobjekte, die als Betrefftextparameter im Betreff der Aufgabe der manuellen Aktivität verwendet werden, müssen bezüglich der zugehörigen Workflowprozessdefinition gültige Attribute sein.
- Wenn ein indexiertes Element aus einem Listen-Workflowdatenobjekt (d. h. 'PersonDetailsList[Context-_Loop.loopCount]') als Betrefftextparameter verwendet wird, muss das Workflowdatenobjekt ein Listen-Workflowdatenobjekt sein und die Aktivität mit der Zuordnung muss in einer Schleife enthalten sein.
- Wenn das Workflowdatenobjekt `Context_Parallel` als Betrefftextparameter verwendet wird, muss die Aktivität mit der Zuordnung eine manuelle Aktivität des Typs `Parallel` sein.

- Wenn ein indexiertes Element aus dem parallelen Listen-Workflowdatenobjekt als Betrefftextparameter verwendet wird, muss die Aktivität mit der Zuordnung eine parallele Aktivität sein (d. h. 'ParallelListWDO[Context_Parallel.occurrenceCount]'). Das anhand des Workflowdatenobjekts 'Context_Parallel' indexierte Workflowdatenobjekt muss ein Listen-Workflowdatenobjekt für eine parallele Aktivität sein.
- Wenn für die Aufgabe der manuellen Aktivität Aktionen festgelegt wurden, müssen alle Workflowdatenobjekt-Attribute, die als Zuordnungen für Aktionstextparameter verwendet werden, bezüglich der zugehörigen Workflowprozessdefinition gültige Attribute sein.
- Wenn für die Aufgabe der manuellen Aktivität Aktionen festgelegt wurden, müssen alle Workflowdatenobjekt-Attribute, die in den Zuordnungen für Aktionslinkparameter einer manuellen Aktivität verwendet werden, bezüglich der zugehörigen Workflowprozessdefinition gültige Attribute sein.
- Wenn ein indexiertes Element aus einem Listen-Workflowdatenobjekt (d. h. 'PersonDetailsList[Context_Loop.loopCount]') in den Zuordnungen für die Textparameter oder Linkparameter der Aktion verwendet wird, muss das Workflowdatenobjekt ein Listen-Workflowdatenobjekt sein und die Aktivität mit der Zuordnung muss in einer Schleife enthalten sein.
- Wenn das Workflowdatenobjekt `Context_Parallel` in den Zuordnungen für die Textparameter oder Linkparameter der Aktion verwendet wird, muss die Aktivität mit der Zuordnung eine manuelle Aktivität des Typs `Parallel` sein.
- Wenn ein indexiertes Element aus dem parallelen Listen-Workflowdatenobjekt in den Zuordnungen für die Textparameter oder Linkparameter der Aktion verwendet wird, muss die Aktivität mit der Zuordnung eine parallele Aktivität sein (d. h. 'ParallelListWDO[Context_Parallel.occurrenceCount]'). Das anhand des Workflowdatenobjekts 'Context_Parallel' indexierte Workflowdatenobjekt muss ein Listen-Workflowdatenobjekt für eine parallele Aktivität sein.
- Die Anzahl von Platzhaltern, die im Betreff- und im Aktionstext der Aufgabe der manuellen Aktivität verwendet wird, muss der Anzahl der zugeordneten Attribute des Workflowdatenobjekts für alle definierten Ländereinstellungen entsprechen.
- Die Priorität einer manuellen Aufgabe kann entweder mithilfe eines Codetabellenwerts oder einer Attributzuordnung des Workflowdatenobjekts angegeben werden. Beides zusammen ist nicht möglich.
- Wenn für die Aufgabe der manuellen Aktivität eine zugeordnete Priorität angegeben ist, muss das entsprechende Attribut des Workflowdatenobjekts bezüglich der zugehörigen Workflowprozessdefinition ein gültiges Attribut sein. Es muss zudem eine Zeichenfolge sein.
- Wenn für die Aufgabe der manuellen Aktivität eine Zuordnung des Erstkommentars angegeben ist, muss das entsprechende Attribut des Workflowdatenobjekts bezüglich der zugehörigen Workflowprozessdefinition ein gültiges Attribut sein. Es muss zudem eine Zeichenfolge sein.
- Das Workflowdatenobjekt, das für die Verwendung in der mehrfach auftretenden Aktion angegeben wurde, muss bezüglich der zugehörigen Workflowprozessdefinition ein gültiges Workflowdatenobjekt sein. Zudem muss es ein Listen-Workflowdatenobjekt sein.
- Mindestens ein Attribut des Workflowdatenobjekts für die mehrfach auftretende Aktion muss in den Linkparametern verwendet werden, die für die mehrfach auftretende Aktion angegebenen wurden.

9.3.3 Code

Aktionsseiten und Aktionsseitenparameter

Die für die manuelle Aktivitätsaufgabe angegebenen Aktionen sind Links zu Cúram-Anwendungsseiten, auf denen die für die Aufgabe erforderliche Arbeit durchgeführt werden kann. Bei den Seiten, die in den Aufgabenaktionen angegeben sind, muss es sich um gültige Cúram-Seiten handeln, die in der Cúram-Anwendung verfügbar sind. Die auf diesen Seiten festgelegten Parameter müssen den Parametern entsprechen, die als Aktionslinkparameter in den zugehörigen Aufgabenaktionen angegeben sind.

API 'LocalizableStringResolver.TaskStringResolver'

Der Aufgabenbetreff und die zugehörigen Aufgabenaktionsnachrichten werden im Posteingang eines Benutzers angezeigt, um ihn darüber zu informieren, welche Arbeit für die zugehörige Aufgabe durchgeführt werden muss. Die API `LocalizableStringResolver.TaskStringResolver` enthält die Funktionen, die zum Auflösen der Aufgabenbetrefftexte und der Aktionsnachrichten für

die richtige Benutzerländereinstellungen erforderlich sind. Als Teil dieser Funktionen werden auch die Platzhalter mit den zugehörigen Attributwerten des Workflowdatenobjekts ersetzt, die in den entsprechenden Zuordnungen angegeben sind.

API für 'TaskAdmin'
In der Klasse `TaskAdmin` stehen eine Reihe von Funktionen zur Bearbeitung von Aufgaben bereit. Weitere Informationen zu den verfügbaren Funktionen finden Sie in der zugehörigen Javadoc-Spezifikation für die Klasse `TaskAdmin`.

API für 'TaskHistoryAdmin'
Während der Lebensdauer einer Aufgabe werden verschiedene Lebenszyklusereignisse für eine Aufgabe (d. h. wenn eine Aufgabe erstellt wird, wenn eine Aufgabe zugeteilt wird, wenn eine Aufgabe geschlossen wird) in die Tabelle `TaskHistory` geschrieben. Für diese API-Klasse wurden eine Reihe von Suchfunktionen zur Untersuchung dieser Einträge bereitgestellt. Weitere Informationen zu den verfügbaren Funktionen finden Sie in der zugehörigen Javadoc-Spezifikation für die Entität `TaskHistoryAdmin`.

API für 'WorkflowDeadlineAdmin'
In der Klasse `WorkflowDeadlineAdmin` stehen eine Reihe von Funktionen zur Bearbeitung von Workflowfristen bereit. Weitere Informationen zu den verfügbaren Funktionen finden Sie in der zugehörigen Javadoc-Spezifikation für die Klasse `WorkflowDeadlineAdmin`.

9.3.4 Laufzeitinformationen

Beim Ausführen einer manuellen Aktivität durch die Workflow-Engine wird eine Aufgabe erstellt, die einem Bearbeiter zur Bearbeitung zugewiesen wird (siehe 9.4, „Zuteilungsstrategie").

9.3.5 Beschreibung von Workflowdatenobjekten des Typs 'Context'

Mit dem Workflowdatenobjekt 'Context_Task' kann die eindeutige Kennung der Aufgabe, die im Rahmen der Ausführung der zugehörigen manuellen Aktivität erstellt wurde, in den verschiedenen Metadatenzuordnungen verwendet werden, die der manuellen Aktivität zugeordnet sind. Beispiele für solche Zuordnungen sind die Zuordnungen für Ereignisübereinstimmungsdaten (siehe 8.3, „Liste der Ereignisse", auf Seite 49) und Eingabezuordnungen für die Fristfunktion (siehe 8.4, „Frist", auf Seite 52). Das folgende Attribut steht für dieses Workflowdatenobjekt zur Verfügung:

Context_Task.taskID
Das Attribut `taskID` steht für die eindeutige Kennung der Aufgabe, die beim Ausführen der zugehörigen manuellen Aktivität erstellt wird.

9.4 Zuteilungsstrategie

Eine Organisation verfügt normalerweise über viele Mitarbeiter mit unterschiedlichen Verantwortlichkeiten, die eine bestimmte Prozessdefinition bearbeiten können. Um einen bestimmten Mitarbeiter oder eine Gruppe von Mitarbeitern auszuwählen, die eine bestimmte manuelle Aktivität bearbeiten können, wird der entsprechenden Aktivität eine Zuteilungsstrategie zugewiesen. Cúram Workflow unterstützt zurzeit vier Zuteilungsstrategietypen: Funktion, klassische Regeln, Cúram Express-Regeln (CER-Regeln) und Ziel. Wenn eine Zuteilungsstrategie des Typs 'Ziel' ausgewählt ist, wird der Mitarbeiter oder die Gruppe der Mitarbeiter, dem bzw. der die Arbeit zugewiesen werden soll, namentlich benannt. Bei Auswahl der Zuteilungsstrategie des Typs 'Funktion' wird die angegebene Zuteilungsfunktion aufgerufen, wenn die zugehörige Aktivität von der Workflow-Engine ausgeführt wird. Wird die Zuteilungsstrategie 'Klassisch' oder 'CER' ausgewählt, wird beim Ausführen der zugehörigen Aktivität das angegebene Regelwerk ausgeführt.

9.4.1 Voraussetzungen

- Wenn die einer manuellen Strategie zugeordnete Zuteilungsstrategie den Typ `Funktion` aufweist, sind diese Zuteilungsfunktionen Cúram-Geschäftsmethoden mit einer bestimmten Signatur. Die Eingabezu-

ordnungen für die formalen Parameter dieser Methoden und deren zugehörigen Metadaten sind in Kapitel 7, „Automatisch", auf Seite 35 näher beschrieben. Dieses Kapitel sollte daher für eine Beschreibung dieser Zuordnungen hinzugezogen werden.

9.4.2 Metadaten

Wie bereits beschrieben, gibt es vier Typen von Zuteilungsstrategien. Die erforderlichen Metadaten für die einzelnen Typen sind in den folgenden Abschnitten beschrieben.

allocation-strategy
Enthält die Details der für die manuelle Aktivität definierten Zuteilungsstrategie. Die einer Zuteilungsstrategie zugeordneten Felder sind nachfolgend beschrieben:

type Enthält den Typ der Zuteilungsstrategie. Cúram Workflow unterstützt zurzeit vier Zuteilungsstrategietypen: Funktion, klassische Regeln, CER-Regeln und Ziel.

identifier
Gibt die Kennung der Zuteilungsstrategie an. Für eine Zuteilungsstrategie des Typs 'Funktion' gibt diese Kennung den vollständig qualifizierten Namen der verwendeten Zuteilungsfunktion an. Für eine Zuteilungsstrategie des Typs 'Regel' oder 'Curam-Express' gibt diese Kennung die Kennung des verwendeten Regelwerks an. Wenn eine Zuteilungsstrategie des Typs 'Ziel' ausgewählt ist, gibt diese Kennung die Kennung des verwendeten Zuteilungsziels an.

9.4.2.1 Funktionszuteilungsstrategie

```
<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

Enthält die Details zu den Eingabezuordnungen für die Formalparameter der angegebenen Zuteilungsfunktion. Bei Zuteilungsfunktionen handelt es sich um Cúram-Geschäftsmethoden (ähnlich wie die, die für automatische Aktivitäten festgelegt werden), die eine bestimmte Rückgabesignatur aufweisen (Zuteilungsfunktionen müssen den Rückgabetyp `curam.util.workflow.struct.AllocationTargetList` haben). Daher entsprechen die für diese Zuordnungen verwendeten Metadaten denen für die Eingabezuordnungen für die GPO-Methoden (GPO, Geschäftsprozessobjekt), die automatischen Aktivitäten zugeordnet sind. Weitere Informationen zu diesen Metadaten und deren Bedeutung finden Sie im Abschnitt 7.4, „Eingabezuordnungen", auf Seite 36 des Kapitels zu automatischen Aktivitäten.

9.4.2.2 CER-Regelzuordnung

```
<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

Enthält die Details aller Zuordnungen für das *Regelwerk*, das in der Zuteilungskennung angegeben ist. Es ist nicht erforderlich, alle im Regelwerk angegebenen Attribute des Regeldatenobjekts zuzuordnen (es können Zuordnungen für eine Untergruppe der Attribute erstellt werden).

rdo-mapping

Enthält die Details einer Zuordnung zwischen einem im Zuteilungsregelwerk angegebenen Attribut des Regeldatenobjekts und seinem zugehörigen Attribut des Workflowdatenobjekts. Die folgenden Metadaten stellen eine gültige Zuordnung dar:

source-attribute

Enthält die Details des Quellenattributs in der Zuordnung (d. h. aus dem die Daten zur Laufzeit bereitgestellt werden). Ein Quellenattribut besteht aus einem Workflowdatenobjektnamen und seinem zugehörigen Attributnamen (siehe Kapitel 4, „Workflowdatenobjekte", auf Seite 17).

target-attribute

Enthält die Details des Zielattributs in der Zuordnung (d. h. worin die Daten zur Laufzeit zugeordnet werden). Ein Zielattribut besteht aus einem Regeldatenobjektnamen und seinem zugehörigen Attributnamen.

9.4.2.3 CER-Regelzuordnung

```
<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="concernRoleID" />
            <target-attribute cer-class-name="SampleAllocationClass"
                              name="concernRoleID" />
          </cer-class-mapping>
        </cer-set-mappings>
      </allocation-strategy>
      <event-wait>
        ...
      </event-wait>
      ...
    </manual-activity>
```

cer-set-mappings
Enthält die Details aller Zuordnungen für das *CER-Regelwerk*, das in der Zuteilungskennung angegeben ist. Der primäre Klassenparameter muss auf eine Regelklasse verweisen, die das Attribut *targets* für diese Zuordnungsstrategie enthält. Es wird empfohlen, dass Zuordnungen für alle angegebenen Attribute in der ausgewählten Regelklasse erstellt werden.

cer-class-mapping
Enthält die Details einer Zuordnung zwischen einem im CER-Rregelwerk angegebenen Attribut des Regelklassenattributs und seinem zugehörigen Attribut des Workflowdatenobjekts. Die folgenden Metadaten stellen eine gültige Zuordnung dar:

source-attribute
Enthält die Details des Quellenattributs in der Zuordnung (d. h. aus dem die Daten zur Laufzeit bereitgestellt werden). Ein Quellenattribut besteht aus einem Workflowdatenobjektnamen und seinem zugehörigen Attributnamen (siehe Kapitel 4, „Workflowdatenobjekte", auf Seite 17).

target-attribute
Enthält die Details des Zielattributs in der Zuordnung (d. h. worin die Daten zur Laufzeit zugeordnet werden). Ein Zielattribut besteht aus einem CER-Klassennamen und seinem zugehörigen Attributnamen.

9.4.2.4 Zielzuteilungsstrategie

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

Für die Beschreibung einer Zuteilungsstrategie des Typs *Ziel* sind keine weiteren Metadaten erforderlich. Wie bereits erwähnt, ist die Kennung in diesem Fall die Kennung des Zuteilungsziels, das den Bearbeiter bzw. die Gruppe von Bearbeitern angibt, dem oder der die Aufgabe zugewiesen wird.

9.4.3 Validierungen

- Für eine manuelle Aufgabe muss eine Zuteilungsstrategie definiert werden.
- Wenn die Zuteilungsstrategie den Typ 'Funktion' aufweist, muss die angegebene Funktion eine gültige Funktion im Klassenpfad der Cúram-Anwendung sein.
- Wenn die Zuteilungsstrategie den Typ 'Funktion' aufweist, muss der Rückgabetyp der Funktion `curam.util.workflow.struct.AllocationTargetList` lauten.
- Handelt es sich um eine Zuteilungsstrategie des Typs 'Funktion', müssen alle zugeordneten Eingabeparameter der angegebenen Funktion gültige Attribute des Workflowdatenobjekts sein und der Attributtyp des Workflowdatenobjekts muss dem Typ des Eingabeparameterfelds entsprechen.
- Wenn die Zuteilungsstrategie den Typ 'Funktion' aufweist und ein indexiertes Element aus einem Listen-Workflowdatenobjekt als Eingabezuordnung verwendet wird, muss das Workflowdatenobjekt ein Listen-Workflowdatenobjekt sein und die Aktivität mit der Zuordnung muss in einer Schleife enthalten sein.
- Wenn die Zuteilungsstrategie den Typ 'Regel' aufweist, muss das angegebene Regelwerk gültig sein.
- Wenn die Zuteilungsstrategie den Typ 'Regel' aufweist, müssen alle in den Zuordnungen angegebenen Quellenattribute bezüglich der zugehörigen Workflowprozessdefinition gültige Attribute des Workflowdatenobjekts sein. Alle Zielattribute müssen bezüglich des zugehörigen Regelwerks gültige Attribute des Regeldatenobjekts sein. Der Typ des Workflowdatenobjekt-Attributs, das als Quellenattribut angegeben ist, muss dem Typ des Regeldatenobjekt-Attributs entsprechen, das als Zielattribut in der Zuordnung angegeben ist.
- Es sind keine doppelten Zielattributzuordnungen zulässig. Anders gesagt kann ein Attribut des Regeldatenobjekts nur einmal in einer Liste mit Regelwerkzuordnungen angegeben werden.
- Wenn ein indexiertes Element aus einem Listen-Workflowdatenobjekt (d. h. 'PersonDetailsList[Context_Loop.loopCount]') in den Zuordnungen für die Zuteilungsstrategien 'Regel' und 'Funktion' verwendet wird, muss das Workflowdatenobjekt ein Listen-Workflowdatenobjekt sein und die Aktivität mit der Zuordnung muss in einer Schleife enthalten sein.
- Wenn das Workflowdatenobjekt `Context_Parallel` in den Zuordnungen für die Zuteilungsstrategien 'Regel' und 'Funktion' verwendet wird, muss die Aktivität mit der Zuordnung eine Aktivität des Typs `Parallel` sein.
- Wenn ein indexiertes Element aus dem parallelen Listen-Workflowdatenobjekt in den Zuordnungen für die Zuteilungsstrategien 'Regel' und 'Funktion' verwendet wird, muss die Aktivität mit der Zuordnung eine parallele Aktivität sein (d. h. 'ParallelListWDO[Context_Parallel.occurrenceCount]'). Das anhand des Workflowdatenobjekts 'Context_Parallel' indexierte Workflowdatenobjekt muss ein Listen-Workflowdatenobjekt für eine parallele Aktivität sein.

9.4.4 Code

Wie bereits zuvor erläutert muss jede als eine Zuteilungsfunktion angegebene GPO-Methode eine Struktur des Typs `curam.util.workflow.struct.AllocationTargetList` zurückgeben.

Ähnlich wie bei mit automatischen Aktivitäten verknüpften Geschäftsmethoden führt das Fehlschlagen der Zuteilungsstrategie beim Ausführen einer manuellen Aktivität dazu, dass die Strategie zur Handhabung von Workflow-Fehlern aufgerufen wird. Dies kann beispielsweise dazu führen, dass mehrmals versucht wird, die der fehlgeschlagenen Methode zugeordnete Aktivität erneut auszuführen. Basierend auf dieser Tatsache sollten für die Zuteilungsfunktionen, die den Zuteilungsstrategien von manuellen Aktivitäten oder Entscheidungsaktivitäten zugeordnet sind, möglichst keine Ausnahmen auftreten, es sei denn, es liegt eine nicht wiederherstellbare Situation vor.

Die Anwendung muss die Rückrufschnittstelle `curam.util.workflow.impl.WorkResolver` implementieren, um festzulegen, wie Aufgaben in der Anwendung zugeteilt werden. Die Anwendungseigenschaft 'curam-.custom.workflow.workresolver' muss sich auf die Implementierungsklasse `curam.util.workflow.impl.WorkResolver` in der Anwendung beziehen, da die Workflow-Engine diese Eigenschaft zum Ermitteln der richtigen Funktion zum Zuteilen der Aufgabe verwendet.

Die Klasse `curam.util.workflow.impl.WorkResolver` verfügt über die überladene Methode `resolveWork`, da die verschiedenen Zuteilungsstrategietypen die Zuteilungsziele in unterschiedlichen Formaten zurückgeben. Dies ist jedoch ein Aspekt der Implementierung, der für einen Entwickler von angepassten geschäftlichen Resolver-Klassen nicht von Bedeutung sein sollte, da die Geschäftsverarbeitung für alle Versionen der Methode gleich sein sollte.

```
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);

            ...
      }
```

Um diesem Problem entgegenzuwirken, stellt die Klasse `curam.core.sl.impl.DefaultWorkResolverAdapter` einen weitaus effizienteren Mechanismus zur Implementierung eines geschäftlichen Resolvers bereit. Diese Klasse implementiert die unterschiedlichen Methoden und konvertiert deren Eingabeparameter in Zuteilungsziellisten. Dadurch können Entwickler von Logiken für den angepassten Resolver diese Klasse erweitern und eine Methode implementieren, die unabhängig von der Quelle der Zuteilungsziele aufgerufen wird.

```
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);

            ...
      }
```

Neben dieser Adapterklasse ist im Lieferumfang dieser Anwendung auch eine sofort einsatzfähige Implementierung für einen geschäftliche Resolver enthalten. Die Klasse heißt `curam.core.sl.impl.DefaultWorkResolver` und dient ebenfalls als Beispiel zur Erweiterung des Adapters.

9.4.5 Laufzeitinformationen

Beim Ausführen einer manuellen Aktivität verarbeitet die Workflow-Engine die in den Metadaten definierte Zuteilungsstrategie, um die Liste der Zuteilungsziele für diese Aufgabe abzurufen. Wenn die Zuteilungsstrategie den Typ 'Funktion' aufweist, verarbeitet die Workflow-Engine die für die zugehörige Zuteilungsfunktion definierten Eingabezuordnungen und ruft diese Funktion auf, um die Liste der Zuteilungsziele abzurufen. Ist die Zuteilungsstrategie vom Typ 'Regel', verarbeitet die Workflow-Engine die Zuordnungen für das angegebene Regelwerk und ruft die Regel-Engine zum Ausführen des Regelwerks aus, mit dem die Liste der Zuteilungsziele abgerufen werden kann. Wenn die Zuteilungsstrategie den Typ 'Ziel' aufweist, ist das Zuteilungsziel ganz einfach das in den Metadaten angegebene Ziel und es ist keine weitere Verarbeitung notwendig.

Wie in den Metadaten für einen Workflowprozess beschrieben (siehe Kapitel 3, „Metadaten der Prozessdefinition”, auf Seite 13), kann eine Fehlerzuteilungsstrategie für einen Prozess festgelegt werden. Diese Strategie wird verarbeitet und verwendet, wenn durch den Aufruf der mit der Aufgabe verknüpften Zuteilungsstrategie keine Zuteilungsziele zurückgegeben werden.

Die Workflow-Engine legt dann mithilfe der Eigenschaft 'curam.custom.workflow.workresolver' die Implementierung der Funktion fest, die zum Zuordnen von Aufgaben in der Anwendung verwendet wird. Diese Funktion wird dann von der Workflow-Engine aufgerufen. Die Workflow-Engine leitet die Liste der durch die Zuteilungsstrategie bestimmten Zuteilungsziele sowie Details der zuzuordnenden Aufgabe an die Funktion weiter.

Nachdem der geschäftliche Resolver für die Aufgabe aufgerufen wurde, ruft die Workflow-Engine die Methode `checkTaskAssignment` in der Klasse `curam.core.sl.impl.TaskAssignmentChecker` auf. Diese Funktion überprüft den Zuweisungsstatus der Aufgabe, um sicherzustellen, dass sie mindestens einem Benutzer, einem Organisationsobjekt (Organisationseinheit, Position oder Job) oder einem Gruppenpostfach zugeordnet wurde. Wenn die Aufgabe nicht zugewiesen wurde, wird die Anwendungseigenschaft 'curam.workflow.defaultworkqueue' untersucht, um herauszufinden, was als Standard-Gruppenpostfach für den Workflow angegeben ist. Die Aufgabe wird dann diesem Gruppenpostfach zugewiesen.

Wenn die Aufgabe einem Benutzer zugewiesen und nur ein Benutzer nach der Arbeit aufgelöst wurde, überprüft das System den Wert der Anwendungseigenschaft 'curam.workflow.automaticallyaddtasktousertasks'. Dieses Flag steuert, ob das System die verarbeitete Aufgabe automatisch der Liste der Aufgaben dieses Benutzers hinzufügt, damit sie bearbeitet werden kann. Der Standardwert für die Eigenschaft ist `NO`. Wenn der Wert jedoch auf `YES` festgelegt wurde, fügt das System diese Aufgabe automatisch in die Liste 'Meine Aufgaben' des Benutzers in dessen Posteingang ein, damit er sie verarbeiten kann.

9.4.6 Beschreibung von Workflowdatenobjekten des Typs 'Context'

Das Workflowdatenobjekt 'Context_Task' ist sowohl für Zuordnungen für Zuteilungsfunktionen als auch für Zuordnungen für Zuteilungsregeln verfügbar. Diese Workflowdatenobjekt und seine Attribute wurden bereits weiter oben beschrieben (siehe 9.3.5, „Beschreibung von Workflowdatenobjekten des Typs 'Context'", auf Seite 67).

9.5 Geschäftsobjektzuordnungen

Manuelle Aktivitäten und der Workflow im Allgemeinen führen Operationen für Entitäten aus der Anwendung aus. Aus diesem Grund kann es hilfreich sein, den Entitäten eine entsprechende Aufgabe für den Prozess zuzuordnen. Geschäftsobjektzuordnungen stellen Verknüpfungen zwischen einer Aufgabe und allen Anwendungsentitäten bereit, die für den Prozess relevant sind. Die wichtigsten Beispiele in Cúram in diesem Zusammenhang sind die Entitäten 'Case' und 'Concern'.

9.5.1 Metadaten

```
<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
Enthält die Details aller Geschäftsobjektzuordnungen, die für die manuelle Aktivität angegeben wurden.

biz-object-association
Enthält die Details einer Geschäftsobjektzuordnung, die für diese manuelle Aktivität angegeben wurde. Dazu zählen der Geschäftsobjekttyp und die Attributzuordnung des Workflowdatenobjekts, die mit diesem Typ verknüpft ist. Diese Attributzuordnung gibt die eindeutige Kennung des Geschäftsobjekts in der Zuordnung an (d. h., für eine Geschäftsobjektzuordnung des Typs 'Case' wäre dies die eindeutige Kennung des Falls, der mit der Aufgabe verknüpft ist).

biz-object-type
Enthält die Details des tatsächlichen Geschäftsobjekttyps für die Geschäftsobjektzuordnung für die manuelle Aktivität. Der Geschäftsobjekttyp muss im Prozessdefinitionstool ausgewählt werden und stammt aus der Codetabelle `BusinessObjectType`.

9.5.2 Validierungen

- Der angegebene Geschäftsobjekttyp muss ein gültiger Codetabellencode aus der Codetabelle `BusinessObjectType` sein.

- Das Attribut des Workflowdatenobjekts, das dem Geschäftsobjekttyp einer Geschäftsobjektzuordnung einer manuellen Aktivität zugeordnet ist, muss ein gültiges Attribut sein. Dieser Attributtyp muss dem Typ 'LONG' zugewiesen werden können, da dieser eine Zuordnung zu einer eindeutigen Kennung (beispielsweise einer Fallkennung oder einer Beteiligtenkennung) darstellt.
- Wenn ein indexiertes Element aus einem Listen-Workflowdatenobjekt (d. h. 'PersonDetailsList[Context-_Loop.loopCount]') in einer Geschäftsobjektzuordnung verwendet wird, muss das Workflowdatenobjekt ein Listen-Workflowdatenobjekt sein und die Aktivität mit der Zuordnung muss in einer Schleife enthalten sein.
- Wenn das Workflowdatenobjekt `Context_Parallel` in einer Geschäftsobjektzuordnung verwendet wird, muss die Aktivität mit der Zuordnung eine manuelle Aktivität des Typs `Parallel` sein.
- Wenn ein indexiertes Element aus dem parallelen Listen-Workflowdatenobjekt in einer Geschäftsobjektzuordnung verwendet wird, muss die Aktivität mit der Zuordnung eine parallele Aktivität sein (d. h. 'ParallelListWDO[Context_Parallel.occurrenceCount]'). Das anhand des Workflowdatenobjekts 'Context-_Parallel' indexierte Workflowdatenobjekt muss ein Listen-Workflowdatenobjekt für eine parallele Aktivität sein.

9.5.3 Code

API für 'BusinessObjectAssociationAdmin'

In der Klasse `BusinessObjectAssociationAdmin` stehen eine Reihe von Funktionen zur Bearbeitung von Geschäftsobjektzuordnungen bereit. Weitere Informationen zu den verfügbaren Funktionen finden Sie in der zugehörigen Javadoc-Spezifikation für die Klasse `BusinessObjectAssociationAdmin`.

9.5.4 Laufzeitinformationen

Geschäftsobjektzuordnungen haben keine funktionalen Auswirkungen auf die Ausführung einer manuellen Aktivität. Die Workflow-Engine untersucht einfach die Metadaten und fügt für jede angegebene Geschäftsobjektzuordnung einen Datensatz in die Entität `BizObjAssociation` ein. Der Geschäftsobjekttyp, der Wert der Attributzuordnung des Workflowdatenobjekts und die Kennung der mit der manuellen Aktivität verknüpften neu erstellten Aufgabe werden alle bei der Erstellung dieses Datensatzes verwendet.

9.6 Event-Wait

Da für eine manuelle Aktivität einige Benutzeraktionen durchgeführt werden müssen, bevor sie abgeschlossen und der Prozess fortgesetzt werden kann, muss es eine Möglichkeit geben, die Workflow-Engine über die erfolgreiche Durchführung der benötigten Aktionen zu benachrichtigen. Da diese Semantik der eines Event-Waits ähnelt, wird der Event-Wait-Mechanismus für manuelle Aktivitäten wiederverwendet.

9.6.1 Voraussetzungen

- Die Details zu einem Event-Wait und dessen zugehörigen Metadaten (die auch für eine manuelle Aktivität gelten) sind in Kapitel 8, „Event-Wait", auf Seite 49 beschrieben.

9.6.2 Beschreibung von Workflowdatenobjekten des Typs 'Context'

Das Workflowdatenobjekt 'Context_Task' kann in den Eingabezuordnungen für Fristfunktionen verwendet werden, die dem Event-Wait einer manuellen Aktivität zugeordnet sind. Es steht zudem für die Eingabezuordnungen zur Verfügung, die mit Zuordnungen für Zuteilungsfunktionen und für Zuteilungsregeln verknüpft sind. Außerdem kann es als Zuordnung für die Ereignisübereinstimmungsdaten eines angegebenen Event-Waits verwendet werden, das einer manuellen Aktivität zugeordnet ist. Diese Workflowdatenobjekt und seine Attribute wurden bereits weiter oben beschrieben (siehe 9.3.5, „Beschreibung von Workflowdatenobjekten des Typs 'Context'", auf Seite 67).

Kapitel 10. Entscheidung

10.1 Voraussetzungen

- Die Basisdetails, die für alle von Cúram Workflow unterstützten Aktivitätstypen gelten, sind in Kapitel 6, „Basisaktivität", auf Seite 31 beschrieben und gelten für die nachfolgend beschriebene Entscheidungsaktivität.
- Es gibt zudem Workflowmetadatenkonstrukte, die sowohl für manuelle als auch für Entscheidungsaktivitäten genutzt werden (z. B. Zuteilungsstrategie, Aufgabenbetreff, Aufgabenfrist usw.). Einzelheiten zu diesen Konstrukten finden Sie in Kapitel 9, „Manuell", auf Seite 61.

10.2 Übersicht

Eine typische Anforderung an Geschäftsprozesse ist, dass durch menschliches Eingreifen Entscheidungen getroffen werden, für die es eine einfache Antwort gibt. Ein Beispiel für eine solche Entscheidung ist das Genehmigen oder Ablehnen eines Falles oder das Bereitstellen einfacher Informationen wie das Alter des Anspruchstellers. Für das Einholen solcher Informationen mithilfe von manuellen Aktivitäten wäre es erforderlich, dass für jede Frage ein anderer Benutzerschnittstellenbildschirm in der Anwendung verfügbar wäre. Dies kann mühsam sein und da sich Prozessdefinitionen mit der Zeit verändern können, wären solche Benutzerschnittstellenbildschirme häufig nur temporärer Natur.

Bei der Entscheidungsaktivität handelt es sich um eine spezialisierte Version einer manuellen Aktivität, in deren Rahmen über eine metadatengesteuerte Benutzerschnittstelle einfache Fragen gestellt werden. Die Fragen und möglichen Antworten sind in den Aktivitätsmetadaten gespeichert; so kann eine einzelne Benutzerschnittstelle für eine Vielzahl verschiedener Fragen verwendet werden. Es werden derzeit zwei Typen von Fragen unterstützt. Dabei handelt es sich einerseits um Multiple-Choice-Fragen und andererseits um Fragen, auf die eine Antwort erforderlich ist, die in einem Feld in der Benutzerschnittstelle bereitgestellt werden kann.

10.3 Aufgabendetails

Ähnlich wie manuelle Aktivitäten (siehe Kapitel 9, „Manuell", auf Seite 61) benachrichtigen Entscheidungsaktivitäten Benutzer über erforderliche anstehende Aufgaben, die diesen Benutzern dann basierend auf der definierten Zuteilungsstrategie zugewiesen werden. Die entsprechende Aufgabe stellt automatisch einen Link zu einer Benutzerschnittstellenseite in der Anwendung her, auf der die Entscheidungsfrage aus den Fragenmetadaten der Entscheidungsaktivität zusammengestellt wird. Sobald die Entscheidungsantwort bereitgestellt wurde, wird der Workflow fortgesetzt. Eine Entscheidungsaktivität kann daher nur über eine zugehörige Aufgabenaktion verfügen, wobei für diese Aktion keine Aktionsseite definiert sein muss.

Zusätzlich zu der Aufgabeaktion können einer Entscheidungsaktivität keine oder mehrere sekundäre Aktionen zugeordnet sein. Sekundäre Aktionen enthalten einen Link zu einer Seite, auf der zusätzliche Informationen für den Benutzer zur Beantwortung der Entscheidungsfrage vorhanden sind.

© Copyright IBM Corp. 2012, 2013

10.3.1 Metadaten

```
<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">
              Participant Home Page %1n for Case %2n.
              </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
Beschreibt die Zuteilungsstrategie, die zum Bestimmen des Benutzers verwendet wird, der der zugehörigen Aufgabe zugeordnet ist. Detaillierte Informationen zu Zuteilungsstrategien finden Sie im Abschnitt 9.4, „Zuteilungsstrategie", auf Seite 67.

message
Gibt den Betreff der parametrisierten Nachricht der erstellten Aufgabe an. Detaillierte Informationen zu parametrisierten Nachrichten finden Sie im Abschnitt Kapitel 9, „Manuell", auf Seite 61.

decision-action
Gibt den Aktionstext der parametrisierten Nachricht an, die der Aufgabe zugeordnet ist. Der Benutzer klickt auf diesen Aktionstext, um einen automatisch erzeugten Entscheidungsbildschirm der Benutzerschnittstelle mit der entsprechenden Frage aufzurufen.

deadline
Beschreibt die Fristdetails für die Entscheidungsaktivität. Wenn für die Entscheidungsaktivität innerhalb der angegebenen Fristdauer keine Antwort bereitgestellt wird, wird die zugehörige Fristhandlermethode aufgerufen. Weitere Informationen zu Fristen finden Sie im Abschnitt 8.4, „Frist", auf Seite 52.

secondary-actions
Beschreibt alle optionalen sekundären Aktionen, die in die Entscheidungsaktivität eingeschlossen werden können.

secondary-action
Eine sekundäre Aktion enthält eine parametrisierte Nachricht und einen parametrisierten Link zu unterstützenden Informationen, um dem Benutzer bei der Beantwortung der Entscheidungsfrage zu helfen. Informationen zu parametrisierten Nachrichten und Links in Aktionen finden Sie im Abschnitt 9.3.1, „Metadaten", auf Seite 62.

page-id
Stellt die Kennung der Cúram-Zielseite dar, die die ergänzenden Informationen für die sekundäre Aktion enthält, die über einen Link abgerufen werden können.

multiple-occurring-action
Gibt an, dass diese sekundäre Aktion eine mehrfach auftretende Aktion darstellt. Wenn also diese Metadaten für eine sekundäre Aktion angegeben werden, erstellt die Workflow-Engine beim Ausführen der Aktivität einen Datensatz der sekundären Aktion für jedes Element im Listen-Workflowdatenobjekt, das als die mehrfach auftretende Aktion festgelegt wurde.

Wenn eine sekundäre Aktion als mehrfach auftretende Aktion verwendet wird, muss ein Attribut des verknüpften Listen-Workflowdatenobjekts als Linkparameter für die sekundäre Aktion verwendet werden.

list-wdo-name
Der Name des Listen-Workflowdatenobjekts, das mit der mehrfach auftretenden Aktion verwendet wird.

10.3.2 Validierungen

- Es muss ein Betreff für die Aktivität definiert sein.
- Bei jedem Attribut des Workflowdatenobjekts, das dem Betreff für eine Entscheidungsaktivität zugeordnet ist, muss es sich um ein gültiges Attribut handeln.
- Wenn ein indexiertes Element aus einem Listen-Workflowdatenobjekt (d. h. 'CaseList[Context_Loop-.loopCount]') als Betrefftextparameter der Entscheidung verwendet wird, muss das Workflowdatenobjekt ein Listen-Workflowdatenobjekt sein und die Aktivität mit der Zuordnung muss in einer Schleife enthalten sein.
- Wenn das Workflowdatenobjekt `Context_Parallel` als Betrefftextparameter der Entscheidung verwendet wird, muss die Aktivität mit der Zuordnung eine Entscheidungsaktivität des Typs `Parallel` sein.

- Wenn ein indexiertes Element aus dem parallelen Listen-Workflowdatenobjekt als Betrefftextparameter der Entscheidung verwendet wird, muss die Aktivität mit der Zuordnung eine parallele Aktivität sein (d. h. 'ParallelListWDO[Context_Parallel.occurrenceCount]'). Das anhand des Workflowdatenobjekts 'Context_Parallel' indexierte Workflowdatenobjekt muss ein Listen-Workflowdatenobjekt für eine parallele Aktivität sein.
- Die Anzahl von Platzhaltern, die im Betreff- und im Aktionstext verwendet wird, muss der Anzahl der zugeordneten Attribute des Workflowdatenobjekts (für alle Ländereinstellungen) entsprechen.
- Wenn ein indexiertes Element aus einem Listen-Workflowdatenobjekt (d. h. 'CaseList[Context_Loop-.loopCount]') als Aufgabenaktionstextparameter der Entscheidung verwendet wird, muss das Workflowdatenobjekt ein Listen-Workflowdatenobjekt sein und die Aktivität mit der Zuordnung muss in einer Schleife enthalten sein.
- Wenn das Workflowdatenobjekt `Context_Parallel` als Aktionstextparameter der Entscheidung verwendet wird, muss die Aktivität mit der Zuordnung eine Entscheidungsaktivität des Typs `Parallel` sein.
- Wenn ein indexiertes Element aus dem parallelen Listen-Workflowdatenobjekt als Aktionstextparameter der Entscheidung verwendet wird, muss die Aktivität mit der Zuordnung eine parallele Aktivität sein (d. h. 'ParallelListWDO[Context_Parallel.occurrenceCount]'). Das anhand des Workflowdatenobjekts 'Context_Parallel' indexierte Workflowdatenobjekt muss ein Listen-Workflowdatenobjekt für eine parallele Aktivität sein.
- Es muss eine Zuteilungsstrategie definiert sein.
- Das als Zuteilungsstrategie angegebene Zuteilungsziel, die Zuteilungsfunktion oder das Regelwerk muss gültig sein. Wenn der Zuteilungstyp eine Funktion ist, muss es sich um eine gültige Cúram-Geschäftsmethode handeln, die im Anwendungsklassenpfad vorhanden ist. Ist der Zuteilungstyp eine Regel, muss es sich um ein gültiges Regelwerk der Anwendung handeln.
- Falls der optionale Fristhandler angegeben ist, muss er eine gültige Cúram-Geschäftsmethode sein.
- Alle Eingabezuordnungen des Fristhandlers müssen gültige Zuordnungen sein. Das bedeutet, dass alle von der angegebenen Methode benötigten Eingabeparameterfelder gültigen Workflowdatenobjekt-Attributen des richtigen Typs zugeordnet werden.
- Für jede sekundäre Aktion muss ein Seitenlink angegeben sein, der keine Leerzeichen enthalten darf.
- Für jede sekundäre Aktion muss eine Nachricht angegeben sein.
- Der Nachrichtentext für die sekundäre Aktion muss eine Reihe von Platzhaltern enthalten, deren Anzahl der Anzahl der angegebenen Nachrichtenparametern entspricht.
- Nachrichtenparameter der sekundären Aktion müssen gültigen Workflowdatenobjekt-Attributen des richtigen Typs zugeordnet sein.
- Seitenlinkparameter der sekundären Aktion müssen gültigen Workflowdatenobjekt-Attributen zugeordnet sein.
- Wenn ein indexiertes Element aus einem Listen-Workflowdatenobjekt (d. h. 'ChildDependents[Context-_Loop.loopCount]') in den Zuordnungen für die Textparameter oder Linkparameter der sekundären Aktion verwendet wird, muss das Workflowdatenobjekt ein Listen-Workflowdatenobjekt sein und die Aktivität mit der Zuordnung muss in einer Schleife enthalten sein.
- Wenn das Workflowdatenobjekt `Context_Parallel` in den Zuordnungen für die Textparameter oder Linkparameter der sekundären Aktion verwendet wird, muss die Aktivität mit der Zuordnung eine Entscheidungsaktivität des Typs `Parallel` sein.
- Wenn ein indexiertes Element aus dem parallelen Listen-Workflowdatenobjekt in den Zuordnungen für die Textparameter oder Linkparameter der sekundären Aktion verwendet wird, muss die Aktivität mit der Zuordnung eine parallele Aktivität sein (d. h. 'ParallelListWDO[Context_Parallel.occurrence-Count]'). Das anhand des Workflowdatenobjekts 'Context_Parallel' indexierte Workflowdatenobjekt muss ein Listen-Workflowdatenobjekt für eine parallele Aktivität sein.
- Das Workflowdatenobjekt, das für die Verwendung in der mehrfach auftretenden Aktion angegeben wurde, muss bezüglich der zugehörigen Workflowprozessdefinition ein gültiges Workflowdatenobjekt sein. Es muss zudem vom Typ 'Liste' sein.

- Mindestens ein Attribut des Workflowdatenobjekts für die mehrfach auftretende Aktion muss in den Linkparametern verwendet werden, die für die mehrfach auftretende Aktion angegebenen wurden.

10.3.3 Laufzeitinformationen

Wenn eine Entscheidungsaktivität ausgeführt wird, erstellt die Workflow-Engine die zugehörige Aufgabe. Eine Momentaufnahme der Daten des Workflowdatenobjekts, die für den Betreff der Entscheidungsaktivität und die Parameter des Aktionstextes sowie für alle Nachrichtentexte für die sekundäre Aktion und für Linkparameter erforderlich sind, wird erstellt und gespeichert. Die der Entscheidungsaktivität zugeordnete Zuteilungsstrategie wird aufgerufen, um die Benutzer zu bestimmen, denen die Entscheidungsaufgabe zugewiesen wird. Zudem erstellt die Workflow-Engine einen Event-Wait für das Ereignis `DECISION.MADE` mit der zugehörigen Aufgabenkennung als Ereignisübereinstimmungsdaten. Der Workflow wird an dieser Stelle dann angehalten, um auf das Auslösen dieses Ereignisses zu warten, das das Ergebnis der getroffenen Entscheidung anzeigt.

10.4 Fragendetails

Die Entscheidungsaktivität unterstützt derzeit als Fragenformat sowohl Multiple-Choice-Fragen als auch Freitextfragen. Die automatisch erstellte Entscheidungsseite untersucht das erforderliche Fragenformat und generiert die entsprechende Frage aus den Fragenmetadaten, sobald der Benutzer auf die der Aufgabe zugeordnete Aktion klickt.

10.4.1 Metadaten

10.4.1.1 Mehrfachauswahl (Multiple-Choice)

```
<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="en">Yes</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="en">No</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
Steht für die der Entscheidungsaktivität zugeordnete Frage, die für eine Multiple-Choice-Frage die unten aufgeführten Metadaten enthält.

message
Stellt den parametrisierten Text der Frage für alle Ländereinstellungen dar.

answers
Stellt eine Liste von Antworten dar, aus der der Benutzer eine Antwort für die Multiple-Choice-Frage auswählen kann.

multiple-selection
Stellt ein Flag dar, das anzeigt, ob der Benutzer aus den bereitgestellten Antworten mehrere oder nur eine Antwort auswählen kann.

answer
Stellt eine Antwort dar, die der Benutzer auswählen kann. Für eine Multiple-Choice-Frage muss mindestens eine Antwort bereitgestellt werden.

name Stellt den Namen der Antwort dar. Sobald der Benutzer eine Antwort bzw. mehrere Antworten auswählt, werden die Namen der ausgewählten Antworten an die Workflow-Engine weitergeleitet und der Prozess wird fortgesetzt. Da die Engine diese Antworten ähnlich wie Workflowdatenobjekt-Attribute verarbeitet, müssen die Namen der Antworten gültige Java-Kennungen sein.

answer-text
Stellt den Antworttext dar, den der Benutzer für alle Ländereinstellungen auswählen kann.

choice-output-mapping
Dieser Tag umschließt die Metadaten, die beschreiben, wie die Ausgabe von einer Multiple-Choice-Frage als persistent definiert wird.

wdo-attribute
Der Name des Workflowdatenobjekt-Attributs, der zum Speichern des Werts der Multiple-Choice-Antwort verwendet wird.

selected-value
Falls angegeben, wird der Wert in diesem Element für das Workflowdatenobjekt-Attribut als persistent definiert, wenn diese Antwort vom Benutzer ausgewählt wurde. Wenn das Workflowdatenobjekt-Attribut den Typ 'Boolesch' aufweist, muss dieser Wert nicht angegeben werden und es wird der Standardwert `true` verwendet.

not-selected-value
Falls angegeben, wird der Wert in diesem Element für das Workflowdatenobjekt-Attribut als persistent definiert, wenn diese Antwort nicht vom Benutzer ausgewählt wurde. Wenn das Workflowdatenobjekt-Attribut den Typ 'Boolesch' aufweist, muss dieser Wert nicht angegeben werden und es wird der Standardwert `false` verwendet.

10.4.1.2 Freier Text

```
<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
Steht für die dieser Entscheidungsaktivität zugeordnete Frage, die für eine Freitextfrage die unten aufgeführten Metadaten enthält.

message
Stellt den parametrisierten Text der Frage für alle Ländereinstellungen dar.

free-text
Enthält die Details der Freitextantwort, die der Benutzer bereitstellen muss.

type Stellt den erforderlichen Datentyp der Freitextantwort dar, die bereitgestellt werden muss.

wdo-attribute
Stellt das Workflowdatenobjekt-Attribut dar, das die Freitextantwort wieder zurück zur Workflow-Engine zuordnet.

10.4.2 Validierungen

- Für eine Entscheidungsaktivität müssen das Anwortformat und der Fragetext angegeben werden.
- Die Anzahl von Platzhaltern, die im Fragetext verwendet wird, muss der Anzahl der zugeordneten Attribute des Workflowdatenobjekts (für alle Ländereinstellungen) entsprechen.
- Wenn ein indexiertes Element aus einem Listen-Workflowdatenobjekt (d. h. 'CaseList[Context_Loop-.loopCount]') als Fragetextparameter verwendet wird, muss das Workflowdatenobjekt ein Listen-Workflowdatenobjekt sein und die Aktivität mit der Zuordnung muss in einer Schleife enthalten sein.
- Wenn das Workflowdatenobjekt `Context_Parallel` als Fragetextparameter verwendet wird, muss die Aktivität mit der Zuordnung eine Entscheidungsaktivität des Typs `Parallel` sein.
- Wenn ein indexiertes Element aus dem parallelen Listen-Workflowdatenobjekt als Fragetextparameter verwendet wird, muss die Aktivität mit der Zuordnung eine parallele Aktivität sein (d. h. 'ParallelList-WDO[Context_Parallel.occurrenceCount]'). Das anhand des Workflowdatenobjekts 'Context_Parallel' indexierte Workflowdatenobjekt muss ein Listen-Workflowdatenobjekt für eine parallele Aktivität sein.

- Für eine Frage mit dem Antwortformat 'Freiformtext' muss der Datentyp der Antwort angegeben werden. Zudem muss das zugeordnete Attribut des Workflowdatenobjekts gültig sein und dem Datentyp für die Antwort entsprechen. Das zugeordnete Attribut kann kein konstantes Workflowdatenobjekt-Attribut sein.
- Für eine Frage mit dem Antwortformat 'Liste' muss mindestens eine Antwortoption aufgelistet sein. Bei allen Antwortnamen muss es sich um gültige Java-Attributnamen handeln.

10.4.3 Laufzeitinformationen

Wenn eine Antwort auf eine Entscheidungsaktivitätsfrage gegeben wurde, wird das Ereignis `DECISION.MADE` mit der Aufgabenkennung der Entscheidungsaktivitätsaufgabe ausgelöst, die als Ereignisübereinstimmungsdaten verwendet wird. Der Workflowereignishandler verarbeitet dieses Ereignis, woraufhin der Workflowprozess fortgesetzt wird.

Wenn die gegebene Antwort eine Freitextantwort ist, wird sie dem angegebenen Attribut des Workflowdatenobjekts zugeordnet und kann so bei Bedarf zu einem späteren Zeitpunkt im Prozess wiederverwendet werden.

10.4.4 Beschreibung von Workflowdatenobjekten des Typs 'Context'

Das Workflowdatenobjekt 'Context_Decision' kann in Datenelement- oder Funktionsbedingungen (siehe Kapitel 16, „Bedingungen", auf Seite 113) für einen Übergang von einer Entscheidungsaktivität verwendet werden. Die verfügbaren Attribute hängen von dem für die Aktivität definierten Antwortformat ab.

Freitextantwort

Wenn es sich bei dem Antwortformat um eine Freitextantwort handelt, ist das folgende Attribut verfügbar:

Context_Decision.value

Der Wert der gegebenen Freitextantwort. Dieser Wert kann in Übergangsbedingungen verwendet und einem bestimmten Workflowdatenobjekt-Attribut zugeordnet werden.

Multiple-Choice-Antwort

In diesem Fall wird das Workflowdatenobjekt 'Context_Decision' mit Attributen für jede der verfügbaren Antworten aufgefüllt. Dabei ist jedes Attribut vom Typ 'boolesch'. Dies zeigt an, ob die jeweilige Antwort ausgewählt wurde oder nicht. In dem Metadatenabschnitt der Multi-Choice-Antwort weiter oben (siehe 10.4.1.1, „Mehrfachauswahl (Multiple-Choice)", auf Seite 82 gilt Folgendes: Wenn der Benutzer die erste Antwort ausgewählt hat ('Yes'), würde das mit dem folgenden auf `true` gesetzten Workflowdatenobjekt-Attribut des Typs 'Context_Decision' wiedergegeben werden:

Context_Decision.yesAnswer

Dies stellt einen booleschen Wert dar, der anzeigt, ob die Antwort 'Yes' für die Frage ausgewählt wurde. Das Attribut kann nur in Übergangsbedingungen aus der Entscheidungsaktivität verwendet werden.

Wenn der Benutzer alternativ die zweite Antwort ('No') ausgewählt hat, würde das mit dem folgenden auf `true` gesetzten Workflowdatenobjekt-Attribut des Typs 'Context_Decision' wiedergegeben werden:

Context_Decision.noAnswer

Dies stellt einen booleschen Wert dar, der anzeigt, ob die Antwort 'No' für die Frage ausgewählt wurde. Auch dieses Attribut kann nur in Übergangsbedingungen aus der Entscheidungsaktivität verwendet werden.

Kapitel 11. Subflow

11.1 Voraussetzungen

- Die Basisdetails, die für alle von Cúram Workflow unterstützten Aktivitätstypen gelten, sind in Kapitel 6, „Basisaktivität", auf Seite 31 beschrieben und gelten für die nachfolgend beschriebene Subflowaktivität.

11.2 Übersicht

Beim Gestalten eines komplexen Geschäftsprozesses kann dieser zu groß werden, um als einzelne umfangreiche Prozessdefinition verwaltet werden zu können. Mithilfe einer Subflowaktivität kann eine weitere Prozessdefinition als Teil eines anderen Prozesses umgesetzt werden.

Es kann unabhängig der Bedenken bezüglich der Größe eines Prozesses hilfreich sein, Prozessdefinitionen als eine Gruppe von Subflows zu gestalten. Dadurch könnten Abschnitte eines Geschäftsprozesses geändert werden, ohne dass sich dies auf andere Abschnitte auswirkt. Außerdem können Subflowprozesse als wiederverwendbare Komponenten dienen, die Kunden zum Erstellen ihrer eigenen übergeordneten Prozessdefinitionen wiederverwenden können.

11.3 Subflowprozess

Um einen Prozess als Subflow umzusetzen, muss in der Subflowaktivität der jeweils umzusetzende Prozess namentlich angegeben werden. Wie auch bei anderen Prozessumsetzungsmechanismen wird jeweils die neueste freigegebene Version des Prozesses umgesetzt.

Subflows können *synchron* umgesetzt werden. Das bedeutet, dass die Verzweigung des übergeordneten Workflows, der die Subflowaktivität enthält, die den Subflowprozess gestartet hat, so lange wartet, bis der Subflowprozess beendet ist, bevor die Bearbeitung fortgesetzt wird.

Alternativ kann ein Subflow auch *asynchron* umgesetzt werden. Dies bedeutet, dass, sobald die Subflowaktivität den Subflowprozess startet, die Verzweigung mit der Subflowaktivität sofort weiter verarbeitet wird, ohne dass das Ergebnis des Subflowprozesses sich auf den übergeordneten Prozess auswirkt.

11.3.1 Metadaten

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

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

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

subflow

workflow-process
Der Name des Workflowprozesses, der beim Ausführen der Aktivität gestartet werden soll. Bei Prozessnamen muss die Groß-/Kleinschreibung berücksichtigt werden, und der hier angegebene Prozessname muss genau mit dem Namen des Prozesses übereinstimmen, der als Subflow gestartet werden soll.

synchronous
Ein Flag, das anzeigt, ob der Subflow im Vergleich zu seinem übergeordneten Prozess synchron ausgeführt ausgeführt werden sollte (siehe 11.3, „Subflowprozess").

© Copyright IBM Corp. 2012, 2013

11.3.2 Validierungen

- Für die Subflowaktivität muss ein Workflowprozess angegeben sein.
- Der als Subflow angegebene Workflowprozess muss mindestens eine freigegebene Version aufweisen.

11.4 Eingabezuordnungen

Für den Subflow werden Daten bereitgestellt, wenn diese von den Workflowdatenobjekten des übergeordneten Prozesses umgesetzt werden. Die Subflowaktivität definiert die Zuordnung zwischen den Workflowdatenobjekten des übergeordneten Prozesses und den Umsetzungsdaten des Subflows.

11.4.1 Metadaten

```
<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

Gibt an, wie Daten aus dem aktuell ausgeführten Prozess einem Unterprozess als Umsetzungsdaten zugeordnet werden, wenn der Unterprozess gestartet wird. Der als Subflow angegebene Prozess verfügt möglicherweise nicht über Workflowdatenobjekt-Attribute, die als für die Umsetzung erforderlich markiert sind, in welchem Fall keine Eingabezuordnungen erforderlich sind.

mapping

Der Tag `mapping` stellt die Daten dar, die von einem Workflowdatenobjekt-Attribut zu einem Attribut in dem Prozess geschoben werden, der als Subflow umgesetzt wird. Wenn eine Liste der Daten zur Umsetzung des Subflowprozesses erforderlich ist, können zu diesem Zweck Attribute aus Listen-Workflowdatenobjekten verwendet werden. Die Anzahl der angegebenen Zuordnungen hängt davon ab, wie viele Attribute als für die Umsetzung erforderlich im Subflowprozess markiert sind, da alle diese Attribute beim Start des Prozesses mit Daten aufgefüllt werden müssen.

source-attribute
Gibt ein Workflowdatenobjekt-Attribut aus dem übergeordneten Prozess an, das zum Auffüllen des zugehörigen Attributs im Subflow bei dessen Umsetzung verwendet wird.

target-attribute
Gibt ein Workflowdatenobjekt-Attribut aus dem Subflow an, das bei der Umsetzung mit Daten aus dem zugehörigen Attribut im übergeordneten Prozess aufgefüllt werden soll.

source/target-attribute

wdo-name
Gibt den Namen eines wie in Kapitel 4, „Workflowdatenobjekte", auf Seite 17 beschriebenen Cúram-Workflowdatenobjekt an.

name Gibt den Namen eines wie in Kapitel 4, „Workflowdatenobjekte", auf Seite 17 beschriebenen Cúram-Workflowdatenobjekt-Attributs an.

11.4.2 Validierungen

- Alle Workflowdatenobjekt-Attribute, die im Subflow als *für die Umsetzung erforderlich* markiert sind, müssen in den Eingabezuordnungen angegeben werden. Wenn keine Attribute als für die Umsetzung erforderlich markiert wurden, sollten keine Eingabezuordnungen festgelegt werden.
- Der Datentyp des Workflowdatenobjekt-Attributs, das durch den Tag `target-attribute` angegeben ist, muss dem durch den Tag `source-attribute` angegebenen Attribut entsprechen bzw. von diesem Attribut zugewiesen werden können.
- Wenn ein indexiertes Element aus einem Listen-Workflowdatenobjekt (d. h. 'PersonDetailsList[Context_Loop.loopCount]') im Tag `source-attribute` der Eingabezuordnung des Subflows angegeben ist, muss dieses Workflowdatenobjekt ein Listen-Workflowdatenobjekt sein und die Subflowaktivität mit der Eingabezuordnung muss in einer Schleife enthalten sein. Der Datentyp des Workflowdatenobjekt-Attributs, das durch den Tag `target-attribute` angegeben ist, muss dem durch den Tag `source-attribute` angegebenen Attribut entsprechen bzw. von diesem Attribut zugewiesen werden können.
- Wenn die angegebene Eingabezuordnung des Subflows ein Listen-Workflowdatenobjekt verwendet, dann müssen die Workflowdatenobjekt-Attribute für den Tag `source-attribute` des übergeordneten Prozesses und den Tag `target-attribute` des Subflowprozesses Listen-Workflowdatenobjekte sein.

11.5 Ausgabezuordnungen

Ausgabezuordnungen gelten nur für *synchrone* Subflowaktivitäten, da asynchrone Subflows weiter verarbeitet werden können, ohne dass die Aktivität abgeschlossen werden muss. Dem übergeordneten Prozess werden Daten aus der Subflowaktivität bereitgestellt, nachdem diese abgeschlossen wurde. Die Subflowaktivität definiert die Zuordnung zwischen dem Workflowdatenobjekt-Attribut eines Subflows und dem Workflowdatenobjekt-Attribut des übergeordneten Prozesses.

11.5.1 Metadaten

```
<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
Gibt an, wie Daten aus dem aufgerufenen Unterprozess bei Abschluss des Unterprozesses dem übergeordneten Prozess zugeordnet werden. Wenn der als Subflow angegebene Prozess keine definierten Ausgabezuordnungen aufweist, wird der Subflow ganz normal beendet.

mapping
Stellt die Daten dar, die von einem Workflowdatenobjekt-Attribut an ein Attribut im übergeordneten Prozess übertragen werden. Wenn eine Liste von Daten vom Subflow an den übergeordneten Prozess übertragen wird, können zu diesem Zweck Attribute aus Listen-Workflowdatenobjekten verwendet werden. Die Anzahl der angegebenen Zuordnungen wird durch die Anzahl der angegebenen Ausgabezuordnungen bestimmt.

source-attribute
Gibt ein Workflowdatenobjekt-Attribut aus dem Subprozess an, das zum Auffüllen des zugehörigen Attributs im übergeordneten Prozesses bei dessen Beendigung verwendet wird.

target-attribute
Gibt ein Workflowdatenobjekt-Attribut aus dem übergeordneten Prozess an, das bei Beendigung des Prozesses mit Daten aus dem zugehörigen Attribut im Subprozess aufgefüllt werden soll.

source/target-attribute

wdo-name
Gibt den Namen eines wie in Kapitel 4, „Workflowdatenobjekte", auf Seite 17 beschriebenen Cúram-Workflowdatenobjekt an.

name Gibt den Namen eines wie in Kapitel 4, „Workflowdatenobjekte", auf Seite 17 beschriebenen Cúram-Workflowdatenobjekt-Attributs an.

11.5.2 Validierungen

- Das übergeordneten Attribut `target-attribute` und das Subflowattribut `source-attribute` des Workflowdatenobjekts, die in der Ausgabezuordnung des Subflows verwendet werden, müssen bezüglich der zugehörigen Prozessdefinition gültige Attribute sein.
- Der Datentyp des Workflowdatenobjekt-Attributs, das durch den übergeordneten Tag `target-attribute` angegeben ist, muss dem durch den Subflow-Tag `source-attribute` angegebenen Attribut entsprechen bzw. von diesem Attribut zugewiesen werden können.
- Wenn die angegebene Ausgabezuordnung des Subflows ein Listen-Workflowdatenobjekt verwendet, dann müssen die zugeordneten Workflowdatenobjekt-Attribute für den Tag `target-attribute` des übergeordneten Prozesses und den Tag `source-attribute` des Subflowprozesses beide vom Typ 'Liste' sein.

Kapitel 12. Schleifenbeginn und Schleifenende

12.1 Voraussetzungen

- Die Basisdetails, die für alle von Cúram Workflow unterstützten Aktivitätstypen gelten, sind in Kapitel 6, „Basisaktivität", auf Seite 31 beschrieben und gelten für die nachfolgend beschriebenen Schleifenbeginn- und Schleifenendaktivitäten.

12.2 Übersicht

Viele Geschäftsprozesse müssen so oft *wiederholt* werden, bis eine bestimmte Bedingung erfüllt ist. In Cúram wird dies mithilfe der Schleifenbeginn- und Schleifenendaktivitäten implementiert. Alle Aktivitäten, die zwischen einer Schleifenbeginn- und der zugehörigen Schleifenendaktivität liegen, werden so lange wiederholt, bis die Schleife beendet ist.

In einer Prozessdefinition werden Schleifenbeginn- und Schleifenendaktivitäten paarweise angegeben und anhand der Metadaten kennt jeder Schleifenbeginn sein zugehöriges Schleifenende (und umgekehrt). Um eine Reihe von Aktivitäten zu einer Schleife hinzuzufügen, wird von der Schleifenbeginnaktivität ein Übergang zur ersten zu wiederholenden Aktivität erstellt. Nachfolgende Aktivitäten in der Abfolge werden mithilfe von Übergängen (wie auch außerhalb einer Schleife) miteinander verknüpft. Die letzte Aktivität in der Abfolge hat jedoch einen Übergang zu der Schleifenendaktivität. Oft wird an dieser Stelle vom Gefühl her versucht, einen Übergang von der Schleifenendaktivität zum Start herstellen, um so einen Zyklus zu erstellen. Dies ist jedoch falsch und führt zu einer ungültigen Prozessdefinition.

In einer Schleife müssen zudem die Kriterien angegeben werden, mit denen bestimmt wird, ob die Schleife beendet wird oder nicht. Um dieses Verhalten zu unterstützen, weist eine Schleife in Cúram Workflow eine Bedingung des Typs 'loop-exit' auf.

Schleifen können andere Schleifen enthalten, solange sie vollständig verschachtelt sind und nicht ineinandergreifen. Dadurch wird sichergestellt, dass die Schleifen und somit auch die Prozessdefinition eine gültige Blockstruktur aufweisen, wie sie von der Cúram-Workflow-Engine benötigt wird (siehe Kapitel 18, „Workflowstruktur", auf Seite 121).

12.2.1 Schleifentyp

Zusätzlich zu der Bedingung 'loop-exit' kann eine Schleife angeben, ob die Bedingung vor dem Ausführen der Schleife (eine While-Schleife) oder am Ende einer Scheifenausführung (eine Do-While-Schleife) getestet werden soll. Es kann sein, dass eine While-Schleife die Aktivitäten in der Schleife niemals ausführt und zur Aktivität springt, die nach der Schleife kommt, wenn die exit-Bedingung am Beginn der Schleife erfüllt wird. Eine Do-While-Schleife hingegen führt die Aktivitäten in der Schleife mindestens einmal aus.

12.3 Metadaten

12.3.1 Schleifenbeginnaktivität

© Copyright IBM Corp. 2012, 2013

```
<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

Der Tag `loop-type` gibt an, wie die Schleife ausgeführt wird (siehe 12.2.1, „Schleifentyp", auf Seite 91). Die beiden einzigen gültige Werte für das Attribut `name` sind `while` und `do-while`.

condition

Der Tag `condition` gibt die Bedingung an, die basierend auf Werten für Workflowdatenobjekte bewertet wird (siehe Kapitel 4, „Workflowdatenobjekte", auf Seite 17). Wenn Listen-Workflowdatenobjekte im Workflow vorhanden sind, werden zwei Attribute, die nicht Bestandteil der Metadaten der Workflowdatenobjekte sind, beim Erstellen eines Schleifenbedingungsausdrucks mithilfe eines Listen-Workflowdatenobjekts zur Verfügung gestellt. Diese lauten wie folgt:

- *size()*: Gibt in Form einer Zahl (einer Ganzzahl) die Anzahl von Elementen in der Liste an.
- *isEmpty()*: Gibt in Form eines booleschen Flags an, ob die Liste Elemente enthält oder nicht.

Die tatsächlichen Metadaten der Bedingung werden an anderen Stellen in den Metadaten der Prozessdefinition verwendet und werden daher in einem dafür vorgesehenen Kapitel (Kapitel 16, „Bedingungen", auf Seite 113) beschrieben.

block-endpoint-ref

Der Tag `block-endpoint-ref` in diesem Kontext ermöglicht es der Schleifenbeginnaktivität (`loop-begin-activity`), ihre zugehörige Schleifenendaktivität (`loop-end-activity`) zu erkennen. Diese Information ist für die Workflow-Engine bei der Ausführung der Schleife von Nutzen. Wenn beispielsweise eine Exitbedingung einer Schleife vor der Ausführung der Schleife den Wert `true` aufweist, zeigt der Tag `block-endpoint-ref` der Workflow-Engine an, von welcher Aktivität an sie die Ausführung des Prozesses fortsetzen soll.

12.3.2 Schleifenendaktivität

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

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

  </loop-end-activity>
```

block-endpoint-ref

Der Tag `block-endpoint-ref` in diesem Kontext ermöglicht es der Schleifenendaktivität (`loop-end-activity`), ihre zugehörige Schleifenbeginnaktivität (`loop-begin-activity`) zu erkennen. Diese Information ist für die Workflow-Engine bei der Ausführung der Schleife von Nutzen. Wenn beispielsweise eine Exitbedingung einer Schleife nach der Ausführung der Schleife den Wert `false` aufweist, zeigt der Tag `block-endpoint-ref` der Workflow-Engine an, von welcher Aktivität an sie eine weitere Iteration der Schleife beginnen soll.

12.4 Laufzeitinformationen

Es wird erwartet, dass Aktivitäten in einer Schleife während der Ausführung einer Prozessinstanz mehr als einmal ausgeführt werden. Damit die Prozessinstanzdaten durch nachfolgende Iterationen nicht beschädigt werden, wird jede Aktivitätsinstanz mit einer bestimmten Iteration verknüpft und kann so von der Workflow-Engine unabhängig von der Anzahl der Schleifenausführungen eindeutig identifiziert werden.

12.5 Beschreibung von Workflowdatenobjekten des Typs 'Context'

Das Workflowdatenobjekt 'Context_Loop' ist in den folgenden Situationen verfügbar:

- Bei der Erstellung der mit einer Schleifenbeginnaktivität verknüpften Schleifenbedingung.
- Bei der Erstellung der Bedingungen für ausgehende Übergänge von einer Schleifenbeginnaktivität oder einer beliebigen, in einer Schleife enthaltenen Aktivität (siehe Kapitel 16, „Bedingungen", auf Seite 113).
- Bei der Erstellung der Eingabezuordnungen für eine automatische Aktivität oder Subflowaktivität in einer Schleife.
- Bei der Erstellung der Eingabezuordnungen für eine beliebige, in einer Aktivität in einer Schleife enthaltenen Zuteilungsstrategie- oder Fristhandlerfunktion.
- Bei der Angabe eines Betrefftextparameters für eine manuelle Aktivität oder eine Entscheidungsaktivität, die in einer Schleife enthalten ist, oder für eine Benachrichtigung, die an eine Aktivität in einer Schleife angehängt ist.
- Bei der Angabe von Aktionstext- oder Aktionslinkparameter für eine manuelle Aktivität oder eine Entscheidungsaktivität, die in einer Schleife enthalten ist, oder für eine Benachrichtigung, die an eine Aktivität in einer Schleife angehängt ist.
- Bei der Angabe der Kennung für eine Geschäftsobjektzuordnung für eine manuelle Aktivität in einer Schleife.
- Bei der Angabe eines Fragetextparameters für eine Freiform- oder eine Multiple-Choice-Frage für eine Entscheidungsaktivität in einer Schleife.
- Bei der Angabe eines Textparameters für eine Benachrichtigung, die an eine in einer Schleife enthaltene Aktivität angehängt ist.

Zu den verfügbaren Attributen des Workflowdatenobjekts 'Context_Loop' zählen:

Context_Loop.loopCount
Die Häufigkeit, mit der eine Schleife durchlaufen wurde.

Kapitel 13. Parallel

13.1 Voraussetzungen

- Die Basisdetails, die für alle von Cúram Workflow unterstützten Aktivitätstypen gelten, sind in Kapitel 6, „Basisaktivität", auf Seite 31 beschrieben und gelten für die nachfolgend beschriebene parallele Aktivität.
- Da parallele Aktivitäten vorhandene Aktivitäten in einer Workflowprozessdefinition umschließen, beziehen sich die in Kapitel 9, „Manuell", auf Seite 61 und Kapitel 10, „Entscheidung", auf Seite 77 beschriebenen Metadaten auch auf die hier erläuterte parallele Aktivität.

13.2 Übersicht

Im Rahmen von Geschäftsprozessen kann es erforderlich sein, mehrere Aufgaben gleichzeitig an unterschiedliche Bearbeiter zu senden, um den Gesamtprozess zu beschleunigen. Wenn die Anzahl paralleler Pfade zur Entwicklungszeit bekannt ist, kann dies mithilfe eines Splits erreicht werden. In einige Fällen ist die Anzahl der Pfade jedoch erst zur Laufzeit bekannt. Solche Situationen können mithilfe von parallelen Aktivitäten modelliert werden.

Eine parallele Aktivität fungiert als umschließendes Element für bestehende Aktivitäten. Die Verwendung dieser neuen Aktivitäten zur Laufzeit wirkt sich so aus, dass mehrere Instanzen der umschlossenen Aktivität parallel ausgeführt werden. Bislang werden lediglich umschlossene Aktivitäten vom Typ "Manuell" (Kapitel 9, „Manuell", auf Seite 61) und "Entscheidung" (Kapitel 10, „Entscheidung", auf Seite 77) unterstützt. Daher entspricht das Ausführen einer parallelen Aktivität derzeit der parallelen Erstellung und Zuteilung mehrerer Aufgaben.

13.3 Metadaten

Eine parallele Aktivität muss den Typ der Aktivität angeben, die sie umschließt. Außerdem muss der parallelen Aktivität ein Listen-Workflowdatenobjekt zugeordnet sein. Die Anzahl der Elemente in diesem Objekt legt die Anzahl der Instanzen dieser umschlossenen Aktivität fest, die zur Laufzeit von der Workflow-Engine erstellt werden.

13.3.1 Generische Metadaten für eine parallele Aktivität

© Copyright IBM Corp. 2012, 2013

```
<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>


    oder ........

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

manual-activity/decision-activity

Gibt den Typ der Aktivität an, der von der parallelen Aktivität umschlossen wird. Derzeit werden zwei Typen von umschlossenen Aktivitäten unterstützt: manuelle (siehe Kapitel 9, „Manuell", auf Seite 61) und Entscheidungsaktivitäten (siehe Kapitel 10, „Entscheidung", auf Seite 77). Die Aktivitätstypen, die von einer parallelen Aktivität umschlossen werden können, sind in der Codetabelle `ParallelActivityType` enthalten.

list-wdo-name

Jeder parallelen Aktivität muss ein Listen-Workflowdatenobjekt zugeordnet sein. Die Anzahl der Instanzen der umschlossenen Aktivität, die zur Laufzeit erstellt werden, wird durch die Anzahl der Elemente in diesem Listen-Workflowdatenobjekt festgelegt.

13.3.2 Metadaten für eine parallele manuelle Aktivität

Das nachfolgende Beispiel veranschaulicht die Metadaten, die der umschlossenen Aktivität des Typs `Manuell` zugeordnet sind. Diese Metadaten entsprechen *genau* den in Kapitel 9, „Manuell", auf Seite 61 beschriebenen Metadaten für eine manuelle Aktivität und werden daher hier an dieser Stelle nicht noch einmal beschrieben. Alle Validierungen für die Zuordnungen der parallelen manuellen Aktivität sind auch in Kapitel 9, „Manuell", auf Seite 61 aufgeführt. Das Workflowdatenobjekt `Context_Parallel` und ein indexiertes Element aus dem Listen-Workflowdatenobjekt der parallelen Aktivität können in allen verfügbaren Zuordnungen für eine parallele manuelle Aktivität verwendet werden. Nachfolgend sind Beispiele für eine solche Verwendung aufgeführt:

```
<parallel-activity id="1" category="AC1">
      <list-wdo-name>EmployerDetailsListWDO</list-wdo-name>
      <manual-activity>
        ...
        <task>
          <message>
            <message-text>
              <localized-text>
                <locale language="en">Check employer
                details for %1s. This is employer number: %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>
```

13.3.3 Metadaten für eine parallele Entscheidungsaktivität

Das nachfolgende Beispiel veranschaulicht die Metadaten, die der umschlossenen Aktivität des Typs `Entscheidung` zugeordnet sind. Diese Metadaten entsprechen *genau* den in Kapitel 10, „Entscheidung", auf Seite 77 beschriebenen Metadaten für eine Entscheidungsaktivität und werden daher hier an dieser Stelle nicht noch einmal beschrieben. Alle Validierungen für die Zuordnungen der parallelen Entscheidungsaktivität sind auch in Kapitel 10, „Entscheidung", auf Seite 77 aufgeführt. Das Workflowdatenobjekt `Context_Parallel` und ein indexiertes Element aus dem Listen-Workflowdatenobjekt der parallelen Aktivität können in allen verfügbaren Zuordnungen für eine parallele Entscheidungsaktivität verwendet werden. Nachfolgend sind Beispiele für eine solche Verwendung aufgeführt:

```
<parallel-activity id="1" category="AC1">
      <list-wdo-name>ChildDetailsListWDO</list-wdo-name>
      <decision-activity>
        ...
        <message>
          <message-text>
            <localized-text>
              <locale language="en">In this task the details
              for child %1s must be validated. This is child
              number: %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="en">Validate the child details
                for %1s associated with this case %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="en">Are the details for this
                child whose first name is %1s and second name
                %2s correct?</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="en">Yes</locale>
                </localized-text>
              </answer-text>
            </answer>
            <answer name="answerNo">
              <answer-text>
                <localized-text>
```

13.3.4 Validierungen

- Für eine parallele Aktivität muss ein Workflowdatenobjekt angegeben werden. Dabei muss es sich um ein gültiges Listen-Workflowdatenobjekt bezüglich der zugehörigen Workflowprozessdefinition handeln.
- Alle anderen Validierungen für parallele Aktivitäten sind in den Kapiteln beschrieben, in den die Aktivitäten aufgeführt sind, die eine parallele Aktivität umschließen kann (Kapitel 9, „Manuell", auf Seite 61 und Kapitel 10, „Entscheidung", auf Seite 77).

13.3.5 Laufzeitinformationen

Die Workflow-Engine lädt die Instanzdaten für das mit der parallelen Aktivität verknüpfte Listen-Workflowdatenobjekt. Für jedes Element im Listen-Workflowdatenobjekt wird eine neue Instanz der umschlossenen Aktivität erstellt und ausgeführt. Was genau bei der Ausführung der Instanzen der umschlossenen Aktivität passiert, wird in den entsprechenden Kapiteln zu den Aktivitäten beschrieben, die eine parallele Aktivität umschließen kann (Kapitel 9, „Manuell", auf Seite 61 und Kapitel 10, „Entscheidung", auf Seite 77).

Zur Laufzeit verarbeitet die Workflow-Engine eine parallele Aktivität als mehrere Aktivitäten, die in einem Split/Join-Block des Typs *Parallel (AND)* enthalten sind. Es wird eine Aktivitätsinstanz pro Element im Listen-Workflowdatenobjekt der parallelen Aktivität erstellt (wenn diese Liste beispielsweise drei Elemente enthält, werden drei Aktivitätsinstanzen erstellt). Dadurch wird sichergestellt, dass zunächst alle der parallelen Aktivität zugeordneten Aktivitätsinstanzen erfolgreich ausgeführt werden müssen, bevor die eigentliche parallele Aktivität als abgeschlossen angesehen wird und der Workflow weiter fortgesetzt werden kann.

Um die mit einer parallelen Aktivität verknüpften Zuordnungen aufzulösen, wird jede Instanz der umschlossenen Aktivität einem Element aus dem Listen-Workflowdatenobjekt der parallelen Aktivität zugeordnet. Das Element wird mithilfe des Workflowdatenobjekts 'Context_Parallel' (z. B. 'ChildDetailsListWDO[Context_Parallel.occurrenceCount]') indexiert.

13.3.6 Beschreibung von Workflowdatenobjekten des Typs 'Context'

Jede Instanz einer parallelen Aktivität ist einem Element aus dem Listen-Workflowdatenobjekt für die parallele Aktivität zugeordnet. Auf dieses Element kann zugegriffen werden, indem das Workflowdatenobjekt 'Context_Parallel' zum Indexieren des Listen-Workflowdatenobjekts der parallelen Aktivität (z. B. 'ChildDetailsListWDO[Context_Parallel.occurrenceCount]') verwendet wird. Indexierte Elemente können dann für die gewöhnliche Zuordnung von Daten verwendet werden. Beispiele für solche Zuordnungen finden Sie in den Metadatenbeispielen weiter oben im Dokument (siehe 13.3.2, „Metadaten für eine parallele manuelle Aktivität", auf Seite 96 und 13.3.3, „Metadaten für eine parallele Entscheidungsaktivität", auf Seite 97. Das folgende Attribut steht für dieses Workflowdatenobjekt zur Verfügung:

Context_Parallel.occurrenceCount

Jede Instanz einer parallelen Aktivität ist einem Element aus dem Listen-Workflowdatenobjekt für die parallele Aktivität zugeordnet. Das Attribut `occurrenceCount` ist der Index dieses Elements innerhalb des Listen-Workflowdatenobjekts. Es handelt sich um einen nullbasierten Index in Form einer Ganzzahl.

Kapitel 14. Aktivitätsbenachrichtigungen

14.1 Übersicht

Die Workflow-Engine kann interessierte Benutzer über den Fortschritt einer Workflowprozessinstanz informieren. Grundsätzlich kann die Workflow-Engine beim Ausführen eine Benachrichtigung auslösen, wenn die Benachrichtigung in den Metadaten der zugehörigen Prozessdefinition angegeben wurde. Eine Benachrichtigung wird für eine Aktivität in Form zusätzlicher Metadaten angegeben, die einem beliebigen Aktivitätstyp angehängt werden können.

Beim Ausführen einer Aktivität prüft die Workflow-Engine, ob für diese Aktivität eine Benachrichtigung angegeben wurde. Ist dies der Fall, wird von der Workflow-Engine eine Benachrichtigung erstellt, die angibt, dass ein bestimmter Schritt im Workflowprozess ausgeführt wurde. Die Bereitstellung dieser Benachrichtigungen an den Benutzer wird durch den Benachrichtigungsbereitstellungsmechanismus bestimmt, der in der Cúram-Anwendung konfiguriert wird. Benachrichtigungen können mithilfe von E-Mails, als Alerts im Posteingang des Benutzers oder sowohl mit E-Mails als auch mit Alerts bereitgestellt werden.

14.2 Benachrichtigungsdetails

Bei einer Benachrichtigung handelt es sich ganz einfach um Informationen, die beim Ausführen eines Prozessschrittes an einen Bearbeiter gesendet werden. Benachrichtigungen erscheinen in Form von Alerts im Posteingang eines Benutzers oder als E-Mails. Die Bearbeiter, an die eine Benachrichtigung gesendet werden muss, werden durch die für die Benachrichtigung angegebene Zuteilungsstrategie (siehe 14.3, „Zuteilungsstrategie für Benachrichtigungen”, auf Seite 105) ermittelt. Die Details, die dem Benutzer im Alert oder in der E-Mail angezeigt werden, stammen aus den Benachrichtigungsmetadaten.

14.2.1 Metadaten

© Copyright IBM Corp. 2012, 2013

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

  ...

    <notification delivery-mechanism="DM1">
      <subject>
       <message>
          <message-text>
           <localized-text>
              <locale language="en">
              The case number %1n for Claimant %2s has
              been closed.
              </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">
              This case concerned %1n and claimant %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">
                View the task associated with the %1n case.
                </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">
                View the case details for %1n.
                </locale>
```

delivery-mechanism
Beschreibt den Mechanismus, der für die Übermittlung der Benachrichtigung verwendet wird. Die verfügbaren Mechanismen sind in der Codetabelle `DeliveryMechanism` der Anwendung aufgeführt. Sowohl die Cúram-Anwendung als auch Cúram-Kunden können diese Codetabelle erweitern und bei Bedarf weitere Mechanismen hinzufügen. Der angegebene Bereitstellungsmechanismus spielt in der Workflow-Engine keine funktionale Rolle, da er ganz einfach den in der Anwendung konfigurierten Mechanismus zur Übermittlung der neu erstellten Benachrichtigung aufruft.

subject
Gibt eine parametrisierte Textnachricht mit dem Betreff der Benachrichtigung für alle Ländereinstellungen an. Dieser Betreff wird im Posteingang des Benutzers für den Benachrichtigungsalert angezeigt. Detaillierte Informationen zu parametrisierten Nachrichten finden Sie in Kapitel 9, „Manuell", auf Seite 61.

body Gibt eine parametrisierte Textnachricht mit dem Text der Benachrichtigung für alle Ländereinstellungen an. Wenn der Benutzer im Posteingang auf den Betreff der Benachrichtigung klickt, wird dieser Text als der vollständige Text der Benachrichtigung angezeigt.

allocation-strategy
Gibt die Zuteilungsstrategie an, die zum Bestimmen der Bearbeiter verwendet wird, an die diese Benachrichtigung gesendet wird (siehe 14.3, „Zuteilungsstrategie für Benachrichtigungen", auf Seite 105).

actions
In der gleichen Art und Weise, wie bei einer manuellen Aktivität (siehe Kapitel 9, „Manuell", auf Seite 61) der Aufgabe Aktionen zugeordnet sein können, kann auch eine Benachrichtigung über zugeordnete Aktionen verfügen, die der benachrichtigte Benutzer ausführen kann. Diese Metadaten stellen die Details dieser Benachrichtigungsaktionen dar. Informationen zu Metadaten für Aktionen finden Sie im Abschnitt 9.3, „Aufgabendetails", auf Seite 61.

multiple-occurring-action
Gibt an, dass diese Benachrichtigungsaktion eine mehrfach auftretende Aktion darstellt. Wenn also diese Metadaten für eine Benachrichtigungsaktion angegeben werden, erstellt die Workflow-Engine beim Ausführen der Aktivität einen Datensatz der Aktion für jedes Element im Listen-Workflowdatenobjekt, das als die mehrfach auftretende Aktion festgelegt wurde.

Wenn eine Benachrichtigungsaktion als mehrfach auftretende Aktion verwendet wird, muss ein Attribut des verknüpften Listen-Workflowdatenobjekts als Linkparameter für die Benachrichtigungsaktion verwendet werden.

list-wdo-name
Der Name des Listen-Workflowdatenobjekts, das mit der mehrfach auftretenden Aktion verwendet wird.

14.2.2 Validierungen

- Für die Benachrichtigung muss ein Betreff definiert werden.
- Bei jedem Attribut des Workflowdatenobjekts, das dem Betreff einer Benachrichtigung zugeordnet ist, muss es sich um ein gültiges Attribut handeln, dass in der zugehörigen Prozessdefinition vorhanden ist.
- Wenn ein indexiertes Element aus einem Listen-Workflowdatenobjekt (d. h. 'CaseList[Context_Loop-.loopCount]') als Betrefftextparameter der Benachrichtigung verwendet wird, muss das Workflowdatenobjekt ein Listen-Workflowdatenobjekt sein und die Aktivität mit der Zuordnung muss in einer Schleife enthalten sein.
- Wenn das Workflowdatenobjekt `Context_Parallel` als Betrefftextparameter der Benachrichtigung verwendet wird, muss die Aktivität mit der Benachrichtigung eine Aktivität des Typs `Parallel` sein.
- Wenn ein indexiertes Element aus dem parallelen Listen-Workflowdatenobjekt als Betrefftextparameter der Benachrichtigung verwendet wird, muss die Aktivität mit der Zuordnung eine parallele Aktivität

sein (d. h. 'ParallelListWDO[Context_Parallel.occurrenceCount]'). Das anhand des Workflowdatenobjekts 'Context_Parallel' indexierte Workflowdatenobjekt muss ein Listen-Workflowdatenobjekt für eine parallele Aktivität sein.

- Es muss ein Benachrichtigungstext definiert sein.
- Bei jedem Attribut des Workflowdatenobjekts, das einem Benachrichtigungstext zugeordnet ist, muss es sich um ein gültiges Attribut handeln, dass in der zugehörigen Prozessdefinition vorhanden ist.
- Wenn ein indexiertes Element aus einem Listen-Workflowdatenobjekt (d. h. 'CaseList[Context_Loop-.loopCount]') als Textparameter der Benachrichtigung verwendet wird, muss das Workflowdatenobjekt ein Listen-Workflowdatenobjekt sein und die Aktivität mit der Zuordnung muss in einer Schleife enthalten sein.
- Wenn das Workflowdatenobjekt `Context_Parallel` als Textparameter der Benachrichtigung verwendet wird, muss die Aktivität mit der Benachrichtigung eine Aktivität des Typs `Parallel` sein.
- Wenn ein indexiertes Element aus dem parallelen Listen-Workflowdatenobjekt als Textparameter der Benachrichtigung verwendet wird, muss die Aktivität mit der Zuordnung eine parallele Aktivität sein (d. h. 'ParallelListWDO[Context_Parallel.occurrenceCount]'). Das anhand des Workflowdatenobjekts 'Context_Parallel' indexierte Workflowdatenobjekt muss ein Listen-Workflowdatenobjekt für eine parallele Aktivität sein.
- Für eine Aktivitätsbenachrichtigung muss eine Zuteilungsstrategie definiert werden.
- Wenn eine Funktion als die Zuteilungsstrategie für eine Benachrichtigung angegeben wurde, muss es sich um eine gültige Cúram-Geschäftsmethode handeln, die das Objekt `AllocationTargetList` zurückgibt.
- Bei einem Regelwerk, das als die Zuteilungsstrategie für eine Benachrichtigung angegeben wurde, muss es sich um ein gültiges Regelwerk für die Zuteilung handeln.
- Für eine Aktivitätsbenachrichtigung muss ein Bereitstellungsmechanismus definiert werden.
- Die Attribute des Workflowdatenobjekts, die den Aktionstext- und Aktionslinkparametern für eine Benachrichtigungsaktion zugeordnet sind, müssen in der zugehörigen Prozessdefinition vorhanden sein.
- Wenn ein indexiertes Element aus einem Listen-Workflowdatenobjekt (d. h. 'PersonDetailsList[Context-_Loop.loopCount]') als Zuordnung für die Aktionstext- oder Aktionslinkparameter der Benachrichtigung verwendet wird, muss das Workflowdatenobjekt ein Listen-Workflowdatenobjekt sein und die Aktivität mit der Zuordnung muss in einer Schleife enthalten sein.
- Wenn das Workflowdatenobjekt `Context_Parallel` als Zuordnung für die Aktionstext- oder Aktionslinkparameter der Benachrichtigung verwendet wird, muss die Aktivität mit der Benachrichtigung eine Aktivität des Typs `Parallel` sein.
- Wenn ein indexiertes Element aus dem parallelen Listen-Workflowdatenobjekt als Zuordnung für die Aktionstext- oder Aktionslinkparameter der Benachrichtigung verwendet wird, muss die Aktivität mit der Zuordnung eine parallele Aktivität sein (d. h. 'ParallelListWDO[Context_Parallel.occurrence-Count]'). Das anhand des Workflowdatenobjekts 'Context_Parallel' indexierte Workflowdatenobjekt muss ein Listen-Workflowdatenobjekt für eine parallele Aktivität sein.
- Die Anzahl von Platzhaltern, die im Betrefftext, im Aktionstext und im Haupttext der Benachrichtigung verwendet wird, muss der Anzahl der zugeordneten Attribute des Workflowdatenobjekts (für alle Ländereinstellungen) entsprechen.
- Das Workflowdatenobjekt, das für die Verwendung in der mehrfach auftretenden Aktion angegeben wurde, muss bezüglich der zugehörigen Workflowprozessdefinition ein gültiges Workflowdatenobjekt sein. Es muss zudem eine Liste sein.
- Mindestens ein Attribut des Workflowdatenobjekts für die mehrfach auftretende Aktion muss in den Linkparametern verwendet werden, die für die mehrfach auftretende Aktion angegebenen wurden.

14.2.3 Code

Für jede definierte Aktion muss sich die Aktionsseite auf eine gültige Cúram-Seite in der Anwendung beziehen, deren Seitenparameter vollständig mit den in den Benachrichtigungsmetadaten enthaltenen Aktionslinkparametern aufgefüllt werden.

Für die Anwendung wird die API `LocalizableStringResolver` bereitgestellt, mit deren Hilfe parametrisierte Nachrichtenzeichenfolgen aufgelöst werden können. Die Methoden in dieser API werden aufgelöst und geben die angegeben Nachricht für die erforderliche Ländereinstellung zurück. Zudem werden alle Workflowdatenobjekte, die in den Platzhaltern der Nachricht verwendet werden sollen, aufgelöst und sind in der zurückgegebenen Zeichenfolge enthalten.

Im Rahmen der API `LocalizableStringResolver` wird die Schnittstelle `NotificationStringResolver` zum Auflösen der mit Benachrichtigungen verknüpften parametrisierten Nachrichten verwendet. Der Betreff, der Hauptteil und der Aktionstext der Benachrichtigung können mithilfe der Methoden aus dieser API für die Verwendung in der Anwendung aufgelöst werden. Sobald die Workflow-Engine die zugehörige Bereitstellungsmethode in der Anwendung aufruft, sollte die Anwendung diese Methoden zur Verarbeitung der Benachrichtigung verwenden.

14.2.4 Laufzeitinformationen

Nachdem die Workflow-Engine die Ausführung einer Aktivität abgeschlossen hat, prüft sie, ob für diese Aktivität eine zugehörige Benachrichtigung definiert wurde. Ist dies der Fall, ermittelt die Engine anhand der implementierten Zuteilungsstrategie die zu benachrichtigenden Benutzer und ruft die Methode für die Benachrichtigungsbereitstellung in der Anwendung auf.

14.3 Zuteilungsstrategie für Benachrichtigungen

14.3.1 Voraussetzungen

Die Zuteilungsstrategie für Benachrichtigungen legt den oder die Benutzer fest, die beim Auftreten der zugehörigen Aktivität entsprechend benachrichtigt werden. Eine Zuteilungsstrategie für Benachrichtigungen wird in der gleichen Art und Weise definiert wie die Strategie für Aufgaben der manuellen Aktivität (siehe 9.4, „Zuteilungsstrategie", auf Seite 67).

14.3.2 Code

Die Anwendung muss die Rückrufschnittstelle `NotificationDelivery` implementieren, um festzulegen, wie Benachrichtigungen in der Anwendung verarbeitet werden.

Zur Verarbeitung der Benachrichtigung ruft die Workflow-Engine die Methode `deliverNotification` in der Implementierungsklasse `curam.util.workflow.impl.NotificationDelivery` auf. Die Engine leitet sowohl die von der Zuteilungsstrategie ermittelte Liste der Zuteilungsziele als auch die Details der erforderlichen Benachrichtigung an diese Anwendungsmethode weiter.

Die Anwendungseigenschaft 'curam.custom.notifications.notificationdelivery' gibt an, welche Implementierung der Schnittstelle `NotificationDelivery` von der Workflow-Engine zur Verarbeitung der Benachrichtigung verwendet wird.

Die Methode `deliverNotification` in dieser Standardimplementierungsklasse ist überladen. Der Grund dafür ist, dass die verschiedenen Zuteilungsstrategietypen die Zuteilungsziele in unterschiedlichen Formaten zurückgeben. Dies ist jedoch ein Aspekt der Implementierung, der für einen Entwickler von angepassten Klassen der Benachrichtigungsbereitstellung nicht von Bedeutung sein sollte, da die Geschäftsverarbeitung für alle Versionen der Methode gleich sein sollte.

```
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);

            ...
      }
```

Um diesem Problem entgegenzuwirken, stellt die Klasse `curam.core.sl.impl.DefaultNotificationDeliveryAdapter` einen weitaus effizienteren Mechanismus zur Implementierung einer Benarchrichtigungsbereitstellung bereit. Diese Klasse implementiert die unterschiedlichen Methoden und konvertiert deren Eingabeparameter in Zuteilungsziellisten. Dadurch können Entwickler von Logiken für die Benachrichtigungsbereitstellung diese Klasse erweitern und eine Methode implementieren, die unabhängig von der Quelle der Zuteilungsziele aufgerufen wird.

```
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);

            ...
      }
```

Neben dieser Adapterklasse ist im Lieferumfang dieser Anwendung auch eine sofort einsatzfähige Implementierung für eine Benachrichtigungsbereitstellung enthalten. Die Klasse heißt `curam.core.sl.impl.DefaultNotificationDelivery` und dient ebenfalls als Beispiel zur Erweiterung des Adapters.

Die Strategien für die Benachrichtigungsbereitstellung sind in der Codetabelle 'DELIVERYMECHANISM' aufgelistet. Das Hinzufügen einer neuen Strategie bedeutet eine einfache Erweiterung dieser Codetabelle mit einer neuen Strategie (beispielsweise SMS) und die Implementierung einer Bereitstellungsstrategie, die diesen Code erkennt und die entsprechende Logik ausführt. Da die Klasse für die Benachrichtigungsstrategie jedoch mithilfe einer einzigen Anwendungseigenschaft festgelegt wird, würde beim Ersetzen der Klasse `curam.core.sl.impl.DefaultNotificationDelivery` der sofort einsetzbare Bereitstellungsmechanismus inaktiviert werden. Wenn das Ziel ist, diesen Mechanismus zu erweitern (und nicht zu ersetzen), sollten benutzerdefinierte Klassen die Klasse `curam.core.sl.impl.DefaultNotificationDelivery` so erweitern, dass die ursprüngliche Funktionalität beibehalten wird. Die Klasse `curam.core.sl.impl.DefaultNotificationDelivery` wurde vor diesem Hintergrund implementiert.

```
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;
   }

   ...

 }
```

Die Klasse `curam.core.sl.impl.DefaultNotificationDelivery` implementiert die Methode `deliverNotification` aus dem abstrakten Adapter, weist jedoch sofort die entsprechende Kennung zur Verwendung einer geschützten Methode an. Die geschützte Methode `selectDeliveryMechanism` kann von Unterklassen zum Ermitteln von benutzerdefinierten Bereitstellungsmechanismen überschrieben werden und führt, wie nachfolgend dargestellt, die entsprechenden Operationen aus:

```
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);
   }
 }
```

Beachten Sie, dass die Methode `selectDeliveryMechanism` in der benutzerdefinierten Klasse zuerst eine Delegierung für die eigene übergeordnete Klasse vornimmt, bevor eigene Logiken ausgeführt werden. Durch die Erweiterung der Funktionalität in dieser Form können benutzerdefinierte Klassen die sofort einsatzfähigen Bereitstellungsmechanismen aufrufen, ohne die speziellen Codes kennen zu müssen, die von der übergeordneten Klasse erkannt werden. Dieser Ansatz ist auch bei Aktualisierungen von Vorteil. Wenn beispielsweise eine zukünftige Version von Cúram mehr sofort einsatzbereite Bereitstellungsmechanismen unterstützt, muss eine wie hier implementierte benutzerdefinierte Klasse nicht geändert werden, um die neue Funktionalität nutzen zu können.

Das von der oben dargestellten Funktion für die Benachrichtigungsbereitstellung zurückgegebene boolesche Flag zeigt der Workflow-Engine an, ob die Benachrichtigung an mindestens einen Benutzer im System gesendet wurde. Ist dies nicht der Fall, verfasst die Engine diesbezüglich einen Worfklowprüfdatensatz.

Kapitel 15. Übergänge

15.1 Übersicht

Übergänge stellen die Verknüpfungen zwischen Aktivitäten her. Sie sind das primäre Ablaufsteuerungskonstrukt und geben die Reihenfolge vor, in der Aktivitäten ausgeführt werden. Übergänge sind unidirektional, und eine Aktivität kann mehrere ausgehende und eingehende Übergänge haben, die Verzweigungs- (Splits) bzw. Synchronisationspunkte (Joins) bilden. Da jede Prozessdefinition eine Start- und eine Endaktivität aufweisen muss (siehe Kapitel 6, „Basisaktivität", auf Seite 31), kann eine Prozessdefinition auch als ein so genannter gerichteter Graph angesehen werden, in dem die Aktivitäten die Ecken (auch Vertices genannt) und die Übergänge die Bögen sind und jeder Weg von einer Startaktivität letztendlich zur Endaktivität führt.

15.2 Metadaten

© Copyright IBM Corp. 2012, 2013

```
<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

Eine Workflowprozessdefinition muss mindestens einen Übergang enthalten. Dieser Tag enthält die Details aller Übergänge zwischen den Aktivitäten in der angegebenen Workflowprozessdefinition.

transition

Dieser Tag enthält die Details eines Übergangs zwischen zwei Aktivitäten in der angegebenen Workflowprozessdefinition. Die obligatorischen Felder für einen Übergang sind nachfolgend aufgeführt:

id Gibt eine 64-Bit-Kennung an, die vom Cúram-Schlüsselserver bereitgestellt wird, wenn Übergänge im Prozessdefinitionstool erstellt werden. Die Übergangskennung muss inner-

halb einer Prozessdefinition eindeutig sein. Eine globale Eindeutigkeit in allen Prozessdefinitionen im System ist jedoch nicht erforderlich.

from-activity-idref
Gibt die `64-Bit`-Kennung der Quellenaktivität des Übergangs an.

to-activity-idref
Gibt die `64-Bit`-Kennung der Zielaktivität des Übergangs an.

condition
Übergänge können optional mit einer Bedingung verknüpft sein, um zu entscheiden, ob der angegeben Übergang verwendet wird oder nicht. Eine Bedingung ist eine Liste mit Ausdrücken, die logische Operationen für Workflowdatenobjekt-Attribute ausführen. Eine nähere Beschreibung zu Bedingungen finden Sie in Kapitel 16, „Bedingungen", auf Seite 113.

15.3 Validierungen

- Die für den Übergang definierte Quellenaktivität muss bezüglich der zugehörigen Workflowprozessdefinition eine gültige Aktivität sein.
- Die für den Übergang definierte Zielaktivität muss bezüglich der zugehörigen Workflowprozessdefinition eine gültige Aktivität sein.
- Die für einen Übergang definierten Quellen- und Zielaktivitäten dürfen nicht dieselben Aktivitäten sein.
- Die Startprozessaktivität in einer Workflowprozessdefinition darf keine eingehenden Übergänge enthalten.
- Die Endprozessaktivität in einer Workflowprozessdefinition darf keine ausgehenden Übergänge enthalten.
- Alle in der Workflowprozessdefinition definierten Aktivitäten (außer Endprozessaktivitäten) müssen mindestens einen eingehenden Übergang enthalten.
- Alle in der Workflowprozessdefinition definierten Aktivitäten (außer Startprozessaktivitäten) müssen mindestens einen ausgehenden Übergang enthalten.

15.4 Laufzeitinformationen

Für Aktivitäten, die im Rahmen der Anwendung durchgeführt werden (im Gegensatz beispielsweise zu Weiterleitungs- und Prozessaktivitäten, die nur von der Workflow-Engine durchgeführt werden), ist eine klare transaktionsorientierte Abgrenzung zwischen Engine- und Anwendungscode erforderlich. Es ist zudem sinnvoll, asynchrone Aufrufe zwischen der Workflow-Engine und der Anwendung zu haben (z. B. sollte ein Benutzer bei Übergängen zur nächsten Workflowaktivität nicht warten müssen, bevor ihm in der Benutzerschnittstelle die Kontrolle wieder zurückgegeben wird).

Aus diesem Grund sind drei spezielle Funktionen in einer Worfklowaktivität vorhanden: `start()`, `execute()` und `complete()`. Nach dem Abschluss einer Aktivität in der Workflowprozessinstanz ruft die Workflow-Engine zum Fortsetzen des Prozesses die entsprechende Funktion auf. Diese Funktion bewertet die ausgehenden Übergänge von der Aktivität, um den richtigen Weg auszuwählen.

Für jede Aktivität, die verarbeitet werden soll, wird die entsprechende `start()`-Funktion aufgerufen. Die entsprechenden Daten der Aktivitätsinstanz werden dann eingerichtet. Wenn die Aktivität direkt ohne JMS-Messaging (die Java Message Service-API (JMS) ist Bestandteil der Java EE) ausgeführt werden soll (eine Weiterleitungsaktivität wird immer direkt ausgeführt, da keine anwendungsbezogene Arbeit erforderlich ist), wird an dieser Stelle die `execute()`-Methode aufgerufen. Andernfalls wird zum Ausführen der angegebenen Aktivität (d. h. eine automatische Aktivität) eine JMS-Nachricht gesendet. Der Nachrichtenhandler des Workflows löst den Prozess und die in der Nachricht angegebene Aktivität auf und ruft für die Aktivität die `execute()`-Funktion auf.

Nachdem der Anwendungscode aufgerufen wurde, um die von der Aktivität angegebene Arbeit auszuführen, wird eine weitere Nachricht zum Abschließen der Aktivität gesendet. Der Nachrichtenhandler des Workflows löst den Prozess und die Aktivität in der Nachricht erneut auf und ruft die `complete`-Funktion für die Aktivität auf. Nachdem die Aktivität als abgeschlossen markiert wurde, wird die Funktion zum Fortsetzen des Prozesses erneut aufgerufen, um die zu verfolgenden Übergänge von der abgeschlossenen Aktivität aufzulösen. Dann wird der Prozess fortgesetzt.

Kapitel 16. Bedingungen

16.1 Übersicht

Die in Kapitel 15, „Übergänge", auf Seite 109 und Kapitel 12, „Schleifenbeginn und Schleifenende", auf Seite 91 beschriebenen Ablaufsteuerungskonstrukte benötigen oder unterstützen die Bewertung von Bedingungen, um den weiteren Verlauf für den Workflow festlegen zu können. Die Schleifenbeginnaktivitäten müssen bestimmte Metadaten aufweisen, die die Schleifenendbedingungen festlegen. Übergänge können optional mit einer Bedingung verknüpft sein, die festlegt, ob der angegeben Übergang verwendet wird oder nicht.

In diesem Kapitel wird das Prozessdefinitionsmetadatenkonstrukt beschrieben, das eine Bedingung darstellt. Eine Bedingung ist eine Liste mit Ausdrücken, die logische Operationen für Workflowdatenobjekt-Attribute ausführen. Bei der Bedingung selbst handelt es sich eine Verbindung, deren Wert sich aus der AND- oder OR-Verknüpfung der enthaltenen Ausdrücke ergibt. Die übergeordneten Konstrukte (Schleifen und Übergänge) initiieren als Folge der Bewertung von Bedingungen die entsprechenden Aktionen.

16.2 Metadaten

© Copyright IBM Corp. 2012, 2013

```
<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

Diese Metadaten sind für eine Schleifenbeginnaktivität obligatorisch (da für eine Schleife eine Exitbedingung angegeben sein muss), für einen Übergang jedoch optional (für einen Übergang muss nicht unbedingt eine Bedingung angegeben sein). Sie enthalten die Details von allen für die Bedingung definierten Ausdrücken.

expression

Enthält die Details eines in einer Bedingung enthaltenen Ausdrucks. Es können ein oder mehrere Ausdrücke für eine zugehörige Bedingung angegeben sein. Es können zwei Typen von Ausdrücken in einer Bedingung definiert sein. Es handelt sich einerseits um Funktionsausdrücke (die die beiden vordefinierten Funktionen `not()` und `isNothing()` verwenden) und andererseits um Datenelementausdrücke (bei denen der erstellte Bedingungsausdruck den ausgewählten Operator entweder auf zwei Workflowdatenobjekt-Attribute oder ein Workflowdatenobjekt-Attribut und eine Konstante anwendet). Ein Übergangsausdruck besteht aus den folgenden Attributen:

id
Gibt eine `64-Bit`-Kennung an, die vom Cúram-Schlüsselserver bereitgestellt wird, wenn Übergangsausdrücke im Prozessdefinitionstool erstellt werden. Die Ausdruckskennung muss innerhalb einer Prozessdefinition eindeutig sein. Eine globale Eindeutigkeit in allen Prozessdefinitionen im System ist jedoch nicht erforderlich.

data-item-rhs
Gibt den Namen des Datenelements an, das auf der rechten Seite des Bedingungsausdrucks verwendet werden soll. Im Fall eines Bedingungsausdrucks für ein Datenelement kann dieses Attribut ein Workflowdatenobjekt-Attribut (siehe Kapitel 4, „Workflowdatenobjekte", auf Seite 17 oder einen konstanten Wert darstellen, auf den der ausgewählte Operator angewendet wird. Für Bedingungsausdrücke für eine Funktion gibt es ein Workflowdatenobjekt-Attribut an, für das eine der beiden vordefinierten Funktionen zur Bewertung der Bedingung verwendet wird.

data-item-lhs
Dieser Metadaten-Tag ist optional, da er für einen Funktionsbedingungsausdruck nicht erforderlich ist. Im Fall eines Bedingungsausdrucks für ein Datenelement gibt er den Namen des Datenelements an, das auf der linken Seite der Bedingung verwendet werden soll (d. h. ein Workflowdatenobjekt-Attribut).

operation
Dieser Metadaten-Tag ist optional, da er für einen Funktionsbedingungsausdruck nicht erforderlich ist. Im Fall eines Bedingungsausdrucks für ein Datenelement gibt er eine Kennung für die logische Operation an, die entweder auf zwei Workflowdatenobjekt-Attribute oder ein Workflowdatenobjekt-Attribut und einen konstanten Wert angewendet wird. Nachfolgend ist eine Liste mit gültigen Operatoren aufgeführt, die in einem Bedingungsausdruck für ein Datenelement verwendet werden können:

Tabelle 4. Operatoren für Bedingungsausdruck

Operator	Erläuterung
==	gleich
!=	ungleich
<=	kleiner oder gleich
>=	größer oder gleich
<	kleiner als
>	größer als

conjunction
Gibt eine Kennung für eine logische Verknüpfung an, die entweder in einem Funktions- oder in einem Datenelementbedingungsausdruck verwendet werden kann. Es gibt zwei mögliche Werte für dieses Attribut: `and` (der Standardwert) und `or`. Wenn eine Bedingung aus mehreren Ausdrücken besteht, wird die logische Verknüpfung bei der Bewertung der vollständigen Bedingung verwendet.

function
Dieser Tag ist optional, da er nur bei Angabe eines Bedingungsausdrucks für eine Funktion verwendet wird. Wie bereits erwähnt, gibt es zwei vordefinierte Funktionen: `Not()`

und `isNothing()`. Die Funktion `Not()` fungiert als ein logischer Umkehrungsoperator. Unter normalen Umständen wird sie auf einen booleschen Wert angewendet. Die Funktion `isNothing()` kann auf alle Workflowdatenobjekt-Attributtypen (außer einen booleschen Wert) angewendet werden. Sie kann zum Testen von Szenarien verwendet werden, in denen erforderliche Daten nicht vorhanden sind oder nicht bereitgestellt wurden. Die Funktion gibt den booleschen Wert `true` zurück, wenn das untersuchte Workflowdatenobjekt-Attribut keine Daten enthält.

opening-brackets
Dieser Tag ist optional (der Standardwert ist 0), da er möglicherweise für keine der Bedingungsausdruckstypen angegeben wird. Er gibt die Anzahl der öffnenden Klammern an, die am Beginn des Ausdrucks eingefügt werden sollen.

closing-brackets
Dieser Tag ist optional (der Standardwert ist 0), da er möglicherweise für keine der Bedingungsausdruckstypen angegeben wird. Er gibt die Anzahl der schließenden Klammern an, die am Ende des Ausdrucks eingefügt werden sollen.

Die Anzahl der für einen einzelnen Ausdruck angegebenen öffnenden und schließenden Klammern muss nicht unbedingt gleich sein (es sei denn, es gibt nur einen Ausdruck in der Bedingung). Die Gesamtanzahl der öffnenden und schließenden Klammern in der gesamten Bedingung (mit allen enthaltenen Ausdrücken) muss jedoch gleich sein. Die Anzahl und die Position der öffnenden und schließenden Klammern in einem einzelnen Ausdruck und in der Bedingung als Ganzes sollten daher mit Bedacht angegeben werden, da anhand der Klammern festgelegt wird, die die Bedingung und die einzelnen Ausdrücke in dieser Bedingung bewertet werden. Das gleiche umsichtige Vorgehen empfiehlt sich auch bei der Angabe der Verknüpfung in der Bedingung, da es andernfalls zu nicht erwarteten Ergebnissen kommen kann.

16.3 Validierungen

- Das Attribut des Workflowdatenobjekts, das als das auf der rechten Seite des Bedingungsausdrucks befindliche Element angegeben ist, muss bezüglich der zugehörigen Workflowprozessdefinition ein gültiges Attribut sein.
- Das Attribut des Workflowdatenobjekts, das als das auf der linken Seite des Bedingungsausdrucks befindliche Element angegeben ist, muss bezüglich der zugehörigen Workflowprozessdefinition ein gültiges Attribut sein.
- Bei dem in einem Bedingungsausdruck des Datenelements eingegebenen Operator muss es sich um einen gültigen und unterstützten Operator handeln.
- Bei der in einem Funktionsbedingungsausdruck angegebenen Funktion muss es sich um eine gültige und unterstützte Funktion handeln.
- Bei der in einem Bedingungsausdruck angegebenen Verknüpfung muss es sich um eine gültige und unterstützte Verknüpfung handeln.
- Die Anzahl der öffnenden und schließenden Klammern muss im Kontext der gesamten Bedingung gleich sein.
- Wenn für einen Funktionsbedingungsausdruck die Funktion `Not()` angegeben ist, muss das auf der rechten Seite des Ausdrucks als Datenelement definierte Attribut des Workflowdatenobjekts den Typ BOOLEAN aufweisen.
- Wenn für einen Funktionsbedingungsausdruck die Funktion `isNothing()` angegeben ist, darf das auf der rechten Seite des Ausdrucks als Datenelement definierte Attribut des Workflowdatenobjekts nicht vom Typ BOOLEAN sein.
- Wenn es sich bei dem Datenelement auf der rechten Seite eines Bedingungsausdrucks des Datenelements um ein Attribut des Workflowdatenobjekts handelt, muss der Typ dieses Attributs mit dem entsprechenden Datenelement (Attribut des Workflowdatenobjekts) auf der linken Seite kompatibel sein.

Wenn entsprechend das Datenelement auf der rechten Seite als ein konstanter Wert angegeben wurde, muss es mit dem Typ des entsprechenden Datenelements (Attribut des Workflowdatenobjekts) auf der linken Seite kompatibel sein.

- Wenn entweder die rechte oder die linke Seite eines Übergangsbedingungsausdrucks ein indexiertes Element aus einem Listen-Workflowdatenobjekt enthält (z. B. 'ChildDependents[Context_Loop.loop-Count].age'), dann muss das zugehörige Workflowdatenobjekt ein Listen-Workflowdatenobjekt sein und die im Übergang verwendeten Aktivitäten müssen in einer Schleife enthalten sein.
- Wenn in einem Schleifenbedingungsausdruck entweder auf der rechten oder auf der linken Seite des Ausdrucks das Attribut *size()* für ein Workflowdatenobjekt angegeben ist, muss es sich bei diesem Workflowdatenobjekt um ein Listen-Workflowdatenobjekt handeln.
- Wenn in einem Schleifenbedingungsausdruck entweder auf der rechten oder auf der linken Seite des Ausdrucks das Attribut *size()* für ein Workflowdatenobjekt angegeben ist, muss das Element auf der anderen Seite des Ausdrucks dem Typ INTEGER zugeordnet werden können.
- Wenn in einem Schleifenbedingungsausdruck entweder auf der rechten oder auf der linken Seite des Ausdrucks das Attribut *isEmpty()* für ein Workflowdatenobjekt angegeben ist, muss es sich bei diesem Workflowdatenobjekt um ein Listen-Workflowdatenobjekt handeln.
- Wenn in einem Schleifenbedingungsausdruck entweder auf der rechten oder auf der linken Seite des Ausdrucks das Attribut *isEmpty()* für ein Workflowdatenobjekt angegeben ist, muss das Element auf der anderen Seite des Ausdrucks dem Typ BOOLEAN zugeordnet werden können.

Kapitel 17. Splits/Joins (Verzweigungs-/ Synchronisationspunkte)

17.1 Einführung

Übergänge verknüpfen Aktivitäten in einer Prozessdefinition. In der einfachsten Konfiguration von Aktivitäten und Übergängen weist jede Aktivität nur einen eingehenden und einen ausgehenden Übergang auf. Es ist jedoch häufig sinnvoll, mehr als einem Weg aus einer Aktivität in Form einer Verzweigung (d. h. mehrere Übergänge, die aus einer Aktivität abgehen) zu folgen. Um eine gültige Blockstruktur in einer Prozessdefinition zu unterstützen (siehe Kapitel 18, „Workflowstruktur", auf Seite 121), muss jede Verzweigung (Split) einen entsprechenden Synchronisationspunkt (Join) aufweisen (d. h. mehrere Übergänge, die an einer Aktivität zusammenlaufen). Für gewöhnlich können bei einem Split mehrere 'Arbeitsgänge' gleichzeitig durchgeführt werden, während ein Join der reziproke Synchronisationspunkt für diese 'Arbeitsgänge' ist.

Es gibt zwei Gründe, warum eine Aktivität einen Split (und daher weiter hinten im Ablauf eine andere Aktivität einen Join) aufweisen sollte. Der erste Grund ist, dass Arbeiten, die keinen Abhängigkeiten unterliegen, parallel durchgeführt werden können. Der andere Grund ist, dass zwischen einer Reihe verschiedener Wege im Workflow ausgewählt werden kann.

Auf Metadatenebene weist jede Aktivität einen Split- und einen Join-Typ auf. Wenn eine Aktivität nur einen ausgehenden oder eingehenden Übergang hat, wird dem Split oder entsprechend dem Join der Typ `none` zugewiesen. Die anderen beiden Split- und Join-Typen `Auswahl` (auch als XOR bezeichnet) und `Parallel` (auch als AND bezeichnet) werden in diesem Kapitel beschrieben.

17.2 Split des Typs 'Auswahl' (XOR)

17.2.1 Metadaten

```
<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 Diesen Tag gibt es für jede Aktivität. Er enthält die Details des Splits (der Verzweigung) von der Aktivität. Dazu zählt eine Liste der Übergänge von der angegebenen Aktivität, die von der Workflow-Engine aufgelöst werden, wenn die zugehörige Aktivität abgeschlossen ist, um zu untersuchen, ob die Übergänge verfolgt werden können oder nicht.

Die Reihenfolge der Übergänge in dieser Liste ist wichtig für den Split-Typ `XOR`, da der erste zulässige Übergang in der geordneten Liste von der Workflow-Engine verarbeitet wird. Wenn im

© Copyright IBM Corp. 2012, 2013

obigen Metadatenbeispiel die Übergangsbedingungen für die Übergangskennungen 2, 3 und 4 erfüllt werden, wird der Übergang mit der Kennung 2 verfolgt, da es sich hierbei um den ersten zulässigen Übergang in der Liste mit den geordneten Übergängen handelt.

type Gibt den Split-Typ an. Wie bereits zuvor erwähnt gibt es drei mögliche Split-Typen. Der Split-Typ `NONE` zeigt an, dass es nur einen ausgehenden Übergang von der angegebenen Aktivität gibt. Der Split-Typ `XOR` steht für eine Auswahl, was bedeutet, dass der erste zulässige Übergang aus der Liste der geordneten Übergänge verfolgt wird. Der Split-Typ `AND` zeigt einen parallelen Ausführungspfad an, der sicherstellt, dass alle zulässigen Übergänge, die in der Liste der Übergänge aufgeführt sind, parallel verfolgt werden.

transition-id

Enthält einen Verweis auf den angegeben Übergang. Wenn der Split-Typ `XOR` oder `AND` ist, gibt es mehrere Einträge dieses Metadaten-Tags.

idref Enthält einen Verweis auf einen Übergang in der Workflowprozessdefinition.

17.3 Split des Typs 'Parallel' (AND)

17.3.1 Metadaten

```
<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>
```

Die Metadaten für den Split-Typ `AND` ähneln denen des Split-Typs `XOR` (siehe 17.2, „Split des Typs 'Auswahl' (XOR)", auf Seite 119). Der Unterschied ist, dass der Typ des Splits als `AND` angegeben ist. Dadurch wird sichergestellt, dass beim Ermitteln der Liste der Übergänge, die von der angegebenen Aktivität aus verfolgt werden sollen, die Reihenfolge der Übergänge in der Liste nicht wichtig ist, da alle zulässigen Übergänge in einem Split des Typs `AND` verfolgt werden. Die geordnete Liste der Übergänge wird in dieser Instanz für diesen Split-Typ gepflegt und beibehalten, um eine Änderung vom Split-Typ `AND` in einen Split des Typs `XOR` zu vereinfachen, bei dem die Reihenfolge der Übergänge wieder wichtig wird.

Kapitel 18. Workflowstruktur

18.1 Übersicht

Die Struktur eines Workflowprozesses basiert auf den Aktivitäten im Prozess und den Übergänge zwischen diesen Aktivitäten. Ein Workflow bildet daher einen Graph, in dem die Aktivitäten die Ecken und die Übergänge die Bögen sind (der von einem Workflow gebildete Graph kann mithilfe der Funktion zur bildlichen Darstellung des Workflowprozesses im Prozessdefinitionstool angezeigt werden).

Damit die Workflow-Engine einen Prozess erfolgreich interpretieren und ausführen kann, muss der von diesem Prozess gebildete Graph bestimmte Kriterien erfüllen. In diesem Kapitel werden diese Kriterien unter zwei Hauptüberschriften dargestellt: Graphenstruktur und Blockstruktur.

18.2 Graphenstrukur

Da eine Gruppe von Aktivitäten und Übergängen in einem Prozess einen Graph bilden, können anhand der Graphentheorie einige bekannte strukturelle Probleme schon frühzeitig vor dem ersten Ausführen eines Prozesses erkannt werden.

Graphentheorie: Die Graphentheorie ist ein Teilgebiet der Mathematik. Glücklicherweise sind die Bereiche der Graphentheorie, die für einen Workflow einen Rolle spielen, sehr einfach. Daher erfordert dieses Kapitel keinerlei Vorkenntnisse zur Graphentheorie und auch keine speziellen Mathematikkenntnisse. Im Internet gibt es eine Vielzahl weiterführender Informationen zur Graphentheorie.

Nehmen Sie beispielsweise einen Prozess an, in dem eine Aktivität einen Übergang zu einer anderen Aktivität aufweist, die wiederum einen Übergang zurück zur ersten Aktivität hat. Dadurch entsteht ein Zyklus im Prozessgraph.

Würden für die Übergänge keine Bedingungen vorhanden sein, würde der Prozess auf jeden Fall in einer unendlichen Schleife enden. Solche Schleifen werden als nicht formale Schleifen (oder auch Ad-hoc-Schleifen) bezeichnet. Sind sie in einem Prozess vorhanden, sind einige nützliche strukturelle Validierungen nicht möglich. Unter anderem aus diesem Grund stehen in Cúram Workflow formale Konstrukte bereit, um iterative Abschnitte eines Prozesses zu definieren (die Schleifenbeginn- und Schleifenendaktivitäten). Dadurch können Ad-hoc-Schleifen in Prozessen erkannt und als Ergebnis nicht freigegeben werden.

Codeähnlichkeiten: Viele Entwickler sind mit der Anweisung GOTO und daher mit den geschweiften Klammern vertraut, die normalerweise verwendet werden, um den Beginn ({) und das Ende (}) einer formalen Schleife anzuzeigen.

GOTO entspricht Ad-hoc-Schleifen in einem Workflow. Die geschweiften Klammern sind analog zu den formalen Schleifenbeginn- und Schleifenendaktivitäten in einem Workflow.

18.3 Blockstruktur

Es gibt verschiedene Workflowelemente, die sich auf die Auswahl des Durchlaufwegs (oder der Wege) durch einen Workflow zur Laufzeit auswirken können. Zu diesen zählen:

- 17.2, „Split des Typs 'Auswahl' (XOR)", auf Seite 119
- 17.3, „Split des Typs 'Parallel' (AND)", auf Seite 120
- Kapitel 12, „Schleifenbeginn und Schleifenende", auf Seite 91

© Copyright IBM Corp. 2012, 2013

Diese Elemente sind immer paarweise angeordnet. Der Grund dafür ist, dass sie Bereiche abgrenzen, in denen der Prozess ein bestimmtes Verhalten aufweisen sollte (ein Verhalten, das sich auf den Steuerungsfluss bezieht). Diese Bereiche werden normalerweise als 'Blöcke' bezeichnet, da sie einen bestimmten Startpunkt haben, der einen entsprechenden Endpunkt aufweisen muss.

Angenommen, ein Prozess verfügt über eine Struktur, bei der alle Wege, die von einem Split des Typs 'Auswahl' (bei dem nur ein ausgehender Pfad verfolgt wird) ausgehen, an einem parallelen Join (der wartet, bis alle eingehenden Pfade 'angekommen' sind, bevor die nächste Aktivität ausgeführt wird) zusammenlaufen. In diesem Fall wird der Prozess auf jeden Fall am parallelen Join angehalten. Dies ist ein Beispiel für ein Problem mit der Blockstruktur, das durch Validierungen bereits vor dem Ausführen eines Prozesses erkannt werden kann.

18.3.1 Analogie für Blöcke

Ein gutes Beispiel dafür, wie Blöcke in einem Workflow funktionieren, ist die Art und Weise, wie Klammern (wie diese!) in einem Satz funktionieren. Klammern haben den expliziten Startpunkt '(', der als Gegenstück immer einen bestimmten Endpunkt ')' hat. Diese Punkte markieren den Bereich eines Satzes, der eine spezielle Bedeutung hat.

Die Art und Weise, wie Klammern in mathematischen Ausdrücken verwendet werden, ist vielleicht ein noch besserer Vergleich. Zusätzlich zu den entsprechenden öffnenden und schließenden Klammern kann ein mathematischer Ausdruck mehrere Arten von Klammern verwenden. Ausdrücke mit Klammern können ineinander verschachtelt sein, jedoch nicht ineinandergreifen. Auf sehr ähnliche Weise werden Blöcke in einem Workflow verwendet.

18.3.2 Vom Workflow unterstützte Blocktypen

In den folgenden Abschnitten werden die verschiedenen Blocktypen in Cúram Workflow beschrieben. Außerdem werden der Anfang und das Ende solcher Blöcke sowie ihr Verwendungszweck erläutert.

18.3.2.1 Block des Typs 'Auswahl' (XOR)

Ein *Block des Typs 'Auswahl'* beginnt an einem Split des Typs 'Auswahl' (XOR) und endet an einem Join des Typs 'Auswahl' (XOR) (den 'Klammern'). Dieser Typ gibt an, dass von allen möglichen Wegen innerhalb dieses Blocks höchstens *einer* verfolgt werden kann.

Der Split weist mehrere ausgehende Übergänge auf, die die möglichen Wege für eine Prozessinstanz anzeigen. Da es sich hierbei um einen Block des Typs 'Auswahl' handelt, schließen sich die Wege gegenseitig aus: Nur einer wird von einer bestimmten Prozessinstanz eingeschlagen.

Der Split des Typs 'Auswahl' muss einen entsprechenden Join des Typs 'Auswahl' aufweisen. Dieser Join zeigt den Punkt an, an dem der Prozess nicht mehr für jeden Weg anders ist und an dem die Wege wieder zusammengeführt werden (d. h. der restliche Prozess ist einheitlich).

18.3.2.2 Block des Typs 'Parallel' (AND)

Ein *Block des Typs 'Parallel'* beginnt an einem Split des Typs 'Parallel' (AND) und endet an einem Join des Typs 'Parallel' (AND) (den 'Klammern'). Dieser Typ gibt an, dass von allen möglichen Wegen innerhalb dieses Blocks *viele oder alle* verfolgt werden können.

Der Split weist mehrere ausgehende Übergänge auf, die die möglichen Wege für eine Prozessinstanz anzeigen. Da es sich um einen Block des Typs 'Parallel' handelt, können beliebig viele der Wege parallel verfolgt werden (unter der Voraussetzung, dass ihre Übergangsbedingungen erfüllt sind).

Der Split des Typs 'Parallel' muss einen entsprechenden Join des Typs 'Parallel' aufweisen. Der Join ist der Punkt, an dem alle parallelen Wege synchronisiert werden müssen, bevor der Workflow weiter fortgesetzt werden kann.

18.3.2.3 Block des Typs 'Schleife'

Ein *Block des Typs 'Schleife'* beginnt mit einer Schleifenbeginnaktivität und endet mit einer Schleifenendaktivität (den 'Klammern'). Er zeigt an, dass der Abschnitt, der von den Schleifenbeginn- und den Schleifenendaktivitäten begrenzt wird, immer wiederholt werden sollte, so lange die Schleifenbedingung erfüllt wird.

Die Schleifenbeginnaktivität markiert den Punkt, an dem die Ausführung wieder starten sollte, wenn die Schleifenbedingung erfüllt wird (d. h. der Rückkehrpunkt, falls die Engine festlegt, dass die Schleife iteriert werden soll). Die Schleifenendaktivität markiert die Stelle, an der die Ausführung fortgesetzt werden soll, wenn die Schleifenbedingung *nicht* erfüllt wird.

18.4 Strukturelle Regeln

Es gibt gewisse strukturelle Regeln, die Ersteller des Workflows beim Festlegen von Workflowdefinitionen beachten sollten. Wenn ein Cúram-Workflowprozess geprüft wird, wird im Rahmen der Validierungen bewertet, ob die Struktur des Prozesses mit diesen Regeln konform ist. Wie bei allen Validierungen soll mit dieser Prüfung sichergestellt werden, dass der Prozess von der Workflow-Engine ausgeführt werden kann.

18.4.1 Regeln für die Graphenstruktur

Ein Cúram-Prozess muss einen gerichteten, zusammenhängenden und azyklischen Graph bilden. Dies klingt möglicherweise kompliziert, jedoch sind diese Begriffe nur technische Begriffe für einige sehr einfache Grapheigenschaften.

- Ein "gerichteter" Graph ist ein Graph, in dem jede Kante nur in eine Richtung verläuft. Ein gerichteter Graph wird normalerweise als Digraph bezeichnet. In Bezug auf einen Workflow bedeutet das, dass ein Übergang von Aktivität A nach Aktivität B nicht verwendet werden kann, um wieder von B zurück nach A zu gelangen. Dies wird in Cúram Workflow als gegeben vorausgesetzt. Es wird an dieser Stelle erwähnt, da die 'azyklische' Eigenschaft (siehe unten) für Graphen und Diagraphen auf unterschiedliche Art und Weise definiert wird.
- Ein 'zusammenhängender' Graph ist ein Graph, in dem jede Ecke erreicht werden kann. In Bezug auf einen Workflow bedeutet das, dass jede Aktivität im Prozess auf mindestens einem Weg von der Startaktivität zur Endaktivität erreichbar sein muss.

  Dadurch wird verhindert, dass ein Workflow eine Struktur aufweist, bei der mindestens eine Aktivität niemals ausgeführt werden kann.
- Ein 'azyklischer' Digraph ist ein Graph, in dem es keine *gerichteten* Zyklen gibt. Für einen Workflow bedeutet das, dass es keine Ad-hoc-Schleifen geben kann (d. h. Schleifen, die anstelle von Schleifenbeginn- und Schleifenendaktivitäten Übergänge verwenden).

  Ad-hoc-Schleifen mögen zwar auf den ersten Blick praktisch in ihrer Anwendung erscheinen, sie können jedoch (wie Anweisungen des Typs GOTO in Programmiersprachen) dazu führen, dass ein Prozess schwer lesbar und somit schwer zu verstehen ist. Die Verwendung von expliziten Schleifenkonstrukten führt zu klareren und verständlicheren Prozessdefinitionen.

  Darüber hinaus weiß die Engine, an welchen Stellen Schleifen auftauchen können. Dadurch kann sie überwachen, wie häufig eine Schleife zur Laufzeit iteriert wurde.

18.4.2 Regeln für die Blockstruktur

Wie bereits erwähnt entspricht die Verwendungsweise von Klammern in mathematischen Ausdrücken in etwa der Art und Weise, wie "Blöcke" in einem Workflow verwendet werden. Zur Erinnerung: Es gibt die Blocktypn 'Auswahl', 'Parallel' und 'Schleife'. In Cúram Workflow bedeutet dies Folgendes:

- Alle einen Block startende Konstrukte (Split des Typs 'Auswahl', Split des Typs 'Parallel' oder Schleifenbeginnaktivität) müssen mit einem entsprechenden, einen Block beendenden Konstrukt (Join des Typs 'Auswahl', Join des Typs 'Parallel' und Schleifenendaktivität) beendet werden.

  Im Fall von Splits und Joins müssen alle von einem Split abgehenden Wege wieder am entsprechenden Join zusammentreffen.

Begründung: Durch die erforderliche Übereinstimmung von Splits und Joins (als Beispiel) wird die Lesbarkeit eines Workflows verbessert. In einem Abschnitt mit mehreren Pfaden wird dadurch verdeutlicht, ob ein einzelner Weg (oder mehrere Wege) verfolgt werden können. Dies wiederum verdeutlicht, ob an dem Punkt, wo die Wege wieder zusammenlaufen, eine Synchronisation erforderlich ist oder nicht.

Wäre es nicht erforderlich, dass die Splits und Joins übereinstimmen müssen, wäre es sehr leicht, Prozesse zu modellieren, die unterbrochen werden können oder deren Ende erreicht werden könnte, bevor einige Aktivitäten erfolgreich abgeschlossen wurden.

- Blöcke können ineinander verschachtelt sein (beispielsweise ein Split des Typs 'Auswahl' in einer Schleife), sie können jedoch nicht ineinandergreifen (beispielsweise kann kein Übergang von einem Split des Typs 'Auswahl' zu einer Aktivität außerhalb der Schleife führen).

  Dadurch können Situationen vermieden werden, die für die Engine schwierig zu verarbeiten und für Workflowentwickler nicht intuitiv genug sind.

  Angenommen, eine Schleife enthält einen Join, wobei der Join über zwei eingehende Übergänge verfügt: einen von einer Aktivität innerhalb der Schleife und einen von einer Aktivität außerhalb der Schleife.

  In dieser Situation ist es sehr schwierig, ein optimales Vorgehen für die Join-Synchronisation festzulegen. Ein eingehender Übergang kann nur einmal ausgelöst werden, der andere mehrmals. Alle Regeln für die Handhabung dieser Situation wären willkürlich und demnach nicht intuitiv.

  Workflows, für die Blöcke des Typs 'Auswahl', 'Parallel' und 'Schleife' definiert wurden, haben eine einfache klare Struktur, deren Bedeutung sofort erkennbar ist.

18.5 Validierungen

Ein gültiger Cúram-Workflow muss einen gerichteten zusammenhängenden azyklischen Graph mit Blockstruktur bilden. Diese Eigenschaften sind überwiegend diskret, sodass sie vor dem Freigeben eines Prozesses unabhängig mit dem Prozessdefinitionstool überprüft werden können. Die für eine Prozessdefinition durchgeführten strukturellen Validierungen laufen in einer bestimmten Reihenfolge ab, die nachfolgend beschrieben ist.

18.5.1 Einfache syntaktische Prüfungen

Die erste Gruppe struktureller Validierungen, die ausgeführt werden, sind so genannte *einfache syntaktische* Prüfungen. Mit diesen Prüfungen wird sichergestellt, dass die Joins und Splits von Aktivitäten (siehe Kapitel 17, „Splits/Joins (Verzweigungs-/Synchronisationspunkte)", auf Seite 119) in der Prozessdefinition ordnungsgemäß eingerichtet sind. Zu den Validierungen zählen:

- Alle Aktivitäten (außer die *Startaktivitäten* und *Endaktivitäten*) müssen mindestens einen eingehenden und einen ausgehenden Übergang aufweisen.
- Für alle Aktivitäten, die mehr als einen eingehenden Übergang aufweisen, *muss* ein Join-Typ angegeben sein (d. h. ein anderer Join-Typ als *NONE*).
- Für alle Aktivitäten, die mehr als einen ausgehenden Übergang aufweisen, *muss* ein Split-Typ angegeben sein (d. h. ein anderer Split-Typ als *NONE*).
- Alle Aktivitäten mit *genau* einem eingehenden Übergang müssen den Join-Typ *NONE* aufweisen.
- Alle Aktivitäten mit *genau* einem ausgehenden Übergang müssen den Split-Typ *NONE* aufweisen.
- Der Split-Typ für eine `parallele` Aktivität muss *NONE* sein.
- Der Join-Typ für eine `parallele` Aktivität muss *NONE* sein.
- Eine `parallele` Aktivität darf *genau* einen eingehenden Übergang aufweisen.
- Eine `parallele` Aktivität darf *genau* einen ausgehenden Übergang aufweisen.
- Der Split-Typ der Aktivität auf der anderen Seite des eingehenden Übergangs zu einer `parallelen` Aktivität *muss* immer *NONE* sein.

- Der Join-Typ der Aktivität auf der anderen Seite des ausgehenden Übergangs von einer `parallelen` Aktivität *muss* immer *NONE* sein.

18.5.2 Graphprüfungen

Die zweite Gruppe struktureller Validierungen, die ausgeführt werden, sind so genannte *Graphprüfungen*. Mit diesen Prüfungen wird sichergestellt, dass der Flussgraph ein gerichteter zusammenhängender azyklischer Graph ist. Zu den Validierungen zählen:

- Der Workflow muss ein 'zusammenhängender' Graph sein. Dies bedeutet, dass jede Aktivität auf mindestens einem Weg von der *Startaktivität* zur *Endaktivität* erreichbar sein *muss*.
- Der Workflow muss einen azyklischen Digraph bilden. Dies bedeutet, dass es *keinen* Weg durch den Workflow geben darf, der zweimal auf dieselbe Aktivität trifft. Bei dieser Validierung wird nach Zyklen gesucht, die nur von Übergängen erstellt wurden. Zyklen, die mit Schleifenbeginn- und Schleifenendaktivitäten erstellt wurden, sind in jeder Hinsicht gültig.
- Jeder Untergraph der Instanz innerhalb des Workflow-Graphs muss richtig beendet werden. Das bedeutet, dass beim Starten von der *Startaktivität* alle möglichen Wege durch den Workflow an der *Endaktivität* enden müssen.

18.5.3 Blockprüfungen

Die dritte Gruppe struktureller Validierungen, die ausgeführt werden, sind so genannte *Blockprüfungen*. Mit diesen Validierungen wird sichergestellt, dass der Flussgraph eine korrekte Blockstruktur aufweist.

Die Blockstart-Konstrukte sind Startprozessaktivität, Schleifenbeginnaktivität, Split-Typ 'Parallel' (AND) und Split-Typ 'Auswahl' (XOR). Die entsprechenden Blockende-Konstrukte sind Endprozessaktivität, Schleifenendaktivität, Join-Typ 'Parallel' (AND) und Join-Typ 'Auswahl' (XOR).

Basierend auf diesen Konstrukten werden die folgenden Validierungen für die Blockstruktur durchgeführt:

- Für jeden Blockstart muss ein entsprechendes Blockende vorhanden sein (d. h., wenn es in der Aktivität eine Schleifenbeginnaktivität gibt, muss es eine entsprechende Schleifenendaktivität geben).
- Die Typen für Blockstart/Blockende muss übereinstimmen (d. h., wenn es einen Split des Typs 'Parallel' (AND) im Workflow-Graph gibt, muss dieser ein entsprechenden Join-Typ des Typs 'Parallel' (AND) haben).
- Blöcke können zwar verschachtelt werden, nicht aber ineinandergreifen.

Kapitel 19. Workflow-Web-Services

19.1 Übersicht

Cúram-Workflows sind durch die Unterstützung für bestimmte Aspekte des BPEL-Standards (*Business Process Execution Language*) der Oasis-Gruppe mit anderen Workflowsystemen kompatibel. BPEL-Prozesse können Cúram-Workflowprozesse umsetzen und benachrichtigt werden, wenn der Prozess beendet wird.

Die Cúram-Workflow-Engine ist nicht als umfassende BPEL-Orchestrierungs-Engine vorgesehen. Stattdessen sollten die Cúram-Workflows Bestandteil von orchestrierten BPEL-Prozessen sein können. Dies wird mit der Funktion ermöglicht, Cúram-Workflowprozesse als Web-Services verfügbar zu machen, die über BPEL-Prozesspartnerlinks aufgerufen werden können.

19.2 Verfügbarmachen eines Workflow-Web-Service

Workflow-Web-Services bauen auf der vorhandenen Unterstützung für Cúram-Web-Services auf. Die Workflow-Engine benötigt hierbei insbesondere ein Geschäftsprozessobjekt (GPO), das als dokumentorientierter Web-Service modelliert ist (Informationen hierzu finden Sie im Kapitel *Cúram Inbound Web Services* des Handbuchs *Cúram Modeling Reference Guide*).

Bei dem Web-Service-GPO handelt sich nur um das Front-End für die Workflowumsetzungs-API (`curam.util.workflow.impl.EnactmentService`). Vor diesem Hintergrund ist nur ein solches GPO pro Anwendung erforderlich. Ein entsprechendes GPO wird bereits in der Cúram-Anwendung bereitgestellt: `Logical View::MetaModel::Curam::Facades:: Workflow::WebService::WorkflowProcessEnactment`.

Um die Workflow-Web-Services verwenden zu können, muss dem BPO `Logical View::MetaModel::Curam::Facades:: Workflow::WebService::WorkflowProcessEnactment` eine Serverkomponente des Typs Web-Service zugewiesen sein.

Cúram-Web-Services können auch anders angepasst werden. Sie können beispielsweise mithilfe von SW-Security sicher gemacht werden (siehe dazu die Informationen im Kapitel *Secure Web Services* des Handbuchs *Cúram Modelling Reference Guide*). Alle Anpassungen für Workflow-Web-Services müssen an diesem GPO vorgenommen werden.

Anmerkung: Da alle Workflow-Web-Services vom selben GPO ausgeführt werden, wirken sich jegliche Anpassungen auf alle Prozessdefinitionen aus, die als Web-Services verfügbar gemacht werden.

19.2.1 Prozessumsetzung

Um eine Cúram-Workflowprozessdefinition als Web-Service verfügbar zu machen, muss diese ganz einfach als solcher entweder im Prozessdefinitionstool (PDT) oder direkt in den Metadaten festgelegt werden (siehe Kapitel 3, „Metadaten der Prozessdefinition", auf Seite 13. Sobald die Prozessdefinitionen als Web-Services markiert wurden, müssen der Server, die Server-EAR-Datei und die Web-Services-EAR-Datei neu erstellt werden.

Wie auch für andere Cúram-Web-Services kann auf die Web Services Description Language (WSDL) für den Service erst dann zugegriffen werden, wenn die Web-Services-EAR-Datei bereitgestellt wurde. Der Name des Workflow-Web-Service entspricht dem Prozessnamen. Dementsprechend kann auf die WSDL über ein URL wie etwa die folgende URL zugegriffen werden: http://testserver:9082/CuramWS/services/<ProcessName>?wsdl.

© Copyright IBM Corp. 2012, 2013

Der Inhalt der WSDL wird teilweise durch die Eingabe für den Prozess (die Workflowdatenobjekt-Attribute, die als erforderlich für die Umsetzung markiert sind) und die Ausgabe für den Prozess (die Workflowdatenobjekt-Attribute, die als Prozessausgabe markiert sind) bestimmt (siehe 4.2, „Metadaten", auf Seite 18). Der WSDL-Porttyp ist der Prozessname und die Operation zur Umsetzung des Prozesses lautet immer `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>
```

Abbildung 2. Porttyp für die Prozessumsetzung

19.2.2 Rückruf-Web-Service für Prozessabschluss

Ein externes System (aber nicht notwendigerweise ein BPEL-Prozess), das einen Cúram-Workflow über Web-Services umsetzt, muss unter Umständen benachrichtigt werden, dass der Prozess abgeschlossen wurde. Des Weiteren benötigt es einige Ausgabedaten von der Prozessdefinition. Dafür muss für jede Prozessdefinition ein Web-Service angegeben werden, der aufgerufen wird, wenn der Prozess beendet wird.

Der Rückruf-Web-Service wird in den Metadaten der Prozessdefinition mithilfe des Prozessdefinitionstools oder direkt in den Metadaten (siehe Kapitel 3, „Metadaten der Prozessdefinition", auf Seite 13) angegeben.

Anmerkung: Bevor er in einer Workflowprozessdefinition verwendet werden kann, muss der Rückruf-Web-Service als ausgehender Cúram-Web-Service-Connector registriert werden, wie im Kapitel *Cúram Outbound Web Service Connectors* im Handbuch *Cúram Modeling Reference Guide* beschrieben.

Der Rückruf-Web-Service muss von einem externen System implementiert werden, muss jedoch einer Porttypdefinition entsprechen, die vom Cúram-Workflow-Web-Service festgelegt wurde. Weitere Informationen hierzu finden Sie im Abschnitt 19.3, „Aufruf von BPEL-Prozessen".

19.3 Aufruf von BPEL-Prozessen

Die Erstellung von BPEL-Prozessen, die Cúram-Workflowprozesse umsetzen, wird in diesem Dokument nicht näher erläutert. Die Web Services Description Language (WSDL) für jeden Workflow-Web-Service enthält jedoch Informationen, die von BPEL-Prozessen verwendet werden können.

Porttyp für den Rückruf

Es gibt in WSDL einen Porttyp für einen Cúram-Workflow-Web-Service, der nicht vom Service selbst implementiert wird. Der Name dieses Porttyps ist der Name des Prozesses, an den das Wort "Complete" angehängt ist (`<Prozessname>Complete`).

Zweck dieses nicht implementierten Porttyps ist, die Web-Service-Schnittstelle zu definieren, von der ein Cúram-Workflow-Web-Service erwartet, dass sie von dem BPEL-Prozess implementiert wird, der sie umgesetzt hat. Dieser Porttyp muss vom Rückruf-Web-Service implementiert werden, der in der Prozessdefinition konfiguriert ist (siehe 19.2.2, „Rückruf-Web-Service für Prozessabschluss").

```
<!--Implemented by the BPEL process-->
  <wsdl:portType name="SomeCuramWorkflowComplete">
    <wsdl:operation name="processCompleted">
      <wsdl:input message="intf:processCompletedRequest"
                  name="processCompletedRequest"/>
    </wsdl:operation>
  </wsdl:portType>
```

Abbildung 3. Porttyp für den Rückruf

Partnerlinktyp

Im Prinzip ist der einzige Schritt, der notwendig ist, damit ein Cúram-Workflowprozess Teil eines orchestrierten BPEL-Prozesses ist, den Prozess als Web-Service verfügbar zu machen. Es ist jedoch möglich, einige Metadaten hinzuzufügen, um dem Entwickler des BPEL-Prozesses seine Arbeit zu erleichtern, indem die im Partnerlink erforderlichen Porttypen definiert und ihre Aufgaben angegeben werden.

Anhand der BPEL-Spezifikation können Partnerlinks in der Web Services Description Language (WSDL) für den Service definiert werden, der im Partnerlink mithilfe des WSDL-Erweiterungsmechanismus aufgerufen werden soll. Die für einen Cúram-Workflow-Web-Service erstellte WSDL definiert den entsprechenden Partnerlinktyp und die erforderlichen Porttypen.

```
<!--Partner link type-->
  <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>
```

Abbildung 4. WSDL-Erweiterungen für BPEL

Kapitel 20. Dateipositionen

20.1 Übersicht

Während es Dienstprogramme wie das Prozessdefinitionstool (PDT) und andere Administrationsbenutzerschnittstellen gibt, müssen die Ausgaben dieser Tools häufig exportiert und einer Versionssteuerung unterzogen werden. Selbstverständlich müssen diese externalisierten Dateien beim Erstellen oder Installieren von Cúram wieder in das Laufzeitsystem zurückgeführt werden. Cúram speichert diese Dateien standardmäßig in einem vordefinierten Quellenordner, von wo aus die Dateien (nach einer möglichen Vorverarbeitung) in die Datenbank geladen werden. In diesem Kapitel wird die Speicherposition der Quellendateien für den Workflow näher erläutert.

20.2 Dateien der Workflowprozessversion

Workflowprozessdefinitionen (sowohl freigegebene als auch nicht freigegebene) können mithilfe des Standardziels **build database** (Datenbank erstellen) in die entsprechende Datenbanktabelle importiert werden.

Diese Workflowprozessdefinitionen müssen in XML-Dateien in einem `Workflow`-Unterverzeichnis im entsprechenden Cúram-Serverkomponentenverzeichnis (z. B. `...\EJBServer\components\core\workflow` für die Komponente `core` oder `...\EJBServer\components\Appeal\workflow` für die Komponente `Appeal` usw.) gespeichert werden.

Jede Komponente in der Cúram-Anwendung kann über ein Workflow-Verzeichnis verfügen, das die für sie relevanten XML-Dateien für die Prozessdefinition enthält. Alle in diesem Workflow-Verzeichnissen gespeicherten Prozessdefinitionsdateien werden automatisch importiert, wenn das Ziel **build database** ausgeführt wird. Wenn die Prozessdefinitionsdateien nicht gültig sind oder der Name und die Version der Definitionen nicht den in den Dateinamen verwendeten Namen und Versionen entsprechen, schlägt der Import fehl.

Die XML-Dateien für die Workflowprozessdefinition im Dateisystem unterliegen einer strikten Namenskonvention. Sie lautet wie folgt: `Prozessname_vProzessversion.xml`. Dabei gilt:

- `Prozessname` ist der Name des Workflowprozesses.
- `Prozessversion` ist die Version des Workflowprozesses.

Die gleiche Version einer Prozessdefinition kann in mehreren Komponenten in der Cúram-Anwendung vorhanden sein. Die importierte Version stammt immer aus der Komponente, die unter den Komponenten an erster Stelle in der Rangfolge steht. Diese Rangfolge wird mithilfe der Umgebungsvariable 'COMPONENT_ORDER_PRECEDENCE' festgelegt.

Jeder Prozessdefinition wird beim Import eine neue Prozessdefinitionskennung zugewiesen, die für die Datenbank eindeutig ist, in die die Definition importiert wird. Unterschiedlichen Versionen derselben Prozessdefinition wird dieselbe eindeutige Kennung zugeordnet und es kann nur eine nicht freigegebene Version einer Prozessdefinition importiert werden. Um ungültige Workflowprozessdefinitionen bearbeiten zu können, die mithilfe des Ziels 'build database' geladen wurden, werden in der Workflow-Engine strenge Validierungen durchgeführt. Diese Validierungen stellen sicher, dass eine Workflowprozessdefinition nur dann in den Prozessdefinitionscache geladen werden kann, wenn sie alle Prozessvalidierungen erfolgreich durchlaufen hat. Eine Beschreibung dieser Validierungen finden Sie in den Kapiteln weiter oben in diesem Dokument.

© Copyright IBM Corp. 2012, 2013

20.2.1 Anpassen von Dateien der Workflowprozessversion

20.2.1.1 Erstellen neuer Dateien der Workflowprozessversion

Alle neuen Dateien der Workflowprozessversion müssen im Workflow-Unterverzeichnis des Verzeichnisses `...\EJBServer\components\custom` erstellt werden. Um eine neue Datei zu erstellen, können mithilfe des PDT die erforderliche Definition erstellt und alle Details eingegeben werden. Die Definition kann dann mit dem Tool in eine Datei exportiert und an der oben angegebenen Position gespeichert werden.

20.2.1.2 Ändern einer vorhandenen Workflowprozessversionsdatei

Zeigen Sie mithilfe des Tools für Workflowprozessdefinitionen (PDT) die aktuelle Version der Prozessdefinition an, die geändert werden soll. Erstellen Sie mit dem Tool eine neue Version dieser Prozessdefinition. Nehmen Sie die gewünschten Änderungen vor, validieren Sie die Version und geben Sie den Workflow frei.

Exportieren Sie die soeben freigegebenen Workflowprozessdefinition mithilfe des PDT und speichern Sie sie im Workflow-Unterverzeichnis des Verzeichnisses `...\EJBServer\components\custom`.

20.3 Ereignisdefinitionsdateien

Ereignisse stellen einen Mechanismus für lose miteinander verbundene Komponenten in der Cúram-Anwendung für die Kommunikation von Informationen zu Statusänderungen im System bereit. Wenn durch ein Modul in der Anwendung ein Ereignis ausgelöst wird, erhalten ein oder mehrere andere Module eine diesbezügliche Benachrichtigung, sofern sie als Listener des betreffenden Ereignisses registriert sind. Um diese Funktion nutzen zu können, müssen Ereignisse definiert werden. Zudem ist Anwendungscode zur Ereignisauslösung erforderlich und es müssen Ereignishandler als Listener für entsprechende Ereignisse definiert und registriert werden.

Ereignisse werden in Cúram in XML-Dateien definiert, die sowohl die Ereignisklassen als auch die Ereignistypen angeben. Diese Dateien werden mit der Erweiterung `.evx` erstellt und im Verzeichnis `events` einer Cúram-Komponente gespeichert (z. B. `...EJBServer\components\core\events`). Von dort werden sie vom Erstellungsscript abgerufen und verarbeitet.

Es gibt zwei Ausgabetypen, die vom Befehl **evgen** erstellt werden: `.java`-Dateien (für Codekonstanten, die Ereignisse verwenden, die weniger anfällig für Fehler sind) und `.dmx`-Dateien (Cúram-Datenbankscripts für das Laden von Ereignisdefinitionen in die Datenbank). Die Java-Artefakte, die aus einer zusammengeführten Ereignisdatei erstellt werden, werden im Verzeichnis `/build/svr/events/gen/[Paket]` gespeichert, wobei `[Paket]` das Paketattribut ist, das in der Ereignisdefinitionsdatei angegeben ist. Die Datenbankscripts, die aus einer zusammengeführten Ereignisdatei erstellt werden, werden im Verzeichnis `/build/svr/events/gen/dmx` abgelegt.

In Kapitel 10 des Handbuchs *Cúram Server Developer's Guide* finden Sie eine umfassende Beschreibung zu Ereignissen und ihrer Anwendung in der Cúram-Anwendung.

Kapitel 21. Konfiguration

21.1 Übersicht

Die meisten Konfigurationsoptionen sind nicht global auf alle Workflowprozessdefinitionen anwendbar. Häufig gelten sie für eine spezielle Definition und sind daher auch Teil der eigentlichen Prozessdefinition. Vor diesem Hintergrund gibt es jedoch einige wenige Anwendungseigenschaften, die sich auf das gesamte Cúram-Workflow-Management-System auswirken. Im vorliegenden Kapitel werden diese Eigenschaften beschrieben.

21.2 Anwendungseigenschaften

Die folgenden Anwendungseigenschaften können in der Datei Application.prx festgelegt werden:

Eigenschaftsname	Beschreibung
curam.custom. workflow.workresolver	*Zweck:* Der vollständig qualifizierte Name der Anwendungsklasse, die die Rückruf-Schnittstelle `WorkResolver` implementiert. Weitere Informationen hierzu erhalten Sie im Abschnitt 9.4, „Zuteilungsstrategie", auf Seite 67. *Typ:* Zeichenfolge *Standardwert:* curam.core.sl.impl.DefaultWorkResolver
curam.workflow. automaticallyaddtasktousertasks	*Zweck:* Wenn nach dem Auflösen der Zuordnungsziele für eine Aufgabe, die nur einem Benutzer zugewiesen ist, der Wert dieser Eigenschaft auf "yes/true" gesetzt ist, fügt das System diese Aufgabe automatisch zu der Liste "Eigene Aufgaben" eines Benutzers in dessen Posteingang hinzu, damit sie von ihm bearbeitet werden kann. *Typ:* Zeichenfolge *Standardwert:* NO
curam.custom.notifications. notificationdelivery	*Zweck:* Der vollständig qualifizierte Name der Anwendungsklasse, die die Rückruf-Schnittstelle `NotificationDelivery` implementiert. Weitere Informationen hierzu erhalten Sie im Abschnitt 14.3, „Zuteilungsstrategie für Benachrichtigungen", auf Seite 105. *Typ:* Zeichenfolge *Standardwert:* curam.core.sl.impl.NotificationDeliveryStrategy
curam.workflow.disable.audit. wdovalueshistory.before.activity	*Zweck:* Die Tabelle 'WDOValuesHistory' für die WDO-Prüfung der Prozessinstanz wird von der Workflow-Engine während der Ausführung der Workflowprozessinstanz an drei bestimmten Punkten mit Daten aufgefüllt (vor dem Ausführen einer Aktivität, nach dem Ausführen einer Aktivität und vor der Bewertung der Übergänge von einer Aktivität). Wenn diese Eigenschaft auf "true" festgelegt ist, wird sichergestellt, dass vor dem Ausführen einer Aktivität keine Daten in die Tabelle geschrieben werden. *Typ:* Boolesch *Standardwert:* FALSE
curam.workflow.disable.audit. wdovalueshistory.after.activity	*Zweck:* Die Tabelle 'WDOValuesHistory' für die WDO-Prüfung der Prozessinstanz wird von der Workflow-Engine während der Ausführung der Workflowprozessinstanz an drei bestimmten Punkten mit Daten aufgefüllt (vor dem Ausführen einer Aktivität, nach dem Ausführen einer Aktivität und vor der Bewertung der Übergänge von einer Aktivität). Wenn diese Eigenschaft auf "true" festgelegt ist, wird sichergestellt, dass nach dem Ausführen einer Aktivität keine Daten in die Tabelle geschrieben werden. *Typ:* Boolesch *Standardwert:* FALSE

© Copyright IBM Corp. 2012, 2013

Eigenschaftsname	Beschreibung
curam.workflow.disable.audit.wdovalueshistory .transition.evaluation	*Zweck:* Die Tabelle 'WDOValuesHistory' für die WDO-Prüfung der Prozessinstanz wird von der Workflow-Engine während der Ausführung der Workflowprozessinstanz an drei bestimmten Punkten mit Daten aufgefüllt (vor dem Ausführen einer Aktivität, nach dem Ausführen einer Aktivität und vor der Bewertung der Übergänge von einer Aktivität). Wenn diese Eigenschaft auf "true" festgelegt ist, wird sichergestellt, dass vor dem Bewerten der Übergänge von einer Aktivität keine Daten in die Tabelle geschrieben werden. *Typ:* Boolesch *Standardwert:* FALSE
curam.custom.workflow.processcachesize	*Zweck:* Die Workflow-Engine stellt freigegebene Versionen von Prozessdefinitionen in den Cache (um beim Abrufen von Metadaten den Systemaufwand zu minimieren). Diese Eigenschaft steuert die maximale Anzahl der im Cache gespeicherten Prozessversionen. Wenn diese Anzahl erreicht wird, beginnt die Engine damit, unter Verwendung der LRU-Ausgaberichtlinie (LRU, Least Recently Used) Prozessversionen aus dem Cache auszugeben. Änderungen, die zur Laufzeit am Wert dieser Eigenschaft vorgenommen wurden, werden beim nächsten Versuch der Workflow-Engine, eine Prozessversion in den Cache einzufügen, wirksam. *Typ:* Ganze Zahl *Standardwert:* 250
curam.batchlauncher.dbtojms. notification.batchlaunchermode	Weitere Informationen finden Sie im Handbuch *Cúram Batch Processing Guide* im Abschnitt 5.3.
curam.batchlauncher.dbtojms. notification.encoding	Weitere Informationen finden Sie im Handbuch *Cúram Batch Processing Guide* im Abschnitt 5.3.
curam.batchlauncher.dbtojms. notification.host	Weitere Informationen finden Sie im Handbuch *Cúram Batch Processing Guide* im Abschnitt 5.3.
curam.batchlauncher.dbtojms. messagespertransaction	Weitere Informationen finden Sie im Handbuch *Cúram Batch Processing Guide* im Abschnitt 5.3.
curam.batchlauncher.dbtojms. notification.port	Weitere Informationen finden Sie im Handbuch *Cúram Batch Processing Guide* im Abschnitt 5.3.

Kapitel 22. JMSLite

22.1 Einführung

JMSLite ist ein von Cúram entwickelter schlanker JMS-Server, der im Rahmen der RMI-basierten Testumgebung läuft. Daher kann er in unterstützten integrierten Entwicklungsumgebungen (IDEs) ausgeführt werden.

Dadurch können Prozessdefinitionen in einer Entwicklungsumgebung getestet werden, ohne dass die Anwendung auf einem EJB-Server bereitgestellt werden muss. Wird JMSLite in Verbindung mit dem Prozessdefinitionstool verwendet, können Entwickler in ihrer integrierten Entwicklungsumgebung Workflows definieren, bereitstellen und umsetzen.

22.2 Funktionsweise von JMSLite

JMSLite ist ein JMS-Server, der nur die Abschnitte der JMS-Spezifikation implementiert, die für die Unterstützung von IDE-basierten Tests von Cúram-Workflows erforderlich sind (nämlich das transaktionsorientierte Punkt-zu-Punkt-Messaging). Dies bedeutet, dass JMSLite ACID-Transaktionen unter Einbeziehung der Anwendungsdatenbank *und* der auf Grundlage der Infrastruktur definierten Workflowwarteschlangenziele unterstützt. Der Server unterstützt keine benutzerdefinierten (anwendungsdefinierten) Warteschlangen oder die Publish/Subscribe-Domäne (d. h. Themen).

Daher ermöglicht JMSLite dem Workflowumsetzungsservice und der Workflow-Engine, JMS-Nachrichten asynchron zu senden. Dies bedeutet, dass Anwendungsaufrufe an die Infrastruktur-APIs, die sich auf den Workflow beziehen (wie den Umsetzungsservice und den Ereignisservice), nicht geblockt werden. Die APIs leiten Nachrichten an die Workflow-Engine weiter, die Prozessinstanzen asynchron weiterverarbeitet (z. B. automatische Aktivitäten ausführt, Aufgaben erstellt und zuteilt usw.).

22.3 Warum JMSLite?

JMSLite zielt darauf ab, dass sich die Workflow-Engine in einer integrierten Entwicklungsumgebung möglichst so verhält wie bei einer Bereitstellung auf einem Anwendungsserver. Dadurch können etwaige Probleme (beim Testen in der integrierten Entwicklungsumgebung) so früh wie möglich erkannt werden, wodurch mögliche Risiken und Kosten gesenkt werden.

Es wird beispielsweise die folgende Situation angenommen: Das (in einer integrierten Entwicklungsumgebung ausgeführte) WMS soll Workflows *synchron* umsetzen.

Zur Erinnerung: In der Produktion werden Workflows *asynchron* umgesetzt, da vorausgesetzt wird, dass sie in Bezug auf normale Benutzeroperationen (die Sekunden oder Millisekunden dauern) eine lange Lebensdauer (Stunden, Tage oder Wochen) haben.
Angenommen, ein Entwickler soll zudem eine Methode schreiben, die einen automatisierten Workflow zur Fallgenehmigung umsetzt, und dann (sofort nach dem Aufruf an den Umsetzungsservice) versuchen, etwas mit dem Ergebnis zu machen (z. B. prüfen, ob der Fall automatisch genehmigt wurde). Da die Testumgebung sich anders (synchron) als die Produktionsumgebung verhält, würde der Code zwar beim Test funktionieren, jedoch in der Produktion fehlschlagen (dies ist ein Beispiel für einen 'zeitlichen Kopplungsfehler').

Da jedoch JMSLite die Ausführung asynchron vollzieht, würde sich dieses Problem in der integrierten Entwicklungsumgebung in gleicher Weise wie auf einem Anwendungsserver zeigen und kann somit vom Entwickler frühzeitig erkannt werden.

© Copyright IBM Corp. 2012, 2013

22.4 Verwendung von JMSLite

Der JMSLite-Server fragt Gruppenpostfächer ab und entpackt alle darin enthaltenen Nachrichten. Diese Nachrichten führen zu Aufrufen vom JMSLite-Server an den RMI-Server, der für das IDE-basierte Testen von Cúram-Methoden (allgemein als `StartServer` bezeichnet) erforderlich ist. Der JMSLite-Server wird beim Aufrufen des (`StartServer`) Prozesses als Thread gestartet. Da der JMSLite-Server Nachrichten an die Workflow-Engine sendet, die auf dem RMI-Server ausgeführt wird, ist es erforderlich, beim Debuggen von Workflowmethoden 'StartServer' im Debugmodus zu starten.

22.5 Debugging von Workflows

Für gewöhnlich werden Cúram-Infrastrukturmethoden von der Anwendung aufgerufen. In Cúram Workflow wird dieser Aufruf häufig genau anders herum durchgeführt, das heißt, die Workflow-Engine (Infrastruktur) ruft eine Anwendungsmethode (z. B. eine Methode zur Arbeitszuteilung) auf. In einem solchen Fall ist es für einen Anwendungsentwickler nicht möglich, vom Aufruf der Methode `curam.util.workflow.impl.EnactmentService.startProcess()` in ihre Anwendungsmethode (zur Arbeitszuteilung) zu wechseln. Der Entwickler muss daher innerhalb der Methode, für die das Debugging durchgeführt werden soll, einen Breakpoint einrichten und dann die Methode für die Umsetzung des Workflows ausführen. Die Workflow-Engine ruft dann (asynchron) die Anwendungsmethode auf, wodurch der Breakpoint erreicht wird. Der Debugger setzt daraufhin die Ausführung am angegebenen Breakpoint aus, sodass ein normales Debugging durchgeführt werden kann.

Folgende Anwendungsmethoden fallen in die oben genannte Kategorie:

- Methoden für automatische Aktivität
- Funktionen für Arbeitszuteilung
- Die Anwendungsmethode der Benachrichtigungsbereitstellung
- Die geschäftliche Resolver-Anwendungsmethode

Kapitel 23. Posteingang und Aufgabenverwaltung

23.1 Übersicht

Aufgaben werden zum Zuweisen und Überwachen der Arbeit von Systembenutzern verwendet und werden erstellt, wenn manuelle Aktivitäten (siehe Kapitel 9, „Manuell", auf Seite 61), Entscheidungsaktivitäten (siehe Kapitel 10, „Entscheidung", auf Seite 77) oder parallele Aktivitäten (siehe Kapitel 13, „Parallel", auf Seite 95) von der Workflow-Engine ausgeführt werden. Der Posteingang und seine zugehörigen Funktionen zur Aufgabenverwaltung werden von den Benutzern der Cúram-Anwendung zum Verwalten dieser Aufgaben verwendet. In den folgenden Abschnitten werden die Konfigurations- und Anpassungsoptionen beschrieben, die für die Bereiche Posteingang und Aufgabenverwaltung des Cúram-WMS verfügbar sind.

23.2 Konfiguration des Posteingangs

23.2.1 Konfigurationseinstellungen für die Größe der Posteingangslisten

Im Posteingang sind verschiedene Listenansichten für Aufgaben verfügbar. Dazu zählen:

- *Meine offenen Aufgaben*: Eine Liste von Aufgaben, die der Benutzer aktuell bearbeitet.
- *Meine zurückgestellten Aufgaben*: Eine Liste von Aufgaben, die der Benutzer zwar bearbeitet, die jedoch auf ein späteres Datum zurückgestellt wurden.
- *Verfügbare Aufgaben*: Eine Liste von Aufgaben, die dem Benutzer zur Bearbeitung zur Verfügung stehen.
- *Aufgabenabfrage - Suchergebnisse*: Eine Liste von Aufgaben, die das Ergebnis der Ausführung einer Aufgabenabfrage sind.
- *Aufgaben in Gruppenpostfach*: Eine Liste von Aufgaben, die einem Gruppenpostfach zugeordnet sind.

Es gibt zudem eine Liste im Posteingang, die die Benachrichtigungen an einen Benutzer enthält.

- *Meine Benachrichtigungen*: Eine Liste mit Benachrichtigungen, die an den Benutzer gesendet wurden.

Die Listenansichten des Posteingangs können so konfiguriert werden, dass dem Benutzer nur eine bestimmte Anzahl von Datensätzen zurückgegeben wird. Dazu können die folgenden Anwendungseigenschaften in der Datei Application.prx festgelegt werden.

© Copyright IBM Corp. 2012, 2013

Tabelle 5. Konfigurationseinstellungen für die Größe der Posteingangslisten

Eigenschaftsname	Beschreibung
curam.inbox.max.task.list.size	*Zweck:* Der Wert der Eigenschaft steuert die Anzahl der Aufgaben, die in den verschiedenen Listenansichten für die Aufgaben im Posteingang angezeigt werden. Zu den Seiten der Aufgabenlisten, auf die sich der Wert dieser Eigenschaft auswirkt, zählen: 'Meine offenen Aufgaben', 'Meine zurückgestellten Aufgaben', 'Verfügbare Aufgaben', 'Aufgabenabfrage - Suchergebnisse', 'Aufgaben in Gruppenpostfach'. Wenn die Anzahl der anzuzeigenden Aufgaben den angegebenen Wert übersteigt, wird eine Nachricht angezeigt, die den Benutzer darüber informiert, dass nicht alle Datensätze, die den Suchkriterien der Seite entsprechen, angezeigt werden. Diese Nachricht enthält sowohl die Anzahl der angezeigten Aufgaben als auch die Gesamtanzahl der Aufgaben, die den Suchkriterien entsprechen. *Typ:* Ganze Zahl *Standardwert:* 100
curam.notification.max.list.size	*Zweck:* Der Wert der Eigenschaft steuert die Anzahl der Benachrichtigungen, die in der Listenansicht 'Meine Benachrichtigungen' des Posteingangs angezeigt werden. Wenn die Anzahl der anzuzeigenden Benachrichtigungen den angegebenen Wert übersteigt, wird eine Nachricht angezeigt, die den Benutzer darüber informiert, dass nicht alle Datensätze, die den Suchkriterien der Seite entsprechen, angezeigt werden. Diese Nachricht enthält sowohl die Anzahl der angezeigten Benachrichtigungen als auch die Gesamtanzahl der Benachrichtigungen, die den Suchkriterien entsprechen. *Typ:* Ganze Zahl *Standardwert:* 100

23.2.2 Konfigurationseinstellungen für Funktionen des Typs 'Nächste Aufgabe abrufen'

Im Posteingang sind einige Verknüpfungsfunktionen verfügbar, um die nächste Aufgabe zur Bearbeitung aufzurufen. Zu diesen Funktionen zählen:

- Nächste Aufgabe abrufen: Ruft die nächste Aufgabe aus den Aufgaben ab, die für den Benutzer verfügbar sind.
- Nächste Aufgabe aus bevorzugter Organisationseinheit abrufen: Ruft die nächste Aufgabe ab, die der bevorzugten Organisationseinheit des Benutzers zugeordnet ist.
- Nächste Aufgabe aus bevorzugtem Postfach holen: Ruft die nächste Aufgabe ab, die dem bevorzugten Gruppenpostfach des Benutzers zugeordnet ist.
- Nächste Aufgabe aus Postfach holen: Ruft die nächste Aufgabe ab, die einem vom Benutzer ausgewählten Gruppenpostfach zugeordnet ist.

Die von diesen Verknüpfungsfunktionen verwendeten Algorithmen zum Abrufen der nächsten Aufgabe können durch Verwendung der folgenden Anwendungseigenschaften in der Datei Application.prx konfiguriert werden:

Tabelle 6. Konfigurationseinstellungen für Funktionen des Typs 'Nächste Aufgabe abrufen'

Eigenschaftsname	Beschreibung
curam.workflow. reservenexttaskwithpriorityfilter	*Zweck:* Der Wert der Eigenschaft steuert, ob der Algorithmus für das Abrufen der nächsten Aufgabe die Priorität einer Aufgabe zum Ermitteln der nächsten abzurufenden Aufgabe verwendet. Ist der Wert auf den Standardwert *YES* gesetzt, wird die Priorität der Aufgabe für diesen Zweck verwendet (die Prioritäten, wie sie in der Eigenschaft 'curam.workflow.taskpriorityorder' festgelegt sind). Andernfalls basiert die abzurufende Aufgabe auf den Aufgaben, die dem Benutzer am längsten zugeordnet sind. *Typ:* Zeichenfolge *Standardwert:* Yes
curam.workflow.taskpriorityorder	*Zweck:* Es sind drei Aufgabenprioritäten im Workflow Management System angegeben: 'Hoch', 'Mittel' und 'Niedrig' (die den Codes TP1, TP2 und TP3 in der Codetabelle `TaskPriority` entsprechen). In einigen Fällen kann es für Kunden erforderlich sein, eine neue Aufgabenpriorität hinzuzufügen (z. B. 'Kritisch' mit dem Codetabellenwert TP4). Das Abrufen von Aufgaben anhand der Aufgabenpriorität mit diesem Wert würde daher sicherstellen, dass wichtige Aufgaben nach den Aufgaben mit einer niedrigen Priorität angezeigt werden würden (wenn die Intention wäre, dass Aufgaben mit dieser Priorität als erstes abgerufen werden sollten). Mit dieser Eigenschaft können die Aufgabenprioritäten in einer beliebigen Reihenfolge angegeben werden, die den Anforderungen eines Kunden entspricht. *Typ:* Zeichenfolge *Standardwert:* TP1,TP2,TP3

23.2.3 Einstellungen für Aufgabenumleitung und Zuteilungssperre

Mithilfe der Aufgabenumleitung kann ein Benutzer Aufgaben für eine bestimmte Zeit an einen anderen Benutzer, ein anderes Organisationsobjekt (Organisationseinheit, Position oder Job) oder ein anderes Gruppenpostfach umleiten. Mit einer Sperre der Aufgabenzuteilung kann ein Benutzer sicherstellen, dass ihm für einen bestimmten Zeitraum keine Aufgaben zugewiesen werden. Diese Funktion steht dem Benutzer im Bereich "Vorgaben für Aufgabe" im Posteingang zur Verfügung. Jedoch benötigen nicht alle Systembenutzer unbedingt Zugriff auf die Einstellungen für Aufgabenumleitung und Zuteilungssperre. Um dieser Tatsache gerecht zu werden, können diese Funktionsbereiche im Posteingang für bestimmte Benutzer mithilfe von Sicherheitskennungen inaktiviert werden. In der folgenden Tabelle sind die Sicherheitskennungen aufgeführt, über die ein Benutzer zur Nutzung dieser Funktionen verfügen muss.

Tabelle 7. Sicherheitskennungen und zugehörige Aktionen

Name der Sicherheitskennung	Zulässige Aktion
UserTaskRedirection. listTaskRedirectionHistoryForUser	Ermöglicht einem Benutzer, alle für ihn angegeben Zeiträume für die Aufgabenumleitung anzuzeigen.
UserTaskRedirection. redirectTasksForUser	Ermöglicht einem Benutzer, für sich selbst einen Aufgabenumleitungszeitraum zu erstellen.
UserTaskRedirection. clearTaskRedirectionForUser	Ermöglicht einem Benutzer, einen seiner Aufgabenumleitungszeiträume zu löschen.
UserTaskAllocationBlocking.list. TaskAllocationBlockingHistoryForUser	Ermöglicht einem Benutzer, alle für ihn angegebenen Zeiträume für die Sperre der Aufgabenzuteilung anzuzeigen.
UserTaskAllocationBlocking. blockTaskAllocationForUser	Ermöglicht einem Benutzer, für sich selbst einen Zeitraum für die Sperre der Aufgabenzuteilung zu erstellen.
UserTaskAllocationBlocking. clearTaskAllocationBlockForUser	Ermöglicht einem Benutzer, einen seiner Zeiträume für die Sperre der Aufgabenzuteilung zu löschen.

23.3 Anpassen des Posteingangs

Das Standardverhalten der Funktionen 'Aktionen im Posteingang', 'Aufgabenaktionen' und 'Aufgabensuche' kann durch Verwendung von Guice zum Aufrufen von angepasstem Code geändert werden, mit dem das Standardverhalten überschrieben werden kann.

Anmerkung: Bei Guice handelt es sich um ein von *Google* entwickeltes Framework, das nicht im Rahmen des vorliegenden Dokuments beschrieben wird. Nähere Informationen zu Guice finden Sie im Benutzerhandbuch zu Guice.

Das Cúram-Workflow-Management-System enthält die folgenden Anpassungen und die entsprechenden Standardimplementierungen:

Tabelle 8. Anpassungen

Anpassung	Schnittstellenklasse	Standardimplementierungsklasse
Aktionen im Posteingang	`curam.core.hook.task.impl.InboxActions`	`curam.core.hook.task.impl.InboxActionsImpl`
Aufgabenaktionen	`curam.core.hook.task.impl.TaskActions`	`curam.core.hook.task.impl.TaskActionsImpl`
Aufgabensuche und Suche nach verfügbaren Aufgaben	`curam.core.hook.task.impl.SearchTask`	`curam.core.hook.task.impl.SearchTaskImpl`
Aufgabenabfrage	`curam.core.hook.task.impl.TaskQuery`	`curam.core.hook.task.impl.TaskQueryImpl`
SQL-Generation für Aufgabensuche	`curam.core.hook.task.impl.SearchTaskSQL`	`curam.core.hook.task.impl.SearchTaskSQLImpl`

Die folgenden *Aktionen im Posteingang* können angepasst werden:
- Nächste Aufgabe abrufen
- Nächste Aufgabe aus bevorzugter Organisationseinheit abrufen
- Nächste Aufgabe aus bevorzugtem Postfach holen
- Nächste Aufgabe aus Gruppenpostfach abrufen
- Gruppenpostfach für Benutzer abonnieren
- Benutzerabonnement des Gruppenpostfachs kündigen

Die folgenden *Aufgabenaktionen* können angepasst werden:
- Kommentar hinzufügen
- Schließen
- Erstellen
- Zurückstellen
- Erneut starten
- Weiterleiten
- Arbeitszeit ändern
- Priorität ändern
- Frist ändern
- Neu zuteilen
- Zu meinen Aufgaben hinzufügen

Die folgenden Methoden für *Aufgabensuche* und *Suche nach verfügbaren Aufgaben* können angepasst werden:
- `countAvailableTasks`
- `countTasks`
- `searchAvailableTasks`
- `searchTask`
- `validateSearchTask`

Die folgenden *Aufgabenabfragen* können angepasst werden:

- createTaskQuery
- modifyTaskQuery
- runTaskQuery
- validateTaskQuery

Die folgenden Methoden der SQL-Generierung für die Aufgabensuche können angepasst werden. Mit diesen Methoden wird das SQL für alle der oben aufgeführten Suchfunktionen generiert.

- getBusinessObjectTypeSQL
- getCategorySQL
- getCountSQLStatement
- getCreationDateSQL
- getDeadlineSQL
- getFromClause
- getOrderBySQL
- getOrgObjectSQL
- getPrioritySQL
- getReservedBySQL
- getRestartDateSQL
- getSelectClause
- getSQLStatement
- getStatusSQL
- getTaskIDSQL
- getWhereClause

23.3.1 Anpassen des Posteingangs

Der folgende Abschnitt enthält eine Beschreibung zur Anpassung der Posteingangsaktion curam.core-.hook.task.impl.InboxActionsImpl.getNextTask. Der hier aufgeführte Prozess kann auch für alle anderen Anpassungen angewendet werden.

Es muss eine benutzerdefinierte Ankerpunktklasse erstellt werden. Diese Klasse *muss* die Standardimplementierungsklasse erweitern. Das unten dargestellte Diagramm veranschaulicht die Beziehungen zwischen den Klassen:

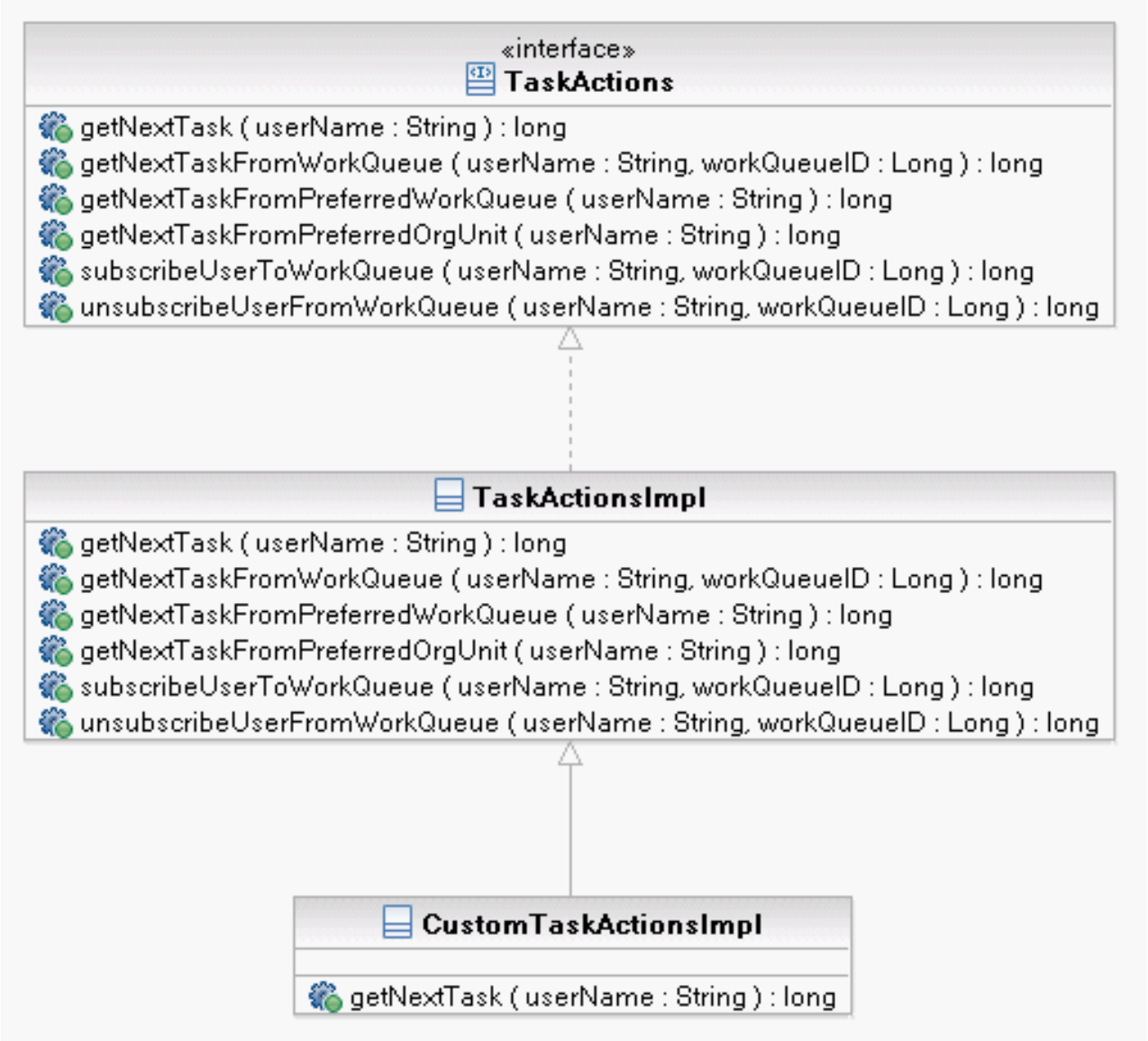

Abbildung 5. Anpassungsklassendiagramm

Anmerkung: Die benutzerdefinierte Klasse darf *niemals* direkt die Schnittstellenklasse implementieren, da dies während einer Aktualisierung zu Ausnahmen bei der Kompilierzeit führen kann, wenn der Schnittstelle neue Methoden hinzugefügt wurden. In diesem Fall würde die benutzerdefinierte Klasse die neuen Methoden nicht implementieren, was zu einem Bruch zwischen der Schnittstellenklasse und der Implementierungsklasse und folglich zu Ausnahmen bei der Kompilierzeit führen würde.

23.3.1.1 Anpassen der Standardimplementierung

Die Signatur der Funktion `getNextTask` in der Schnittstelle `curam.core.hook.task.impl.InboxActions` lautet folgendermaßen:

```
package curam.core.hook.task.impl;

@ImplementedBy(InboxActionsImpl.class)
public interface InboxActions {

  public long getNextTask(String userName);

   .
```

```
   .
   .
   .
}
```

Die Standardimplementierung für die Funktion ist in der Klasse `curam.core.hook.task.impl.InboxActionsImpl` angegeben.

```
package curam.core.hook.task.impl;

public class InboxActionsImpl implements InboxActions {

  public long getNextTask(String userName) {
    // Default implementation code is here....
  }

  .
  .
  .
  .
}
```

Um die Funktion `getNextTask` anzupassen, muss die Methode in der neuen, zuvor erstellten benutzerdefinierten Klasse implementiert werden, die die Standardimplementierungsklasse `curam.core.hook.task.impl.InboxActionsImpl` erweitert.

```
package custom.hook.task.impl;

public class CustomInboxActionsImpl extends InboxActionsImpl {

  public long getNextTask(final String userName) {
    // Custom implementation code goes here
  }

}
```

Um sicherzustellen, dass die Anwendung die neue benutzerdefinierte Klasse (und nicht die Standardimplementierung) ausführt, muss eine neue Klasse `custom.hook.task.impl.Module.java` als Erweiterung von `com.google.inject.AbstractModule` mit der implementierten Methode `configure` geschrieben werden, wie das folgende Beispiel zeigt:

```
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);
    }
}
```

Abschließend muss der Klassenname `custom.hook.task.impl.Module` in die Spalte *ModuleClassName* der Datenbanktabelle *ModuleClassName* eingefügt werden. Dies kann durch das Hinzufügen einer zusätzlichen Zeile in der Datei `ModuleClassName.DMX` oder bei Bedarf auch direkt in die Datenbanktabelle erfolgen.

Bei Verwendung dieses Ansatzes ruft das System bei der erneuten Bereitstellung der Anwendung die angepasste Version der Funktion `getNextTask` (und nicht die Standardimplementierung) auf.

Bemerkungen

Die vorliegenden Informationen wurden für Produkte und Services entwickelt, die auf dem deutschen Markt angeboten werden. Möglicherweise bietet IBM die in dieser Dokumentation beschriebenen Produkte, Services oder Funktionen in anderen Ländern nicht an. Informationen über die gegenwärtig im jeweiligen Land verfügbaren Produkte und Services sind beim zuständigen IBM Ansprechpartner erhältlich. Hinweise auf IBM-Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Services von IBM verwendet werden können. Anstelle der IBM Produkte, Programme oder Services können auch andere, ihnen äquivalente Produkte, Programme oder Services verwendet werden, solange diese keine gewerblichen oder anderen Schutzrechte von IBM verletzen. Die Verantwortung für den Betrieb von Produkten, Programmen und Services anderer Anbieter liegt beim Kunden. Für die in diesem Handbuch beschriebenen Erzeugnisse und Verfahren kann es IBM Patente oder Patentanmeldungen geben. Mit der Auslieferung dieses Handbuchs ist keine Lizenzierung dieser Patente verbunden. Lizenzanforderungen sind schriftlich an folgende Adresse zu richten (Anfragen an diese Adresse müssen auf Englisch formuliert werden):

IBM Director of Licensing

IBM Europe, Middle East & Africa

Tour Descartes

2, avenue Gambetta

92066 Paris La Defense

France

Trotz sorgfältiger Bearbeitung können technische Ungenauigkeiten oder Druckfehler in dieser Veröffentlichung nicht ausgeschlossen werden.

Die hier enthaltenen Informationen werden in regelmäßigen Zeitabständen aktualisiert und als Neuausgabe veröffentlicht. IBM kann ohne weitere Mitteilung jederzeit Verbesserungen und/oder Änderungen an den in dieser Veröffentlichung beschriebenen Produkten und/oder Programmen vornehmen. Verweise in diesen Informationen auf Websites anderer Anbieter werden lediglich als Service für den Kunden bereitgestellt und stellen keinerlei Billigung des Inhalts dieser Websites dar.

Das über diese Websites verfügbare Material ist nicht Bestandteil des Materials für dieses IBM Produkt. Die Verwendung dieser Websites geschieht auf eigene Verantwortung.

Werden an IBM Informationen eingesandt, können diese beliebig verwendet werden, ohne dass eine Verpflichtung gegenüber dem Einsender entsteht. Lizenznehmer des Programms, die Informationen zu diesem Produkt wünschen mit der Zielsetzung: (i) den Austausch von Informationen zwischen unabhängig voneinander erstellten Programmen und anderen Programmen (einschließlich des vorliegenden Programms) sowie (ii) die gemeinsame Nutzung der ausgetauschten Informationen zu ermöglichen, wenden sich an folgende Adresse:

IBM Corporation

Dept F6, Bldg 1

294 Route 100

© Copyright IBM Corp. 2012, 2013

Somers NY 10589-3216

U.S.A.

Die Bereitstellung dieser Informationen kann unter Umständen von bestimmten Bedingungen - in einigen Fällen auch von der Zahlung einer Gebühr - abhängig sein.

Die Bereitstellung des in diesem Dokument beschriebenen Lizenzprogramms sowie des zugehörigen Lizenzmaterials erfolgt auf der Basis der IBM Rahmenvereinbarung bzw. der Allgemeinen Geschäftsbedingungen von IBM, der IBM Internationalen Nutzungsbedingungen für Programmpakete oder einer äquivalenten Vereinbarung.

Alle in diesem Dokument enthaltenen Leistungsdaten stammen aus einer kontrollierten Umgebung. Die Ergebnisse, die in anderen Betriebsumgebungen erzielt werden, können daher erheblich von den hier erzielten Ergebnissen abweichen. Einige Daten stammen möglicherweise von Systemen, deren Entwicklung noch nicht abgeschlossen ist. Eine Gewährleistung, dass diese Daten auch in allgemein verfügbaren Systemen erzielt werden, kann nicht gegeben werden. Darüber hinaus wurden einige Daten unter Umständen durch Extrapolation berechnet. Die tatsächlichen Ergebnisse können davon abweichen. Benutzer dieses Dokuments sollten die entsprechenden Daten in ihrer spezifischen Umgebung prüfen.

Alle Informationen zu Produkten anderer Anbieter stammen von den Anbietern der aufgeführten Produkte, deren veröffentlichten Ankündigungen oder anderen allgemein verfügbaren Quellen.

IBM hat diese Produkte nicht getestet und kann daher keine Aussagen zu Leistung, Kompatibilität oder anderen Merkmalen machen. Fragen zu den Leistungsmerkmalen von Produkten anderer Anbieter sind an den jeweiligen Anbieter zu richten.

Aussagen über Pläne und Absichten von IBM unterliegen Änderungen oder können zurückgenommen werden und repräsentieren nur die Ziele von IBM.

Alle von IBM angegebenen Preise sind empfohlene Richtpreise und können jederzeit ohne weitere Mitteilung geändert werden. Händlerpreise können u. U. von den hier genannten Preisen abweichen.

Diese Veröffentlichung dient nur zu Planungszwecken. Die in dieser Veröffentlichung enthaltenen Informationen können geändert werden, bevor die beschriebenen Produkte verfügbar sind.

Diese Veröffentlichung enthält Beispiele für Daten und Berichte des alltäglichen Geschäftsablaufs. Sie sollen nur die Funktionen des Lizenzprogramms illustrieren und können Namen von Personen, Firmen, Marken oder Produkten enthalten. Alle diese Namen sind frei erfunden; Ähnlichkeiten mit tatsächlichen Namen und Adressen sind rein zufällig.

COPYRIGHTLIZENZ:

Diese Veröffentlichung enthält Musteranwendungsprogramme, die in Quellensprache geschrieben sind und Programmiertechniken in verschiedenen Betriebsumgebungen veranschaulichen. Sie dürfen diese Musterprogramme kostenlos kopieren, ändern und verteilen, wenn dies zu dem Zweck geschieht, Anwendungsprogramme zu entwickeln, zu verwenden, zu vermarkten oder zu verteilen, die mit der Anwendungsprogrammierschnittstelle für die Betriebsumgebung konform sind, für die diese Musterprogramme geschrieben werden. Diese Beispiele wurden nicht unter allen denkbaren Bedingungen getestet. IBM kann daher die Zuverlässigkeit, Wartungsfreundlichkeit oder Funktion dieser Programm nicht garantieren oder implizieren. Die Musterprogramme werden "WIE BESEHEN", ohne Gewährleistung jeglicher Art bereitgestellt. IBM übernimmt keine Haftung für Schäden, die durch Ihre Verwendung der Musterprogramme entstehen.

Kopien oder Teile der Musterprogramme bzw. daraus abgeleiteter Code müssen folgenden Copyrightvermerk beinhalten:

© (Name Ihres Unternehmens) (Jahr). Teile des vorliegenden Codes wurden aus Musterprogrammen der IBM Corp. abgeleitet.

© Copyright IBM Corp. _Jahreszahl oder Jahreszahlen eingeben_. Alle Rechte vorbehalten.

Wird dieses Buch als Softcopy (Book) angezeigt, erscheinen keine Fotografien oder Farbabbildungen.

Marken

IBM, das IBM Logo und ibm.com sind weltweit Marken oder eingetragene Marken der International Business Machines Corporation. Weitere Produkt- und Servicenamen können Marken von IBM oder anderen Herstellern sein. Eine aktuelle Liste der IBM Marken finden Sie auf der Website "Copyright and trademark information" unter http://www.ibm.com/legal/us/en/copytrade.shtml.

Apache ist eine Marke der Apache Software Foundation.

Java und alle auf Java basierenden Marken und Logos sind eingetragene Marken der Oracle Corporation und/oder ihrer verbundenen Unternehmen.

Andere Namen sind möglicherweise Marken der jeweiligen Rechtsinhaber. Namen anderer Unternehmen, Produkte und Dienstleistungen können Marken oder Dienstleistungsmarken anderer Unternehmen sein.

Gedruckt in Deutschland