IBM Cúram Social Program Management

Guía de referencia de Cúram Workflow

Versión 6.0.5

IBM Cúram Social Program Management

Guía de referencia de Cúram Workflow

Versión 6.0.5

Nota

Antes de utilizar esta información y el producto al que hace referencia, lea la información que figura en el apartado "Avisos" en la página 145

Revisado: mayo de 2013

Esta edición se aplica a IBM Cúram Social Program Management, versión 6.0 5 y a todos los releases posteriores a menos que se indique lo contrario en ediciones nuevas.

Contenido

Figuras

Tablas

Capítulo 1. Introducción

1.1 Descripción general

Esta es la Guía de referencia del flujo de trabajo y está destinada a proporcionar explicaciones detalladas de los conceptos de Workflow Management System (WMS) de Cúram. Su objetivo es describir cómo definir un proceso para lograr determinados objetivos proporcionando descripciones detalladas de los metadatos del flujo de trabajo, así como de los efectos que dichos metadatos tienen en tiempo de ejecución. Este documento no está concebido como un tutorial, sino como una descripción concisa de todas las características disponibles en el flujo de trabajo de Cúram.

1.2 Requisitos previos

Este documento presupone cierta familiaridad con los conceptos de flujo de trabajo y con la forma en que se llevan a cabo en el WMS de Cúram. En concreto, se presupone que al menos ha leído *Guías de analistas empresariales: Guía general de los flujos de trabajo de Cúram*.

1.3 Cómo utilizar este documento

Al ser este documento un manual de consulta, se intenta que los capítulos sean entre sí lo más independientes posible. El objetivo es que un lector sea consciente del concepto del que desea obtener más detalles, encuentre el capítulo relevante en este documento y sólo necesite leer dicho capítulo. Aunque no se espera que este documento se lea de principio a fin, se ha estructurado de forma que tal lectura sea posible y productiva.

Algunas partes del propio WMS de Cúram están fuertemente relacionadas entre sí, y esto se refleja en la documentación. Dichas referencias externas son de dos tipos: requisitos previos que son referencias a información imprescindible para comprender una determinada sección, y enlaces generales que destacan información relacionada pero no imprescindible.

1.4 Estructura de este documento

Este documento también se puede ver en una serie de secciones diferentes, cada una de las cuales refleja un área del WMS de Cúram, y cómo interaccionan entre sí. Las secciones siguientes incluyen un resumen de cuáles son estas secciones lógicas, qué capítulos se incluyen en ellas y qué áreas del WMS de Cúram se contemplan en esos capítulos relacionados.

1.4.1 Procesos de flujo de trabajo

La sección *Procesos de flujo de trabajo* del documento describe los metadatos asociados a una definición de proceso de flujo de trabajo. También se describe el ciclo de vida de una definición de proceso.

Capítulo 2, “Creación de un proceso de flujo de trabajo”, en la página 5 describe cómo crear y visualizar un proceso de flujo de trabajo utilizando el sistema de flujo de trabajo de Cúram. También se describe la publicación de un proceso, así como el efecto que esto tiene en el versionado asociado a las definiciones de proceso. Se analizan la exportación e importación de definiciones de proceso, mientras que la traducción del texto contenido en un proceso se trata de forma esquemática. La ejecución de un proceso de flujo de trabajo utilizando el motor de flujo de trabajo de Cúram se describe con detalle. También se proporciona una descripción de la biblioteca de métodos y del objeto de datos de flujo trabajo (consulte Capítulo 4, “Objetos de datos de flujo de trabajo”, en la página 17).

Capítulo 3, “Metadatos de una definición de proceso”, en la página 13 describe los metadatos asociados a una definición de proceso de flujo de trabajo. Se esquematiza cada campo de metadatos, mientras que se detallan las validaciones y los objetos de datos de flujo de trabajo asociados al proceso de flujo de trabajo en conjunto.

1.4.2 Flujo de datos

La sección *Flujo de datos* del documento describe cómo se almacenan y manipulan los datos en una instancia de proceso. En particular, se describen las cuestiones de cómo transportar los datos desde el mundo exterior (en la promulgación de un proceso) y entre actividades y transiciones dentro del proceso.

Capítulo 4, “Objetos de datos de flujo de trabajo”, en la página 17 describe los objetos utilizados para mantener y pasar datos en el motor de flujo de trabajo. Se describen en detalle los metadatos de que constan los objetos de datos de flujo de trabajo y sus atributos. Se tratan las validaciones que corresponden a la creación y modificación de un objeto de datos de flujo de trabajo. Finalmente, también se describen los objetos de datos de flujo de trabajo de contexto disponibles gracias a la herramienta de definición de procesos y al motor de flujo de trabajo.

Capítulo 5, “Promulgación de procesos”, en la página 25 describe el inicio de una instancia de proceso (es decir, la realización del trabajo especificado en la definición de proceso). Se tratan el API de servicios de promulgación y los metadatos de las correlaciones de promulgación asociados a la promulgación de un proceso. También se proporcionan validaciones y ejemplos de código. También es posible iniciar un proceso en respuesta a un suceso que se está activando, y esto también se describe en este capítulo. Los datos de configuración para realizar esta acción se describen en detalle. Se describen las validaciones que se ejecutan cuando se crean las correlaciones entre sucesos y procesos de flujo de trabajo.

1.4.3 Actividades

Las actividades son fundamentales en un proceso de flujo de trabajo, porque son los pasos en que tiene lugar el proceso de negocio del flujo de trabajo. Cúram WMS soporta varios tipos de actividad, y se describen todos en la sección *Actividades* del documento. Como las notificaciones también son pertinentes para cada tipo de actividad, también se describen en esta sección del documento.

Capítulo 6, “Actividad base”, en la página 31 describe los detalles de los metadatos comunes a todos los tipos de actividad soportados en el sistema de flujo de trabajo de Cúram. También se describen las validaciones que se ejecutan al crear o modificar una actividad. Por último, se describen algunos de los tipos de actividad más sencillos, incluida la actividad de ruta y las actividades de inicio y fin de proceso.

Capítulo 7, “Automática”, en la página 35 describe los detalles de los metadatos asociados a una actividad automática. Se tratan en detalle las correlaciones de entrada y salida especificadas para el método asociado a la actividad automática. Se describen las validaciones ejecutadas cuando se crean o modifican los metadatos de una actividad automática. Por último, también se describen los objetos de datos de flujo de trabajo `Context_Result` y `Context_Error` disponibles para su uso en transiciones desde actividades automáticas.

Capítulo 8, “Espera de suceso”, en la página 49 describe los detalles de los metadatos asociados a una actividad de espera de suceso. Esto incluye la lista de sucesos, los detalles de la fecha límite (incluidos los recordatorios de fecha límite) asociados a una espera de suceso así como cualquier correlación de salida que pueda especificarse. También se describen las validaciones ejecutadas cuando se crean o modifican metadatos de una espera de suceso. También se describe en detalle la información de tiempo de ejecución que el motor de flujo de trabajo asocia a la ejecución de actividades de espera de suceso. Por último, también se detallan los objetos de datos de flujo de trabajo `Context_Event` y `Context_Deadline` disponibles para su uso en transiciones desde actividades de espera de suceso.

Capítulo 9, “Manual”, en la página 61 describe los detalles de los metadatos asociados a una actividad manual. Esto incluye los detalles de la tarea manual, la estrategia de asignación, las asociaciones de objeto de negocio y la espera de suceso asociada a la actividad manual. También se describen las validaciones

ejecutadas cuando se crean o modifican los metadatos de una actividad manual. También se describe en detalle la información de tiempo de ejecución que el motor de flujo de trabajo asocia a la ejecución de actividades manuales. Por último, también se proporciona una descripción del objeto de datos de flujo de trabajo `Context_Task` que está disponible para su uso en las diversas correlaciones asociadas a una actividad manual.

Capítulo 10, "Decisión", en la página 77 describe los detalles de los metadatos asociados a una actividad de decisión. Estos metadatos incluyen los detalles de la tarea de decisión (que son similares a los detalles de la tarea de actividad manual) y los detalles de pregunta de las preguntas de opciones múltiples y de texto libre. También se describen las diversas validaciones que se ejecutan al crear o modificar los detalles de actividad o pregunta asociados a una actividad de decisión Este capítulo también incluye una descripción de la información de tiempo de ejecución que esté presente cuando el motor de flujo de trabajo ejecuta una actividad de decisión. También se proporciona en este capítulo una descripción del objeto de datos de flujo de trabajo `Context_Decision`.

Capítulo 11, "Subflujo", en la página 87 describe los detalles de los metadatos asociados a una actividad de subflujo. Esto incluye los detalles del proceso de subflujo asociado a la actividad de subflujo y las correlaciones de entrada necesarias para promulgar dicho proceso de subflujo. Existen varias validaciones que se ejecutan al crear o modificar estos metadatos, y también se proporciona una descripción de de las mismas.

Capítulo 12, "Inicio de bucle y fin de bucle", en la página 91 describe los detalles de los metadatos asociados a una actividad de inicio y de fin de bucle. Se describen el tipo de bucle, la condición de bucle y la referencia de la actividad de fin de bucle de una actividad de inicio de bucle. Este capítulo también incluye una descripción de la información de tiempo de ejecución que esté presente cuando el motor de flujo de trabajo ejecuta un bucle en una definición de proceso de flujo de trabajo. También se proporciona en este capítulo una descripción del objeto de datos de flujo de trabajo `Context_Loop`.

Capítulo 13, "Paralela", en la página 95 describe los detalles de los metadatos asociados a una actividad paralela. Las actividades paralelas envuelven tipos de actividad existentes, incluidas las actividades Capítulo 9, "Manual", en la página 61 y las actividades de Capítulo 10, "Decisión", en la página 77. Puesto que los metadatos asociados a estos tipos de actividad siguen siendo los mismos, no se volverán a describir en este capítulo. También se describen las validaciones ejecutadas cuando se crean o modifican los metadatos de una actividad paralela. También se describe en detalle la información de tiempo de ejecución que el motor de flujo de trabajo asocia a la ejecución de actividades paralelas. Por último, también se proporciona una descripción del objeto de datos de flujo de trabajo `Context_Parallel` que está disponible para su uso en las diversas correlaciones asociadas a una actividad paralela.

Capítulo 14, "Notificaciones de actividad", en la página 101 describe los detalles de los metadatos asociados a una notificación de actividad. Estos detalles incluyen el mecanismo de entrega, el asunto, el cuerpo, la estrategia de asignación y las acciones asociadas a la notificación. Existen una serie de validaciones que se ejecutan al crear o modificar los metadatos de notificación, y estas también se describen en este capítulo. También se proporciona una descripción de la información de tiempo de ejecución cuando el motor de flujo de trabajo crea una notificación. Por último, hay una serie de detalles de implementación que son necesarios en la aplicación Cúram para permitir que las notificaciones se entreguen correctamente. Estos también se describen en este capítulo.

1.4.4 Control de flujo

Un proceso de flujo de trabajo modela el flujo de información a través de una organización, que pasa por etapas llevadas a cabo por agentes humanos o por software informático para alcanzar un objetivo empresarial. La sección *Flujo de control* del documento detalla cómo se especifica y gestiona dicho flujo de información (entre actividades) en el WMS de Cúram.

Capítulo 15, "Transiciones", en la página 109 describe los enlaces entre actividades. Los metadatos asociados a las transiciones se describen en detalle. También se tratan las validaciones que corresponden

a la creación y modificación de transiciones. También se describe la información de tiempo de ejecución que el motor de flujo de trabajo asocia al procesamiento de transiciones.

Capítulo 16, “Condiciones”, en la página 113 describe la construcción de metadatos definición de proceso que representa una condición. También se tratan las validaciones que corresponden a la creación y modificación de condiciones.

Capítulo 17, “División/Unión”, en la página 119 describe los metadatos asociados a las divisiones y uniones de actividades, cuándo deben utilizarse y los diferentes tipos disponibles.

Capítulo 18, “Estructura del flujo de trabajo”, en la página 121 describe la estructura de un proceso de flujo de trabajo que viene determinada por las actividades del proceso y las transiciones entre ellas. Se describen las restricciones que se aplican al construir una definición de proceso para garantizar que sea una estructura de bloque válida, así como las validaciones que se ejecutan como parte de dichas restricciones.

1.4.5 Desarrollo y entorno de ejecución

La sección *Desarrollo y entorno de ejecución* del documento describe los aspectos específicos del entorno de desarrollo y ejecución de los flujos de trabajo de Cúram. En concreto, detalla cómo ejecutar, configurar y depurar flujos de trabajo.

Capítulo 19, “Servicios web de flujo de trabajo”, en la página 127 describe los pasos necesarios para permitir la promulgación de procesos mediante servicios web exponiendo el proceso de flujo de trabajo de Cúram como un servicio web.

Capítulo 20, “Ubicaciones de archivo”, en la página 131 detalla dónde se exportan las distintas salidas de programas de utilidad tales como la herramienta de definición de procesos y otras interfaces de usuario de administración y cómo se controlan las versiones. Dichas salidas incluyen archivos de metadatos de definición de procesos y también los archivos fuente asociados a sucesos.

Capítulo 21, “Configuración”, en la página 133 describe las propiedades de aplicación relacionadas con el flujo de trabajo, sus nombres, sus valores predeterminados y para qué se utilizan en el sistema de flujo de trabajo de Cúram.

Capítulo 22, “JMSLite”, en la página 135 detalla el servidor ligero JMS de Cúram que puede ejecutar junto con el entorno de pruebas RMI en un entorno de desarrollo integrado (IDE) soportado. Se describen los pasos necesarios para iniciar el servidor JMSLite, así como una descripción detallada de cómo depurar flujos de trabajo utilizando JMSLite.

1.4.6 Personalización y configuración de la bandeja de entrada

La sección *Configuración y personalización de la bandeja de entrada* del documento describe las opciones de configuración y personalización disponibles en las secciones Bandeja de entrada y Gestión de tareas del WMS de Cúram. En concreto, detalla cómo configurar el número de tareas que se muestran en las diversas listas que aparecen en la bandeja de entrada, y también cómo personalizar las diversas acciones de Bandeja de entrada y Gestión de tareas disponibles en el sistema.

Capítulo 23, “Bandeja de entrada y gestión de tareas”, en la página 137 describe las opciones de configuración disponibles para su uso en la bandeja de entrada. También detalla cómo personalizar las funciones disponibles de Bandeja de entrada y de Gestión de tareas mediante el uso de la infraestructura Guice de Google.

Capítulo 2. Creación de un proceso de flujo de trabajo

2.1 Ciclo de vida de una definición de proceso

La definición de proceso es el concepto central en cualquier sistema de flujo de trabajo, así que el modo en que se crea y utiliza es de gran importancia. En este capítulo se describen los recursos que proporciona el sistema de flujo de trabajo de Cúram para crear y administrar definiciones de proceso.

2.1.1 Creación de un proceso

Cúram workflow system proporciona una *herramienta de definición de procesos* (PDT en sus siglas inglesas) para crear y mantener definiciones de proceso que luego pueda interpretar el motor de flujo de trabajo. La creación de una definición de proceso implica la utilización de la herramienta de definición de procesos para describir el comportamiento deseado del proceso en cuanto a actividades y transiciones.

Como parte de la herramienta de definición de procesos, se proporciona una serie de programas de utilidad que pueden asistir en la creación de procesos. La PDT permite que una definición de proceso se visualice durante el diseño. Los procesos también se pueden copiar, importar y exportar utilizando la PDT.

2.1.2 Visualización de un proceso

Se proporciona un programa de utilidad gráfica de sólo lectura como parte de la herramienta de definición de procesos que permite a los administradores de procesos visualizar los procesos a medida que se crean o modifican. Esta herramienta permite a los administradores ver todas las actividades y transiciones de una definición de proceso y proporciona una vista de alto nivel de todas las rutas posibles a través del proceso de flujo de trabajo durante la ejecución. A continuación se muestra un ejemplo de una representación gráfica de una definición de proceso de flujo de trabajo.

Figura 1. Visualización de la definición de proceso de flujo de trabajo de cierre de un caso

El proceso visualizado consta de una serie de nodos en un gráfico que representa las actividades del proceso. Los nodos están enlazados mediante flechas que representan las transiciones definidas en la definición de proceso. Al pulsar en una actividad del gráfico, se muestran los detalles de la actividad en la PDT. Asimismo, al pulsar en una transición entre actividades del gráfico, se mostrarán los detalles de dicha transición en la PDT.

La herramienta gráfica muestra la siguiente información de cada proceso visualizado:

- Tipo y nombre de cada actividad. Cada tipo de actividad se identifica mediante un icono específico.
- Las notificaciones definidas en cada actividad (consulte Capítulo 14, “Notificaciones de actividad”, en la página 101). Si una actividad tiene una notificación asociada, se representará como un sobre en el que se puede pulsar para ir la página de notificación de la actividad asociada.
- El tipo división/unión (consulte Capítulo 17, “División/Unión”, en la página 119) de cada actividad. Una división o unión de tipo "selección" en una actividad se representa mediante un círculo, mientras que una división o unión de tipo "paralela" se representa mediante un cuadrado.
- Las transiciones entre actividades. Cuando una transición entre actividades tiene una condición de transición asociada (consulte Capítulo 16, “Condiciones”, en la página 113), esto se representa mediante un asterisco. Los detalles de la condición se visualizan cuando se pasa el ratón por encima del asterisco.

- El orden de cada división de selección (consulte Capítulo 17, “División/Unión”, en la página 119) que parte de una actividad. Como el orden de una división de selección de una actividad es importante (se seguirá la primera transición elegible de la lista), el orden de cada transición de la actividad se visualiza como un número en dicha transición.

2.1.3 Publicación de un proceso

Una vez que una definición de proceso se ha creado y está lista para su uso, deberá publicarse antes de que el motor de flujo de trabajo pueda ejecutarla (consulte 2.2, “Ejecución de un proceso”, en la página 8). Cuando se va a publicar un proceso utilizando la PDT, se examina para garantizar que toda la información que el motor necesita para ejecutar el proceso está presente y es coherente internamente. Las validaciones necesarias para publicar un proceso se describen en las distintas secciones que tratan de metadatos de este documento.

Sólo aquellos procesos que hayan pasado todas las validaciones necesarias podrán publicarse y quedar a disposición del motor de flujo de trabajo. Una vez publicada una definición de proceso, se vuelve de sólo lectura y ya no podrá editarse con la herramienta de definición de procesos sin crear una nueva versión.

2.1.4 Versiones de un proceso (edición de procesos)

Con el tiempo, puede ser necesario modificar un proceso publicado, pero, como un proceso publicado es de solo lectura, será necesaria una nueva versión antes de que se puedan aplicar las modificaciones. Cuando se intenta editar un proceso publicado en la PDT, se crea de forma automática una nueva versión sin publicar de dicho proceso.

Solo puede haber una versión sin publicar de un proceso en un momento dado. Si el administrador desea editar un proceso publicado, deberán publicarse o suprimirse antes las versiones no publicadas que existan.

2.1.5 Importación, exportación y copia de un proceso

La funcionalidad de importación y exportación permite a los desarrolladores mover definiciones de proceso según sea necesario. Por ejemplo, una definición de proceso podría desarrollarse en un sistema de desarrollo y moverse a producción solo cuando se hayan completado las pruebas.

Cuando se exporta un proceso, se exportan los metadatos de dicho proceso al sistema de archivos. Estos metadatos se podrán importar luego utilizando la opción de importación de proceso de la PDT. A un proceso importado de esta forma se le asignará el número de versión más alto disponible y se anulará su publicación independientemente de su estado de publicación en el momento de su importación. Esto se hace así para garantizar que las definiciones de proceso importadas estén sujetas a las mismas validaciones de publicación que las definiciones desarrolladas localmente. Al importar, existe una opción de sobrescritura que garantiza que cualquier versión existente del proceso no publicada se sobrescriba con la versión importada.

Pueden darse situaciones en las que una definición de proceso solo difiera ligeramente de otra en el sistema de flujo de trabajo. Existe una opción de `copia` de procesos que permite copiar un proceso existente en un nuevo proceso cuando sea necesario. Siempre se anulará la publicación de un nuevo proceso cuando se copie con una versión de 1, independientemente del estado del proceso original.

2.1.5.1 Validaciones

- Una definición de proceso no se podrá importar si ya existe una versión no publicada de un proceso con el mismo nombre y no se ha seleccionado la opción de sobrescritura.
- Una definición de proceso no se podrá importar si no se ha especificado un nombre para ese proceso.
- Una definición de proceso no se podrá importar si ya existe un proceso con el mismo nombre y con distinto identificador de proceso. Esta validación garantiza que una definición importada no pueda sobrescribir de forma inadvertida una definición de proceso existente a menos que coincidan los identificadores de proceso.

- Cuando se copia un proceso existente, el nombre del nuevo proceso deberá ser exclusivo dentro del sistema de flujo de trabajo.
- La longitud del nombre de la definición de proceso de flujo de trabajo que se importe no deberá exceder la longitud máxima permitida para ese nombre. Dicha longitud es de 254 caracteres.
- La longitud de los nombres de cualquiera de los objetos de dato de flujo de trabajo contenidos en la definición de proceso de flujo de trabajo que se importe no deberá exceder la longitud máxima permitida para esos nombres. Dicha longitud es de 75 caracteres.
- La longitud de los nombres de cualquiera de los atributos de objeto de dato de flujo de trabajo contenidos en la definición de proceso de flujo de trabajo que se importe no deberá exceder la longitud máxima permitida para esos nombres. Dicha longitud es de 75 caracteres.
- Los valores de la tabla de códigos contenidos en la definición de proceso de flujo de trabajo que se está importando deberán ser válidos (es decir, la tabla de códigos debe existir y el código especificado debe existir en dicha tabla de códigos).
- Deberá haber texto para el entorno local predeterminado en todas las cadenas de texto localizables especificadas en la definición de proceso que se importe.
- Los identificadores de actividades, transiciones, expresiones de condición de transición, expresiones de condición de bucle, sucesos y recordatorios deberán ser exclusivos en la definición de proceso de flujo de trabajo que se importe.

2.1.6 Localización

Las definiciones de proceso de flujo de trabajo contienen texto de metadatos que distintos usuarios deben visualizar en distintos idiomas. Por ejemplo, cuando se ejecuta una actividad manual, crea una tarea que tiene un asunto asociado. La herramienta de definición de procesos permite al desarrollador de procesos localizar esta cadena del asunto para cada uno de los entornos locales que soporta la aplicación.

Las cadenas localizables pueden identificarse en una definición de proceso mediante los metadatos especificados en 6.2.1, “Texto localizado”, en la página 32. Todas las cadenas de texto localizables especificadas en una definición de proceso debe tener una entrada para el entorno local de servidor predeterminado. El entorno local de servidor predeterminado se especifica en la aplicación Cúram mediante la propiedad: curam.environment.default.locale. De forma predeterminada, la PDT utiliza este entorno local al añadir cadenas localizadas a una definición de proceso. Todos los demás entornos locales obligatorios deberán añadirse utilizando la opción de localizar proporcionada.

Lo siguiente es una lista de las cadenas de texto localizables que pueden especificarse en una definición de proceso.

- Descripción del proceso
- Nombre de visualización del objeto de datos de flujo de trabajo
- Descripción del objeto de datos de flujo de trabajo
- Nombre de visualización del atributo del objeto de datos de flujo de trabajo
- Nombre de la actividad
- Descripción de la actividad
- Mensaje de tarea de actividad manual
- Mensaje de acción de tarea de actividad manual
- Mensaje de tarea de actividad manual paralela
- Mensaje de acción de tarea de actividad manual paralela
- Mensaje de acción de actividad de decisión
- Mensaje de pregunta de actividad de decisión
- Mensaje de acción secundaria de actividad de decisión
- Valor de visualización de respuesta de actividad de decisión
- Mensaje de acción de actividad de decisión paralela

- Mensaje de pregunta de actividad de decisión paralela
- Mensaje de acción secundaria de actividad de decisión paralela
- Valor de visualización de respuesta de actividad de decisión paralela
- Mensaje del asunto de una notificación de actividad
- Mensaje del cuerpo de una notificación de actividad
- Mensaje de la acción de una notificación de actividad
- Mensaje del asunto de una notificación de recordatorio
- Mensaje del cuerpo de una notificación de recordatorio
- Mensaje de la acción de una notificación de recordatorio

El API de `LocalizableStringResolver` proporciona rutinas que resuelven y devuelven las distintas cadenas localizables de las tareas y notificaciones que existen en una definición de proceso de flujo de trabajo del entorno local del usuario actual. Cuando una cadena de texto no se localice para el entorno local del usuario actual, se devolverá en su lugar el texto del entorno local de servidor predeterminado.

2.2 Ejecución de un proceso

Una definición de proceso de flujo de trabajo describe las tareas y el flujo de un proceso de negocio en términos que comprenda Cúram Workflow Management System. Para llevar a cabo el trabajo descrito en la definición de proceso especificada, el motor de flujo de trabajo deberá crear y ejecutar una instancia de la misma. En esta sección se describe el mecanismo mediante el cual se hace esto. Una instancia de proceso puede considerarse como los datos en tiempo de ejecución de una definición de proceso de flujo de trabajo promulgada.

2.2.1 Comportamiento básico del motor

Cúram Workflow Management System incluye un motor de flujo de trabajo que proporciona el entorno de ejecución para una instancia de proceso. Existen varios mecanismos disponibles para promulgar un proceso de flujo de trabajo, descritos en 19.2.1, “Promulgación de procesos”, en la página 127. Cuando se promulga un proceso, el motor de flujo de trabajo examina la tabla de base de datos relevante y utiliza la *versión publicada más reciente* de la definición de proceso especificada para crear la instancia de proceso a ejecutar.

A medida que se ejecuta cada actividad, el motor de flujo de trabajo crea y gestiona un registro de instancia de actividad asociada. Este registro contiene los datos de tiempo de ejecución de una instancia de actividad del flujo de trabajo promulgado. A medida que el flujo de trabajo progresa, el motor evalúa las transiciones (consulte Capítulo 15, “Transiciones”, en la página 109) de las diversas actividades para decidir qué ruta seguir a través del proceso. Esto implica determinar los tipos de divisiones y uniones (consulte Capítulo 17, “División/Unión”, en la página 119) que la actividad posee, y también ejecutar cualquier condición (consulte Capítulo 16, “Condiciones”, en la página 113) que puedan tener las diversas transiciones en el proceso. El motor también crea y gestiona los registros de instancia de transición (que contienen los datos de tiempo de ejecución de una transición de flujo de trabajo) de cada transición seguida en el proceso del flujo de trabajo.

2.2.2 Ejecución de varias versiones

La modificación y liberación de una nueva versión de un proceso no afectará a ninguna instancia de dicho proceso que esté actualmente en ejecución. Un proceso se ejecutará hasta su finalización en el motor de flujo de trabajo con la versión con la que se haya iniciado, independientemente de las versiones posteriores que se publiquen.

2.2.3 Administración de instancias de proceso

Un administrador de flujos de trabajo tiene la capacidad de influir en la ejecución de una instancia de proceso en ejecución mediante la interfaz de administración de flujos de trabajo de Cúram. Las funciones siguientes están disponibles para este fin:

Suspensión de una instancia de proceso

Cualquier instancia de proceso que esté ejecutando en un momento dado puede suspenderse. Cuando esto ocurre, el motor de flujo de trabajo permitirá que se completen todas las instancias de actividad que estén *en curso* dentro de esa instancia de proceso. Sin embargo, motor de flujo de trabajo iniciará y suspenderá inmediatamente el siguiente conjunto de actividades que es necesario ejecutar para dicha instancia de proceso. El motor de flujo de trabajo también suspenderá los procesos de subflujo asíncronos (consulte Capítulo 11, “Subflujo”, en la página 87) asociados a la instancia de proceso que se suspende.

Reanudación de una instancia de proceso

Se podrá reanudar cualquier instancia de proceso de flujo de trabajo que se haya suspendido. Cuando esto ocurre, el motor de flujo de trabajo reiniciará las instancias de actividad de dicho proceso que estaban suspendidas previamente. Asimismo, el motor de flujo de trabajo reanudará los procesos de subflujo suspendidos (consulte Capítulo 11, “Subflujo”, en la página 87) asociados a esa instancia de proceso.

Anulación de una instancia de proceso

Cualquier instancia de proceso que esté ejecutando o suspendida en un momento dado puede anularse. Se completarán todas las actividades que estén *en curso* en la instancia de proceso anulada. Si el proceso contiene actividades manuales o de decisión en curso, el motor de flujo de trabajo cerrará las tareas asociadas cuando se anule la instancia de proceso. El motor de flujo de trabajo no iniciará ninguna actividad asociada a una instancia de proceso anulada. Asimismo, se anularán los procesos de subflujo síncronos (consulte Capítulo 11, “Subflujo”, en la página 87) asociados a la instancia de proceso. Una instancia de proceso anulada no se podrá reanudar.

2.3 Biblioteca de referencia de métodos

Existen varias situaciones en Cúram Workflow Management System en las que es necesario interactuar con la aplicación de Cúram llamando a algún proceso de negocio o a métodos de entidad (consulte 7.3, “Métodos de negocio de Cúram”, en la página 35 para obtener un ejemplo de tal interacción). El motor de flujo de trabajo puede invocar cualquier objeto de proceso de negocio (BPO) o método de entidad en la aplicación. Sin embargo, hay demasiados métodos de estos como para presentárselos a un diseñador de procesos para que los utilice en sus definiciones de proceso de una forma aceptable. La finalidad de esta biblioteca es permitir que un administrador asigne los métodos que con más probabilidad se utilizan en las definiciones de proceso a una lista más manejable para su uso en el diseño de procesos. Por supuesto, no es necesario rellenar previamente la biblioteca con todos los métodos que podrían utilizarse en el futuro. Pueden ir añadiéndose nuevos métodos a la biblioteca a medida que vayan haciendo falta.

2.3.1 Referencia a métodos Cúram

Los métodos de un objeto de proceso de negocio (BPO) de Cúram o de una entidad deben añadirse a la biblioteca de referencia de métodos antes de que puedan referenciarse en una definición de proceso. El tipo del método definido al añadirlo a la biblioteca determinará dónde estará disponible dicho método para su uso en una definición de proceso.

Tenga en cuenta que la eliminación de una referencia de método de la biblioteca de referencia de métodos no lo eliminará de las definiciones de proceso que lo referencien. Mientras el método siga siendo un método de aplicación Cúram válido, las definiciones de proceso que lo referencien seguirán siendo válidas.

2.3.2 Tipos de método

Un método de un objeto de proceso de negocio (BPO) de Cúram o de una entidad debe añadirse a la biblioteca de referencia de métodos con uno de los tres tipos de método definidos. Un método puede estar asociado a más de un tipo de método, pero el método tendrá que añadirse una vez por cada tipo de método distinto. A continuación se detallan los distintos tipos de método en la biblioteca de referencia de métodos, junto con las restricciones que se aplican a su uso.

General

Los métodos de tipo *General* sólo están disponibles como métodos de aplicación que se invocan desde actividades automáticas (consulte 7.3, "Métodos de negocio de Cúram", en la página 35). La herramienta de definición de procesos limita el acceso a estos métodos cuando se selecciona un método que debe invocarse desde una actividad automática.

Asignación

Los métodos de la biblioteca de tipo *Asignación* sólo están disponibles para su uso como funciones de estrategia de asignación asociadas a actividades manuales, actividades de decisión, actividades paralelas y notificaciones de actividad. (Consulte 9.4, "Estrategia de asignación", en la página 67). Todos los métodos especificados con un tipo de método de asignación deben tener un tipo de retorno de `curam.util.workflow.struct.AllocationTargetList`.

Fecha límite

Los métodos de tipo *Fecha límite* en la biblioteca de métodos solo pueden referenciarse como métodos manejadores de fecha límite asociados a actividades de espera de suceso, manuales, de decisión y paralelas.(Consulte 8.4, "Fecha límite", en la página 52).

2.4 Plantillas WDO

Los datos se mantienen y circulan en el motor de flujo de trabajo como objetos de datos de flujo (consulte Capítulo 4, "Objetos de datos de flujo de trabajo", en la página 17). Los objetos de datos de flujo de trabajo que un proceso puede utilizar se definen en la propia definición de proceso. Sin embargo, cabe suponer que algunos objetos de datos de flujo de trabajo sean útiles en muchas definiciones de proceso. Por lo tanto, convendría que pudieran importarse de alguna agrupación en lugar de tener que volver a crearse en cada proceso individual. Tal es la finalidad de esta biblioteca.

2.4.1 Metadatos

```
<wdo is-list-wdo="false" initialize-attributes="false">
    <wdo-name>TaskCreateDetails</wdo-name>
    <display-name>
        <localized-text>
            <locale language="es">NombreDetallesCreacionNombre</locale>
        </localized-text>
    </display-name>
    <description>
        <localized-text>
            <locale language="es">Plantilla de WDO de detalles de creación
              de tarea</locale>
        </localized-text>
    </description>
    <attributes>
        <attribute>
            <attribute-name>subject</attribute-name>
            <display-name>
                <localized-text>
                    <locale language="es">Asunto de la tarea</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="es">Fecha de vencimiento de la tarea</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>
```

Los metadatos definidos para plantillas de objeto de datos de flujo de trabajo son exactamente los mismos que los definidos para los objetos de datos de flujo de trabajo. Para obtener una descripción completa de estos metadatos, consulte Capítulo 4, “Objetos de datos de flujo de trabajo”, en la página 17. La biblioteca de plantillas de objeto de datos de flujo de trabajo se almacena en la tabla de base de datos WDOTemplateLibrary.

Tenga en cuenta que el elemento `initialize-attributes` de un objeto de datos de flujo de trabajo y los elementos `required-enactment`, `process-input` y `constant-value` de un atributo de objeto de datos de flujo de trabajo no están disponibles para su edición en las plantillas de objeto de datos de flujo de trabajo y se inicializan automáticamente a sus valores predeterminados en los metadatos asociados.

2.4.2 Importación y sincronización

Las plantillas definidas en la biblioteca de plantillas de objeto de datos de flujo de trabajo están disponibles para su uso al crear definiciones de proceso. La importación de una plantilla de objeto de datos de flujo de trabajo de la biblioteca añadirá el objeto de datos de flujo de trabajo y todos sus atributos a la definición de proceso actual.

Una vez que una plantilla de objeto de datos de flujo de trabajo se ha importado en una definición de proceso, podrá sincronizarse en cualquier momento con su entrada correspondiente en la biblioteca de

plantillas de objeto de datos de flujo de trabajo. La sincronización de la plantilla de una definición de proceso forzará que el nombre y el nombre de visualización del objeto de datos de flujo de trabajo se actualicen a partir de la biblioteca de plantillas. Junto a esto, cualquier entrada de atributo nueva que exista en la entrada de biblioteca de plantillas se añadirá de forma automática al objeto de datos de flujo de trabajo en la definición de proceso. El usuario puede optar por sustituir los atributos existentes en el objeto de datos de flujo de trabajo con los que están en la biblioteca de plantillas cuando se sincronice. Debe tenerse en cuenta que la sustitución de atributos puede invalidar la definición de proceso y requerir actualizaciones donde se hayan utilizado los valores de atributo antiguos.

2.4.3 Validaciones

- Un objeto de datos de flujo de trabajo no podrá importarse de una plantilla si ya existe uno con ese nombre en la definición de proceso de flujo de trabajo asociada.

Capítulo 3. Metadatos de una definición de proceso

3.1 Descripción general

El proceso es el concepto de nivel superior en una definición de proceso. Básicamente contiene información para identificar y describir la definición de proceso. Esta información incluye el identificador y la versión de la definición de proceso, su nombre y una breve descripción. También incluye una descripción de la estrategia de asignación de errores que se puede especificar para un proceso. Las secciones siguientes describen esta información de nivel superior.

3.2 Metadatos

```
<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="es">Este proceso de flujo de trabajo puede
        promulgarse para aprobar un elemento planificado.</locale>
      </localized-text>
    </description>
    <documentation>Consulte la documentación de aprobación
                   de un elemento.
    </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

Este elemento es la etiqueta padre de todos los metadatos de definición de proceso.

id Es un identificador de 64 bits proporcionado por el servidor de claves de Cúram cuando se crea un proceso en la herramienta de definición de procesos. El identificador de proceso debe ser exclusivo en el sistema de flujo de trabajo de Cúram. El motivo de esto es que el identificador de proceso junto con el número de versión de proceso es la forma que tiene el motor de flujo de trabajo de distinguir un registro de definición de proceso de otro a efectos de lecturas de base de datos.

process-version

Este número representa la versión de una definición de proceso de flujo de trabajo. Un registro de definición de proceso de flujo de trabajo se identifica de forma exclusiva mediante su identificador y número de versión. Una definición de proceso puede tener muchas versiones publicadas y una versión en edición. Una vez publicada una definición de proceso, se crea una versión nueva y ya no podrá actualizarse. Las actualizaciones posteriores requerirán la creación de una nueva versión, y dicha versión no estará activa

mientras no se publique. Cuando se promulga un proceso, se utiliza el número de versión mas alto. Las instancias de proceso que comienzan con un determinado número de versión permanecen vinculadas a dicha versión hasta la terminación.

language-version
Los metadatos de definición de proceso son el lenguaje de flujo de trabajo de Cúram. A medida que se añaden nuevas mejoras y características, este lenguaje puede cambiar. Este número de versión permitirá al motor de flujo de trabajo ejecutar antiguas versiones del lenguaje diferentes de las nuevas o, lo que es más probable, herramientas de actualización que conviertan antiguas definiciones de proceso a nuevas versiones del lenguaje.

released
Representa un distintivo booleano que indica si la definición de proceso se ha publicado. Sólo las definiciones de proceso que se hayan publicado podrán promulgarse o seleccionarse como subprocesos de una actividad de subflujo (consulte Capítulo 11, "Subflujo", en la página 87).

category
Una definición de proceso debe colocarse en una categoría. La categoría debe seleccionarse en la herramienta de definición de procesos y se obtiene de la tabla de códigos `ProcessCategory`. Este atributo se utiliza en la funcionalidad de búsqueda de definiciones de proceso y no tiene ningún efecto funcional sobre la actividad.

createdBy
Representa el nombre del usuario que ha creado la definición de proceso de flujo de trabajo. Este atributo se utiliza en la funcionalidad de búsqueda de definiciones de proceso y no tiene ningún efecto funcional sobre la actividad.

creationDate
Representa la fecha y hora de creación de la definición de proceso de flujo de trabajo. Este atributo se utiliza en la funcionalidad de búsqueda de definiciones de proceso y no tiene ningún efecto funcional sobre la actividad.

name El nombre de la definición de proceso es la forma que tiene el proceso de identificarse a efectos de promulgación. El servicio de promulgación (el API utilizada para promulgar un proceso por código) identifica el proceso a promulgar por su nombre. Como tal, este nombre debe ser exclusivo dentro del sistema de flujo y no se puede cambiar una vez creado el proceso. Puesto que el nombre de proceso es una constante, no es localizable como un nombre de actividad.

description
Un proceso también puede tener una descripción opcional que especifique brevemente lo que hace el proceso para facilitar las cosas a quienes editen la definición del proceso en el futuro. Se trata de un campo de texto localizable en el mismo formato que todos los campos localizables de una definición de proceso (consulte 6.2.1, "Texto localizado", en la página 32).

documentation
Un proceso también puede tener un enlace a alguna documentación que explique el proceso de forma más descriptiva. Es un campo de texto de formato libre donde el desarrollador puede especificar el nombre de un documento pertinente al proceso de flujo de trabajo o incluso un enlace a dicho documento.

web-service
Este elemento opcional describe los detalles del servicio web de un proceso de flujo de trabajo. Un proceso se puede marcar como servicio web definiendo este valor de metadato, que indica que el proceso debe exponerse como servicio web. Esto permite que el proceso pueda participar en un proceso orquestado BPEL (Business Process Execution Language) y significa que el proceso puede ser llamado desde un proceso BPEL. Pueden consultarse detalles adicionales sobre esta funcionalidad en Capítulo 19, "Servicios web de flujo de trabajo", en la página 127.

expose
Este atributo representa un distintivo booleano que indica si la definición de proceso debe

exponerse como servicio web. Una definición de proceso de flujo de trabajo no está expuesta como servicio web de forma predeterminada.

callback-service
Esto es un elemento opcional porque no todas las invocaciones desde un proceso BPEL requieren una devolución de llamada. El valor es el nombre completo de una clase que extiende la clase `org.apache.axis.client.Service` (que forma parte del servicio (API de Axis) del proyecto Apache Axis). La clase `org.apache.axis.client.Service` la genera la funcionalidad del conector de servicios web de Cúram para servicios web de salida.

failure-allocation-strategy
Un proceso también puede tener especificada una estrategia de asignación de errores opcional. Cuando se asigna una tarea (asociada a una actividad Capítulo 9, "Manual", en la página 61 o Capítulo 10, "Decisión", en la página 77), el motor de flujo de trabajo invoca la estrategia de asignación asociada para recuperar la lista de destinos de asignación. Si esta invocación no devuelve ningún destino de asignación, el motor de flujo de trabajo comprobará entonces la presencia de una estrategia de asignación de errores y utilizará dicha estrategia para intentar asignar la tarea. Puesto que la estrategia de asignación de tipo *TARGET* especifica un destino de asignación directamente, nunca habrá necesidad de una retrotracción a la estrategia de asignación de errores. La estrategia de asignación de errores es una estrategia que abarca a todo el proceso y, si se especifica, se utilizará en todas las actividades manuales y de decisión del proceso cuando sea necesario.

allocation-strategy
Este elemento describe la estrategia de asignación de errores que se utiliza en el proceso. La estrategia de asignación de errores debe ser de tipo *TARGET*. Si el programa de resolución de trabajo no puede asignar la tarea a un usuario, a un objeto organizativo (p.ej. unidad organizativa, puesto o trabajo) o a una cola de trabajos utilizando el destino de asignación especificado, se asignará la tarea a la cola de trabajos predeterminada. El atributo identificador representa el identificador del destino de asignación utilizado como estrategia de asignación de errores.

3.3 Validaciones

- Un proceso de flujo de trabajo deberá tener un nombre de proceso exclusivo. Esto significa que un proceso no se podrá crear si el nombre del proceso está vacío o si ya existe un proceso con el mismo nombre.
- Un proceso de flujo de trabajo es necesario para especificar una categoría.
- Una versión publicada del proceso de flujo de trabajo no se podrá suprimir una vez promulgada. Esto es necesario porque, incluso existiendo una versión más reciente de un proceso, las instancias de proceso que están en curso cuando la nueva versión pasa a estar disponible ejecutan hasta el final con la versión con que fueron iniciadas. Las definiciones de proceso son también un registro histórico necesario del que se obtiene la información de auditoría.
- Una versión publicada de un proceso de flujo de trabajo no se podrá suprimir si está referenciado por una actividad de subflujo en una versión publicada de otro proceso, donde dicha versión publicada es la última versión publicada.
- Si se ha especificado una estrategia de asignación de errores para el proceso de flujo de trabajo, su tipo deberá ser *TARGET*.
- El nombre de la clase de servicio de devolución de llamada no se podrá especificar si el proceso de flujo de trabajo no se ha expuesto como un servicio web.
- El nombre de la clase de servicio de devolución de llamada deberá representar una clase que pueda encontrarse en la vía de acceso de clases de la aplicación.
- El nombre de la clase de servicio de devolución de llamada deberá representar una clase que extienda la clase `org.apache.axis.client.Service`.

3.4 Descripción de WDO de contexto

Cierta información genérica de entorno de ejecución relativa al motor de flujo de trabajo debe ponerse a descripción de las actividades y transiciones durante el tiempo de vida de una instancia de proceso. Los detalles del objeto de datos de flujo de trabajo `Context_RuntimeInformation` que proporciona dicha información pueden consultarse en la siguiente ubicación: 4.4, “Lista de WDO de contexto”, en la página 21.

Capítulo 4. Objetos de datos de flujo de trabajo

4.1 Descripción general

Los datos se mantienen y se transfieren en el motor de flujo de trabajo como objetos de datos de flujo de trabajo y objetos de datos de flujo de trabajo de lista. Estos son objetos lógicos definidos en la definición de proceso que tienen un nombre y una lista de atributos de varios tipos a los que se les pueden asignar datos. Son conceptualmente similares a los objetos en los lenguajes de programación, aunque, por supuesto, se manifiestan en el sistema de flujo de trabajo de forma bastante distinta. Los valores de objeto de datos de flujo de trabajo se pueden grabar en la promulgación de proceso o desde la salida de varios tipos de actividad.

Las instancias de objeto de datos de flujo de trabajo y las instancias de objeto de datos de flujo de trabajo de lista existen tan pronto como se promulga el proceso y hasta que el proceso se completa. Como tales, están disponibles para ser utilizados en las actividades y en las transiciones a lo largo del tiempo de vida de dicha instancia de proceso. Por lo tanto, es responsabilidad del diseñador de procesos garantizar que los atributos de los objetos de datos de flujo de trabajo se informen antes de que se utilicen. Los intentos de utilizar los atributos de objeto de datos de flujo de trabajo antes de que se informen darán como resultado errores en tiempo de ejecución.

4.2 Metadatos

```
<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="es">Detalles de creación de tarea</locale>
          </localized-text>
        </display-name>
        <description>
          <localized-text>
            <locale language="es">Este objeto de datos de flujo de trabajo
            contiene los atributos necesarios para la
            creación manual de una tarea.</locale>
          </localized-text>
        </description>
        <attributes>
          <attribute>
            <attribute-name>subject</attribute-name>
            <display-name>
              <localized-text>
                <locale language="es">Asunto de la tarea</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="es">ID del rol del participante</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="es">Fecha límite</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="es">Duración de la fecha límite</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 Este elemento es opcional (ya que una definición de proceso de flujo de trabajo no tiene por qué contener objetos de datos de flujo de trabajo) y contiene los detalles de todos los objetos de datos de flujo de trabajo definidos para la definición de proceso de flujo de trabajo.

wdo Contiene los detalles de uno de los objetos de datos de flujo de trabajo definidos para la definición del proceso de flujo de trabajo. Esto incluye los detalles genéricos del propio objeto de datos de flujo de trabajo y también los detalles de cada uno de sus atributos. Los metadatos que describen un objeto de datos de flujo de trabajo y sus atributos se describen a continuación:

is-list-wdo
Contiene un valor BOOLEAN que indica si el objeto de datos de flujo de trabajo especificado es un objeto de datos de flujo de trabajo de lista o no. Cuando se establece a `true`, el objeto de datos de flujo de trabajo especificado actuará como una lista y, por lo tanto, se podrá utilizar para hacer que las listas de datos estén disponibles en todo el flujo de trabajo.

initialize-attributes
Contiene un valor BOOLEANO que indica si los atributos asociados al objeto de datos de flujo de trabajo deben inicializarse cuando el objeto de datos de flujo de trabajo se utiliza por primera vez. Los valores predeterminados utilizados son los mismos que establecerían en una estructura Cúram.

wdo-name
Contiene el nombre del objeto de datos de flujo de trabajo.

display-name
Contiene el nombre de visualización del objeto de datos de flujo de trabajo. Este nombre representa una breve descripción del objeto de datos de flujo de trabajo y se visualiza en la herramienta de definición de procesos. Se trata de una cadena localizable que no contiene ningún parámetro. Para obtener más detalles sobre el texto localizado y los metadatos asociados, consulte 6.2.1, “Texto localizado”, en la página 32.

description
Contiene una descripción más detallada del objeto de datos de flujo de trabajo. También es una cadena localizable sin parámetros. Para obtener más detalles sobre el texto localizado y los metadatos asociados, consulte 6.2.1, “Texto localizado”, en la página 32.

attributes
Contiene los detalles de todos los atributos asociados con el objeto de datos de flujo de trabajo.

attribute
Contiene los detalles de uno de los atributos asociados al objeto de datos de flujo de trabajo. Los metadatos siguientes conforman un atributo de objeto de datos de flujo de trabajo:

attribute-name
Contiene el nombre del atributo del objeto de datos de flujo de trabajo.

display-name
Representa el nombre de visualización del objeto de datos de flujo de trabajo. Este nombre representa una descripción breve del atributo del objeto de datos de flujo de trabajo. Se trata de una cadena localizable que no contiene ningún parámetro. Para obtener más detalles del texto localizado y de los metadatos asociados, consulte 6.2.1, “Texto localizado”, en la página 32.

type Cada atributo de objeto de datos de flujo de trabajo definido debe especificar un tipo que debe ser un dominio base de Curam válido. Al crear un atributo de objeto de datos de flujo de trabajo en la herramienta de definición de procesos, este tipo se selecciona de la tabla de códigos `DomainType`. Esta tabla de códigos debe consultarse para obtener la lista completa de los tipos disponibles para los atributos de un objeto de datos de flujo de trabajo. El tipo de un atributo de objeto de datos de flujo de trabajo se utiliza para garantizar que las correlaciones de datos contenidas en un proceso de flujo de trabajo sean compatibles y no provoquen errores en tiempo de ejecución. Un ejemplo: si un

campo de parámetro de método de objeto de proceso de negocio es de tipo STRING (cadena), el atributo de objeto de datos de flujo de trabajo utilizado para correlacionar los datos en ese campo también deberá ser de tipo STRING.

required-at-enactment
Las correlaciones de promulgación representan la cantidad mínima de datos necesaria para promulgar el flujo de trabajo. Deben contener una entrada por cada atributo de objeto de datos de flujo de trabajo que tenga su distintivo de obligatoriedad para la promulgación establecido a `true`. Por el contrario, si se establece este distintivo a `false` (el valor predeterminado), este atributo de objeto de datos de flujo de trabajo no será obligatorio para la promulgación del proceso asociado. La herramienta de definición de procesos se utiliza para crear estas correlaciones de promulgación, y lo hace examinando cada atributo de objeto de datos de flujo de trabajo que se haya definido, creando una correlación para aquellos que tengan el distintivo de promulgación obligatoria establecido a `true`. Cuando una definición de proceso de flujo de trabajo publicada se ha seleccionado como un proceso de subflujo en una actividad de subflujo (consulte Capítulo 11, “Subflujo”, en la página 87), todos los objetos de datos de flujo de trabajo que se hayan marcado como necesarios para la promulgación en el proceso de subflujo deberán estar correlacionados antes que pueda publicarse la definición de proceso padre.

process-output
Un proceso de flujo de trabajo se puede marcar como servicio web estableciendo un valor de metadatos que indique que el proceso debe exponerse como servicio web. Esto permite que el proceso pueda participar en un proceso orquestado BPEL (Business Process Execution Language) y significa que el proceso puede ser llamado desde un proceso BPEL de forma síncrona o asíncrona. También puede ser necesario sacar los datos correlacionándolos desde un proceso de flujo de trabajo de vuelta al proceso BPEL que lo ha llamado. Cuando se establece a `true`, este elemento opcional indica que los datos de este atributo de objeto de datos de flujo de trabajo deben pasarse de vuelta al proceso BPEL invocante cuando finaliza el proceso de flujo de trabajo de Cúram. El valor predeterminado de este elemento es `false`.

constant-value
Este elemento opcional indica si el atributo de objeto de datos de flujo de trabajo representa un valor constante. En muchas partes de una definición de proceso de flujo de trabajo, los atributos de objeto de datos de flujo de trabajo se utilizan en correlaciones de entrada (es decir, correlaciones de funciones de asignación, correlaciones de funciones de fecha límite, etc.). En algunos de estos casos, es necesario poder utilizar constantes en algunas de estas correlaciones. Al proporcionarse un valor constante, los atributos de objeto de datos de flujo de trabajo de este tipo podrán utilizarse con este fin. Un atributo de objeto de datos de flujo de trabajo no puede tener su distintivo de obligatoriedad para la promulgación establecido a `true` y contener a su vez un valor constante. Los datos pasados como datos de promulgación se consideran dinámicos y sujetos a cambios. Los datos especificados en un atributo de objeto de datos de flujo de trabajo constante no son adecuados para este fin, porque su valor ya se conoce.

initial-value
Este elemento indica si el atributo de objeto de datos de flujo de trabajo tiene un valor inicial. Esta característica puede ser útil en situaciones en las que un atributo de objeto de datos de flujo de trabajo se utilice en el flujo de trabajo antes de haber sido informado por una actividad automática o de otra forma (es decir, para evitar tener que utilizar una actividad automática para informar los atributos de objeto de datos de flujo trabajo sólo para asegurarse de que dichos atributos no sean nulos cuando se utilicen después en las condiciones de transición del flujo de trabajo). Cuando se ha informado este elemento, el atributo de objeto de datos de flujo de trabajo se inicializa al valor especificado la primera vez que se utiliza. Las diversas correlaciones de salida que existen en un proceso de flujo de trabajo podrán sobrescribir después el valor inicial de un atributo de objeto de datos

de flujo de trabajo. Un atributo de objeto de datos de flujo de trabajo no puede tener un valor constante y un valor inicial especificado.

4.3 Validaciones

- Un proceso de flujo de trabajo deberá contener un único objeto de datos de flujo de trabajo `Context_RuntimeInformation`.
- Un nombre de objeto de datos de flujo de trabajo deberá ser exclusivo en el contexto de la definición de proceso de flujo de trabajo contenedora.
- El nombre de un objeto de datos de flujo de trabajo deberá ser un identificador Java™ válido.
- Un nombre de objeto de datos de flujo de trabajo definido por el usuario no podrá contener el prefijo `Context_`, porque es un prefijo reservado en el sistema de flujo de trabajo de Cúram.
- Cada objeto de datos de flujo de trabajo especificado en la definición de proceso de flujo de trabajo deberá contener al menos un atributo asociado.
- El nombre de atributo de objeto de datos de flujo de trabajo deberá ser un identificador Java válido.
- Un atributo de objeto de datos de flujo de trabajo no podrá crearse con el nombre `"value"`. Se trata de un nombre de atributo reservado en el sistema de flujo de trabajo de Cúram.
- El tipo de un atributo de objeto de datos de flujo de trabajo deberá ser un dominio base de Cúram válido y deberá estar contenido en la tabla de códigos `DomainType`.
- Un atributo de objeto de datos de flujo de trabajo no podrá marcarse a la vez como obligatorio para la promulgación y como valor constante.
- Un atributo de objeto de datos de flujo de trabajo no puede tener un valor constante y un valor inicial especificado.
- Si un atributo de objeto de datos de flujo de trabajo se ha marcado como constante, deberá proporcionarse un valor constante. Por el contrario, si el atributo no se ha marcado como constante, no deberá especificarse tal valor.
- Si el atributo de objeto de datos de flujo de trabajo se ha marcado como constante, solo podrá especificarse un valor en blanco para el atributo si el tipo del atributo es STRING.
- Si el atributo de objeto de datos de flujo de trabajo se ha especificado con un valor inicial, solo podrá especificarse un valor en blanco inicial para el atributo si el tipo del atributo es STRING.
- Si el atributo de objeto de datos de flujo de trabajo se ha marcado como constante, el valor especificado como dicha constante deberá ser compatible con el tipo del atributo asociado.
- Si el atributo de objeto de datos de flujo de trabajo se ha especificado con un valor inicial, el valor especificado como dicho valor inicial deberá ser compatible con el tipo del atributo asociado.
- El distintivo de salida de proceso sólo podrá establecerse a true para un atributo de objeto de datos de flujo de trabajo especificado si el proceso de flujo de trabajo asociado se ha expuesto como un servicio web.

4.4 Lista de WDO de contexto

Los objetos de datos de flujo de trabajo de contexto son aquellos que no están explícitamente definidos en los metadatos de la definición de proceso de flujo de trabajo, pero que tanto la herramienta de definición de procesos como el motor de flujo de trabajo hacen que estén disponibles en diversos lugares durante la ejecución de un proceso. Lo siguiente es una breve descripción de estos objetos de datos de flujo de trabajo de contexto, y se proporcionan enlaces en los que se puede encontrar más información sobre ellos.

Objeto de datos de flujo de trabajo Context_RuntimeInformation

El objeto de datos de flujo de trabajo Context_RuntimeInformation es un objeto de datos de flujo de trabajo puesto a disposición y mantenido por el motor de flujo de trabajo. Contiene información relevante a lo largo del ciclo de vida de una instancia de proceso de flujo de trabajo y los atributos disponibles reflejan esto. Estos atributos son los siguientes:

- `processInstanceID`: identificador generado por el sistema de la instancia de proceso (obtenido del servidor de claves de Cúram utilizando el conjunto de claves de flujo de trabajo).
- `enactingUser`: el nombre de usuario del usuario cuyas acciones en la aplicación han dado lugar a que se promulgue el proceso de flujo de trabajo.
- `enactmentTime`: la fecha y hora de promulgación del proceso.

Objeto de datos de flujo de trabajo Context_Result
Una transición procedente de una actividad automática debería poder utilizar el valor de retorno del método invocado directamente en su condición sin necesidad de correlaciones con atributos de objeto de datos de flujo de trabajo. Sin embargo, debido al modelo transaccional del motor de flujo de trabajo, estos datos se tienen que persistir fuera de la transacción de la invocación del método de objeto de proceso de negocio. Para lograr esto, se crea una definición de objeto de datos de flujo de trabajo en tiempo de ejecución si el valor de retorno se utiliza en condiciones de transición de salida. Nunca es necesario persistir estas definiciones de valor de retorno, porque se deducen en el motor de flujo de trabajo cuando hacen falta. Los datos reales de objeto de datos de flujo de trabajo se persisten hasta después de que se evalúen las transiciones procedentes de las instancias de actividad en cuestión, momento en que se suprimen. Para obtener más detalles sobre el objeto de datos de flujo de trabajo Context_Result, consulte 7.6, “Descripción de WDO de contexto”, en la página 46

Objeto de datos de flujo de trabajo Context_Event
El objeto de datos de flujo de trabajo Context_Event está disponible para su uso en condiciones de elemento de datos o de función (consulte Capítulo 16, “Condiciones”, en la página 113) para una transición procedente de una actividad que contenga una espera de suceso. Pone a disposición cierta información (p.ej. la clase y tipo del suceso generado, la hora de generación del suceso, etc.) contenida en el suceso generado para completar esa instancia de actividad. Esta información se puede utilizar para modelar la ruta de esa actividad especificada. Para obtener más detalles sobre el objeto de datos de flujo de trabajo Context_Event, consulte 8.5.4, “Descripción de WDO de contexto”, en la página 57.

Objeto de datos de flujo de trabajo Context_Decision
El objeto de datos de flujo de trabajo Context_Decision está disponible para su uso en una condición de elemento de datos o de función (consulte Capítulo 16, “Condiciones”, en la página 113) para una transición desde una actividad de decisión. Los atributos disponibles dependerán del formato de respuesta definido para la actividad de decisión. Para obtener más detalles sobre el objeto de datos de flujo de trabajo Context_Decision, consulte 10.4.4, “Descripción de WDO de contexto”, en la página 85

Objeto de datos de flujo de trabajo Context_Task
El objeto de datos de flujo de trabajo Context_Task está disponible para su uso en diversas correlaciones asociadas a una tarea de actividad manual (p. ej. correlaciones de entrada de función de asignación, correlaciones de entrada de función de fecha límite, parámetros de enlace de acción de actividad manual). Este objeto de datos de flujo de trabajo de contexto pone a disposición el identificador de la tarea creada como resultado de la ejecución de la actividad contenedora. Para obtener más detalles sobre el objeto de datos de flujo de trabajo Context_Task, consulte 9.3.5, “Descripción de WDO de contexto”, en la página 67.

Objeto de datos de flujo de trabajo Context_Loop
El objeto de datos de flujo de trabajo Context_Loop está disponible cuando se crea la condición de bucle asociada a una actividad de inicio de bucle. También está disponible para crear condiciones de transición de salida para cualquier actividad dentro de un bucle, y para cuando se especifican correlaciones de entrada, parámetros de texto y parámetros de enlace de acción para algunas actividades y funciones contenidas en un bucle. Este objeto de datos de flujo de trabajo de contexto pone a disposición de dichas correlaciones el número de veces que el un bucle se ha iterado. Para obtener más detalles sobre el objeto de datos de flujo de trabajo Context_Loop, consulte 12.5, “Descripción de WDO de contexto”, en la página 93.

Objeto de datos de flujo de trabajo Context_Deadline
El objeto de datos de flujo de trabajo Context_Deadline está disponible para su uso cuando se crea un elemento de datos o una condición de de función (consulte Capítulo 16, “Condiciones”, en la página 113) para una transición procedente de una actividad que tiene una espera de suceso con una fecha límite especificada. Está disponible para permitir a un desarrollador que modele diferentes rutas de ejecución desde una actividad que contiene una fecha límite en función de si esa fecha límite ha caducado. Para obtener más detalles sobre el objeto de datos de flujo de trabajo Context_Deadline, consulte 8.4.6, “Descripción de WDO de contexto”, en la página 55.

Objeto de datos de flujo de trabajo Context_Parallel
El objeto de datos de flujo de control Context_Parallel está disponible para su uso en las diversas correlaciones asociadas a una actividad manual paralela (por ejemplo, asunto de tarea y parámetros de texto de acción de tarea, correlaciones de estrategia de asignación, etc.) y una actividad de decisión paralela (p.ej. parámetros de texto de acción de decisión, parámetros de texto de acción secundaria, parámetros de texto de pregunta, etc.). Pone a disposición el índice del elemento del objeto de datos de flujo de trabajo de lista de actividad paralela que se utiliza para crear la instancia especificada de la actividad envuelta. Para obtener más detalles sobre el objeto de datos de flujo de trabajo Context_Parallel, consulte 13.3.6, “Descripción de WDO de contexto”, en la página 99.

Objeto de datos de flujo de trabajo Context_Error
El objeto de datos de flujo de trabajo Context_Error está disponible para su uso en una condición de elemento de datos o condición de función (consulte Capítulo 16, “Condiciones”, en la página 113) para una transición desde una actividad automática. Permite que un desarrollador de proceso modele una ruta de excepción fuera de una actividad automática, es decir, una transición que se sigue si la actividad automática falla debido a una excepción no manejada. Para obtener más detalles sobre el objeto de datos de flujo de trabajo Context_Error, consulte 7.6, “Descripción de WDO de contexto”, en la página 46

4.5 Información de entorno de ejecución

Las instancias de objetos de datos de flujo de trabajo y objetos de datos de flujo de trabajo de lista existen tan pronto como sea promulga un proceso de flujo de trabajo, y existen hasta que el proceso se completa. Estas instancias de objeto de datos de flujo de trabajo están por tanto disponibles para ser utilizadas en las actividades (p.ej., pasar datos a un método de BPO) y en las transiciones (p. ej., hacer que los datos estén disponibles en la evaluación de las condiciones de transición) durante todo el tiempo de vida de esa instancia de proceso.

El atributo `enactingUser` del objeto de datos de flujo de trabajo `Context_RuntimeInformation` se establece al nombre de usuario del usuario cuyas acciones en la aplicación dieron lugar a la promulgación del proceso de flujo de trabajo. Esto *no* da lugar a que se asigne el mismo valor a la transacción cuando se invoca posteriormente un método de BPO en la instancia del proceso de flujo de trabajo. Esto se debe a la demarcación de transacción en el motor de flujo de trabajo cuando las actividades automáticas (es decir, métodos BPO) se invocan en el servidor de aplicaciones. Debido a la naturaleza asíncrona de esta invocación y al requisito de garantizar que la llamada al código de aplicación esté en su propia transacción, el motor de flujo de trabajo invoca el método de (usuario SYSTEM) en lugar del usuario que promulgó el proceso de flujo de trabajo. En efecto, desde la óptica real de un negocio, puede que la persona que promulga el flujo de trabajo ni siquiera sepa que ha invocado ese método de BPO.

De forma similar, debe tenerse en cuenta que el usuario que promulga de una instancia de proceso de flujo no se pasa a ninguna de las instancias de proceso de subflujo que pueden invocarse desde el proceso padre. Si el usuario que promulga la instancia de proceso padre fuera necesario en alguna de las instancias de proceso de subflujo, deberá pasarse explícitamente utilizando un atributo de objeto de datos de flujo de trabajo en las correlaciones de entrada de dicho proceso de subflujo.

También debe tenerse cuidado al actualizar los datos de instancia de atributo de objeto de datos de flujo de trabajo cuando se ejecuten actividades automáticas paralelas en una instancia de proceso de flujo de

trabajo. Si tales actividades automáticas invocan el mismo método de BPO y y dicho método intenta actualizar los datos exactamente del mismo atributo de objeto de datos de flujo, podría producirse una situación de bloqueo de registro de base de datos. El diseñador del proceso de flujo de trabajo debería paliar tales situaciones diseñando la definición de proceso de flujo de trabajo de tal forma que garantice que las actividades automáticas que ejecutan en paralelo no actualicen el mismo atributo de objeto de datos de flujo.

Capítulo 5. Promulgación de procesos

5.1 Descripción general

Una definición de proceso define la estructura de un proceso de negocio y, para empezar a realizar el trabajo definido en esa definición de proceso, deberá crearse una instancia del proceso. El inicio de una instancia de proceso se conoce como *promulgación del proceso*. La mayoría de las definiciones de proceso requieren un conjunto mínimo de datos iniciales que se utiliza principalmente para identificar los objetos de negocio específicos sobre los que opera la instancia de proceso. Todos los mecanismos de promulgación deben tener una forma de aceptar los datos de entrada para iniciar un proceso determinado. Dichos datos de entrada se conocen como *datos de promulgación* de un proceso.

El flujo de trabajo de Cúram soporta actualmente cuatro mecanismos de promulgación:

- Promulgación desde código
- Promulgación desde un suceso
- Promulgación como un subflujo
- Promulgación vía web service

Los dos primeros mecanismos están descritos en este capítulo. El mecanismo de promulgación de subflujo se describe en Capítulo 11, “Subflujo”, en la página 87. El mecanismo de promulgación de servicio web se describe en Capítulo 19, “Servicios web de flujo de trabajo”, en la página 127.

5.2 Promulgación por código (API del servicio de promulgación)

El modo más directo de promulgar un proceso es identificar una ubicación de la aplicación desde la que se deba iniciar una instancia de proceso. Se debe insertar entonces el código en ese punto para llamar al API del servicio de promulgación. Esta API permite al desarrollador especificar el nombre del proceso que se va a iniciar y suministrar los datos de promulgación necesarios para el proceso.

Aunque promulgar un proceso de este modo es simple e intuitivo, tiene el inconveniente de estar codificado de forma rígida en la lógica de aplicación. Siendo esto así, alteraciones como eliminar la promulgación, cambiar el proceso que se va a iniciar o incluso cambios menores en los datos de promulgación necesarios requerirán cambios en el código y un nuevo despliegue de la aplicación.

5.2.1 Metadatos

```
<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
Contiene una lista de correlaciones que pueden utilizarse como datos iniciales en la promulgación la instancia de proceso asociada. No es obligatorio que una definición de proceso tenga definidas correlaciones de promulgación para que se promulgue.

mapping
El elemento `mapping` (correlación) representa un elemento de datos proporcionado desde un atributo de estructura de Cúram que se utiliza en la promulgación de la instancia de proceso asociada.

source-attribute
Esto representa un atributo de estructura Cúram que se utiliza para llenar los datos de promulgación del proceso y es obligatorio en la correlación de una promulgación.

struct-name
El nombre de una estructura Cúram que contiene un atributo obligatorio para promulgar el proceso de flujo de trabajo. Las estructuras agregadas y de lista también pueden utilizarse para pasar datos de promulgación a un proceso de flujo de trabajo como se ilustra en el fragmento de metadatos anterior.

name El nombre del atributo de una estructura Cúram necesario para promulgar el proceso de flujo de trabajo asociado. Allá donde se utilice un campo de una estructura de lista o agregada, este nombre representará el nombre completo de dicho campo. En tal caso, el nombre constará del nombre de rol procedente de la asociación entre las estructuras padre e hijo además del nombre real del campo. Esto se muestra en el fragmento de metadatos anterior.

target-attribute
Representa un atributo de objeto de datos de flujo de trabajo que hay que llenar con datos de promulgación del proceso y es obligatorio en la correlación de una promulgación.

wdo-name
El nombre de un objeto de datos de flujo de trabajo de Cúram que contiene el atributo de destino que debe correlacionarse. (Consulte Capítulo 4, “Objetos de datos de flujo de trabajo”, en la página 17).

name El nombre de un atributo de objeto de datos de flujo de trabajo de Cúram que está marcado como obligatorio en la promulgación. El valor del correspondiente atributo de origen de la estructura de Cúram se correlacionará con este atributo cuando se promulgue el proceso.

5.2.2 Validaciones

- El atributo de estructura de Cúram utilizado como parámetro de origen en un correlación de promulgación deberá ser válido y tener el tipo correcto para el atributo de objeto de datos de flujo de trabajo asociado de destino.
- El atributo de objeto de datos de flujo de trabajo de destino en una correlación de promulgación deberá ser válido y deberá estar marcado como obligatorio para la promulgación.
- Si el atributo de destino de la correlación de promulgación procede de de un objeto de datos de flujo de trabajo de lista, el atributo de origen deberá ser un campo de una estructura de lista.

5.2.3 Código

```
// Se crea la lista que se va a pasar al servicio de promulgación.
    final List enactmentStructs = new ArrayList();

    final TaskCreateDetails taskCreateDetails =
      new TaskCreateDetails();

    taskCreateDetails.subject = "El asunto de una tarea;
    taskCreateDetails.reservedBy = "unUsuario";

    enactmentStructs.add(taskCreateDetailsStruct);

    // Estructura agregada.
    GroupMemberDetails groupMemberDetails
      = new GroupMemberDetails();

    groupMemberDetails.dtls.memberName = "Usuario de prueba";

    enactmentStructs.add(groupMemberDetails);

    // Estructura de lista.
    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);
```

- El API de `EnactmentService` (servicio de promulgaciones) se proporciona para permitir la promulgación de procesos de flujo de trabajo mediante código de aplicación. La lista de estructuras Cúram proporcionada al método `startProcess()` debe ser suficiente para llenar completamente las

correlaciones de promulgación del proceso asociado. Tenga en cuenta que esta forma de promulgar un proceso es asíncrona y el proceso se lanzará en cuanto termine la transacción actual de la aplicación.

- El método `startProcessInV3CompatibilityMode` solo se proporciona para su uso en el API de tareas de aplicación central. El uso directo de este método en código personalizado no está soportado y puede obstaculizar futuras actualizaciones.

5.3 Promulgación de sucesos

Es posible iniciar un proceso en respuesta a un suceso generado. Esto requiere la configuración de algunos datos de configuración (ya sea mediante una interfaz de administración o como entradas de base de datos preconfiguradas). La configuración especifica el proceso o procesos que se van a iniciar en respuesta a un suceso específico que se genere. Las correlaciones de datos de suceso con los datos de promulgación necesarios para el proceso se pueden configurar también de este modo.

La configuración de sucesos de promulgación de procesos se almacena en la base de datos y se proporciona una interfaz de usuario para permitir la manipulación de estos datos. Como tal, la promulgación de proceso creada de este modo se puede habilitar, inhabilitar e incluso eliminar durante la ejecución. El inconveniente principal de este enfoque es que dado que los sucesos tienen una cantidad de información limitada, solo se podrán promulgar de este modo las definiciones de proceso que requieran una cantidad de datos de promulgación así de limitada.

En Cúram se proporciona un manejador de sucesos de promulgación de procesos y se registra de forma automática para que escuche sucesos generados en la aplicación. Cuando se haya configurado un proceso para promulgarse a partir de un suceso, se correlacionarán los datos del suceso en los datos de promulgación del proceso y se iniciará el proceso.

5.3.1 Datos de configuración

La habilitación de un suceso para promulgar un proceso requiere configurar una asociación suceso-proceso. Cada suceso generado en la aplicación comprueba si se ha asociado algún proceso que necesite promulgarse. Siempre se promulgará la última versión publicada del proceso de un evento asociado.

El registro de un suceso que desencadene un proceso se almacena como un registro en la tabla ProcEnactmentEvt. El manejador de sucesos de promulgación de procesos busca una representación en memoria caché de esta tabla para emparejar entradas cuando se genera un suceso en la aplicación y promulga los procesos que coincidan. En la tabla siguiente se describen los datos necesarios para llenar la tabla ProcEnactmentEvt.

Tabla 1. Descripción de la tabla ProcEnactmentEvt

Nombre del campo de la entidad	Descripción del campo
procStartEventID	El identificador exclusivo de la asociación suceso-proceso.
eventClass	La clase de suceso del suceso especificado para promulgar el proceso de flujo de trabajo.
eventType	La tipo de suceso del suceso especificado para promulgar el proceso de flujo de trabajo.
processToStart	Si se genera un suceso que contiene la clase y el tipo de suceso especificados, se promulgará la última versión publicada del proceso de flujo de trabajo especificado por este nombre.

Tabla 1. Descripción de la tabla ProcEnactmentEvt (continuación)

Nombre del campo de la entidad	Descripción del campo
enabled	Este distintivo booleano indica si la asociación suceso-proceso está habilitada. Esto permite habilitar/inhabilitar en tiempo de ejecución la promulgación de un proceso de flujo de trabajo por un suceso especificado.

La tabla ProcEnactEvtData almacena los datos que se deben correlacionar desde un suceso de negocio al flujo de trabajo que se promulga cuando se genera el suceso especificado. En la tabla siguiente se describen los datos necesarios para llenar la tabla ProcEnactEvtData.

Tabla 2. Descripción de la tabla ProcEnactEvtData

Nombre del campo de la entidad	Descripción del campo
procEventMappingID	El identificador exclusivo de la correlación de datos de un suceso de promulgación de procesos.
procStartEventID	El identificador exclusivo de la asociación suceso-proceso. Este campo es la clave exclusiva de la tabla ProcEnactmentEvt asociado y se utiliza para asociar todos los datos necesarios para promulgar el proceso de flujo de trabajo cuando se genera un suceso especificado.
eventField	Esto indica cuál de los tres campos de un suceso se va a utilizar para rellenar el atributo del objeto de datos de flujo de trabajo. Los valores de este campo se toman de la tabla de códigos `EventField` y se describen con más detalle a continuación.
wdoAttribute	Nombre completo de un atributo de objeto de datos de flujo de trabajo que se rellena con datos procedentes del campo de suceso dado cuando se promulga un proceso. Esta tabla incluirá una entrada para cada atributo de objeto de datos de flujo de trabajo que se haya marcado como obligatorio para la promulgación en el proceso que el suceso generado va a promulgar.

Existen tres campos de un suceso que pueden utilizarse como correlaciones de promulgación. Se enumeran en la tabla `EventField` y se describen a continuación.

datos de suceso primarios
Un identificador exclusivo relacionado con la clase del suceso a partir de la cual se genera el suceso. Por ejemplo, cuando el tipo de objeto de negocio especificado para un suceso es igual a 'Case' (caso), los datos de suceso podrían ser el identificador de caso.

datos de suceso secundarios
Esto puede ser cualquier valor numérico y está pensado para los sucesos que deben representar una asociación entre dos entidades.

generado por usuario
El nombre de usuario de Cúram del usuario que ha generado el suceso.

5.3.2 Validaciones

- Los datos disponibles procedentes de un suceso deberán ser suficientes para llenar completamente los datos de promulgación de la definición de proceso asociada.
- Allá donde un proceso ya se haya configurado para una promulgación basada en suceso, las modificaciones posteriores de los datos de promulgación del proceso deberán satisfacer las correlaciones de datos de suceso existentes.
- Allá donde se haya configurado un proceso para promulgarse a partir de un suceso, no se podrá suprimir su última versión publicada si no se pueden rellenar por completo a partir del suceso los datos de promulgación de la siguiente última versión publicada.

Capítulo 6. Actividad base

6.1 Descripción general

Todos los tipos de actividad soportados por el flujo de trabajo de Cúram tienen algunos detalles básicos en común. Esta información permite que el motor de flujo de trabajo los identifique de forma exclusiva y se muestren textual y gráficamente en la herramienta de definición de procesos. Cada actividad tiene un nombre y una descripción opcional localizables. Esto permite a las distintas interfaces de usuario de administración visualizar la información en el entorno local adecuado.

Este uniformidad de nivel básico permite que el motor de flujo de trabajo identifique y ejecute las actividades sin conocer el tipo específico de la actividad. Cada tipo de actividad conoce sus propios metadatos y cómo comportarse cuando se ejecuta. Este diseño permite la adición de nuevos tipos de actividad, si fuera necesario, sin que ello afecte al comportamiento esencial del motor de flujo de trabajo.

6.2 Metadatos

```
<automatic-activity id="1" category="AC1">
    <name>
      <localized-text>
        <locale language="es">AprobarCaso</locale>
      </localized-text>
    </name>
    <description>
      <localized-text>
        <locale language="es">Esta actividad automática
        se ejecutará para aprobar un caso.</locale>
      </localized-text>
    </description>

    ...

  </automatic-activity>
```

id Es un identificador de 64 bits proporcionado por el servidor de claves de Cúram cuando se crean las actividades en la herramienta de definición de procesos. El identificador de actividad debe ser exclusivo dentro de la definición de un proceso, pero no es necesaria la exclusividad global dentro de todas las definiciones de proceso del sistema.

category
Una actividad puede colocarse de forma opcional en una categoría. La categoría debe seleccionarse en la herramienta de definición de procesos y se obtiene de la tabla de códigos `ActivityCategory`. Este atributo se utiliza en la funcionalidad de búsqueda basada en actividades y no tiene ningún efecto funcional sobre la actividad.

name El nombre de la actividad es la forma que tiene la actividad de identificarse a efectos de visualización, mientras que el identificador de actividad se utiliza para identificar la actividad a efectos de ser ejecutada por el motor de flujo de trabajo.

description
Una actividad también puede tener una descripción opcional que especifica brevemente lo que hace la actividad para facilitar las cosas a quienes editen la definición del proceso en el futuro.

6.2.1 Texto localizado

Tal y como se ha mostrado en el fragmento de XML anterior, el nombre y la descripción de la actividad no son meros campos de texto, sino que se definen en términos de un elemento `localized-text`. Se trata de un elemento de propósito general utilizado en los metadatos de definición de procesos allí donde el texto deba traducirse.

Un elemento `localized-text` debe tener al menos un elemento hijo `locale`. Esto garantiza que haya siempre algún texto a mostrar en un campo determinado. En la herramienta de definición de procesos, cualquier texto localizable que se especifique en las pantallas de interfaz de usuario, aparte de la pantalla de localización, se guardará con el entorno local predeterminado del servidor tal especificado en la propiedad: curam.environment.default.locale.

```
<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 Contiene el texto del entorno local especificado por los atributos `language` y `country`. Nota: un entorno local se identifica de forma exclusiva mediante el idioma y el país, lo que significa que *en*, *en_US* y *en_GB* representan entornos locales diferentes.

language
Es obligatorio y representa el código de idioma ISO de dos letras.

country
Es opcional y representa el código de país ISO de dos letras.

6.3 Validaciones

- El nombre de la actividad es obligatorio y debe ser exclusivo dentro de una determinada definición de proceso de flujo de trabajo. Sin embargo, el nombre de la actividad es también una cadena localizable. Esta validación también garantiza que un nombre de la actividad especificado también sea exclusivo para cada entorno local especificado.
- Una actividad debe ser uno de los tipos de actividad permitidos. En la práctica esta regla se cumple por sí misma, ya que no hay forma de crear actividades sin seleccionar un tipo adecuado en la herramienta de definición de procesos. Incluso cuando se construyen definiciones de proceso manualmente en un editor de texto, los nombres de tipo de actividad se corresponden con los nombres de elemento de metadatos, con lo que se imposibilita la creación de un código válido que represente un tipo de actividad inexistente.

6.4 Tipos de actividad básicos

Algunos tipos de actividad (route, start-process y end-process) no tienen metadatos adicionales aparte de los comunes a todos los tipos de actividad. Asimismo, su comportamiento es lo suficientemente intuitivo como para describirlo aquí. Todos los demás tipos de actividad tienen capítulos dedicados.

6.4.1 Actividad de ruta

Una actividad de ruta es aquella que no realiza ninguna funcionalidad de negocio. Puede considerarse una actividad nula y su ejecución no afecta en modo alguno a los datos de aplicación ni al proceso de negocio.

El propósito principal de una actividad de ruta es ayudar a controlar de flujo. Las actividades de ruta se suelen utilizar como puntos de ramificación (división) y de sincronización (unión). También son útiles cuando las actividades que necesita un proceso de negocio no forman de manera natural una estructura de bloque válida que pueda ejecutar el motor de flujo de trabajo.

Dado que todos los tipos de actividad pueden tener notificaciones asociadas (consulte Capítulo 14, “Notificaciones de actividad”, en la página 101), las actividades de ruta pueden utilizarse para surtir el efecto de una mera notificación que no esté conectada a ninguna otra funcionalidad.

6.4.2 Actividades de inicio y finalización de proceso

Las actividades de inicio y finalización de proceso (start-process-activity y end-processs-activity respectivamente) proporcionan marcadores para el inicio y el fin de un proceso. Son puntos de anclaje a los que pueden adjuntarse otras actividades empleando transiciones y creando así una serie de pasos desde el inicio al final del proceso. En una definición de proceso válida, cuando se atraviesan todas las transiciones entre actividades partiendo la actividad de inicio de proceso, se debe llegar a la actividad de fin de proceso (tenga en cuenta que, en una instancia de proceso en ejecución, no todas las rutas se atraviesan necesariamente; por ejemplo, cuando se encuentra una división (consulte Capítulo 17, “División/Unión”, en la página 119) solo se seguirán algunas de las rutas dependiendo de la evaluación de las condiciones de transición). Como tal, la definición de proceso más simple (y, por cierto, la más inútil) es la que solo contiene estas dos actividades y una transición desde la actividad de inicio del proceso a la actividad de fin de proceso.

Cada definición de proceso debe tener exactamente una actividad de inicio de proceso y una actividad de fin de proceso. Cuando se define un proceso utilizando la herramienta de definición de procesos, estas dos actividades se crean automáticamente al crearse el proceso y no es necesario que el usuario las cree de forma explícita (de hecho, no puede hacerlo).

Las actividades de inicio y fin de proceso forman el bloque más externo de una definición de proceso correctamente estructurada en bloque tal y como exige el flujo de trabajo de Cúram.

Capítulo 7. Automática

7.1 Requisitos previos

- Los detalles básicos comunes a todos los tipos de actividades soportadas en un flujo de trabajo Cúram se describen en Capítulo 6, "Actividad base", en la página 31 y son aplicables a la actividad automática descrita aquí.

7.2 Descripción general

Una actividad automática es un paso de un proceso de flujo de trabajo que está completamente automatizado y en el que, bajo circunstancias normales, no es necesaria la intervención humana para la terminación de tal paso. Un paso de actividad automática invoca un método de la aplicación para realizar algún proceso necesario como parte del proceso de negocio global. Entre los usos habituales de las actividades automáticas se incluyen: realización de cálculos, actualización de entidades en la aplicación e inserción de datos en el motor de flujo de trabajo.

7.3 Métodos de negocio de Cúram

Gran parte del proceso de una actividad automática se lleva a cabo en el código de aplicación que se invoca. Las actividades automáticas hacen su trabajo invocando métodos de negocio de Cúram (tanto los BPO (objeto de proceso de negocio) como los métodos de entidad están soportados). Técnicamente son métodos públicos en entidades y objetos de proceso de negocio de Cúram. Una parte crítica de la definición de actividad automática es el método a invocar y los parámetros que hay que pasarle. Las secciones siguientes los describen.

7.3.1 Metadatos

```
<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

Contiene los detalles del método de negocio Cúram que se va a invocar cuando se ejecute la actividad automática asociada. Entre dichos detalles se incluyen el nombre de la interfaz y el método asociado, así como cualquier correlación de entrada y retorno asociada al método que se invoca. Las correlaciones de entrada y salida se describen en las secciones siguientes. Los atributos obligatorios de una correlación de un objeto de proceso de negocio (BPO) se describen a continuación.

interface-name

Representa el nombre completo de la interfaz Cúram que contiene el método asociado a la actividad automática.

method-name
Representa el método de la interfaz Cúram especificada que se va a invocar cuando se ejecute la actividad automática.

7.3.2 Validaciones

- Tanto la interfaz como los nombres de método deben especificarse para la correlación de métodos de objeto de proceso de negocio de la actividad automática.
- El nombre de interfaz especificado debe ser una clase válida, y dicha clase debe existir en la vía de acceso de clases de la aplicación Cúram.
- El nombre de método debe ser un nombre de método válido y debe existir en la interfaz especificada.

7.3.3 Código

Como se ha indicado anteriormente, cualquier método público válido de negocio de Cúram (BPO o entidad) puede estar asociado a una actividad automática en un proceso de flujo de trabajo y, por lo tanto, invocarse cuando se ejecuta dicha actividad. En general, un error en tales métodos al ejecutar una actividad automática provocará que se ejecute la estrategia de manejo de errores de flujo de trabajo. Esto puede provocar, por ejemplo, que la actividad asociada al método fallido se reintente una serie de veces. Por este motivo, los métodos asociados a las actividades automáticas no deberán lanzar excepciones por lo general. Si se utiliza la característica de modelado excepciones, cuando un método BPO lanza una excepción y se ha reintentado el número necesario de veces, se evaluarán todas las transiciones que parten de la actividad automática que contiene el objeto de datos de flujo de trabajo `Context_Error`. Si alguna de dichas transiciones evalúa a true, se sigue su ruta y, de esta forma, puede tener lugar el proceso de recuperación después de que el método BPO de actividad automática haya fallado.

7.4 Correlaciones de entrada

Tiene que haber un modo de proporcionar los parámetros necesarios a un método para poder invocarlo en el motor de flujo de trabajo. El motor de flujo de trabajo tiene una agrupación de datos a su disposición en forma de objetos de datos de flujo de trabajo (consulte Capítulo 4, “Objetos de datos de flujo de trabajo”, en la página 17). Las correlaciones de entrada se utilizan para declarar qué atributos de objeto de datos de flujo de trabajo se van a utilizar para informar los valores de los parámetros específicos del método cuando se invoca dicho método. Las correlaciones de entrada son opcionales cuando se especifican campos de estructura como parámetros del método. Sin embargo, los parámetros de tipo básico deben correlacionarse.

7.4.1 Metadatos

Los metadatos siguientes son comunes a los tres tipos de correlaciones de entrada de parámetros (tipo básico, estructura y estructuras agregadas) y, por lo tanto, no se describirán de nuevo.

formal-parameters
Contiene la lista de parámetros formales tal y como están definidos en la signatura del método de negocio de la actividad automática.

formal-parameter
Contiene los detalles de una correlación de entrada de parámetro formal tal y como se define en la signatura del método de negocio asociado. En esta instancia, existirá una entrada de correlación de parámetros formal por cada parámetro definido en el método de negocio asociado.

index Representa la posición del parámetro formal en la lista de parámetros formales definidos para el método especificado. Se trata de un índice basado en cero.

7.4.1.1 Correlaciones de entrada para parámetros de tipo básico

Los parámetros de tipo básico proporcionan el tipo más simple de correlación de entrada. En esta instancia, las correlaciones de entrada se crean para cada parámetro formal de tipo básico contenido en el método de negocio asociado a la actividad automática. Un parámetro de tipo básico en un método de

negocio Cúram representa una definición de dominio (consulte la *Guía de referencia de modelado de Cúram* para obtener detalles sobre las definiciones de dominio).

```
<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

Contiene los detalles de una correlación de entrada de tipo básico. Una correlación de entrada de tipo básico indica que el campo con el que se correlaciona es primitivo (a diferencia de las correlaciones de estructura y de estructura anidada descritas más abajo). Una correlación de entrada de tipo básico contiene el siguiente atributo obligatorio:

type Describe el tipo del campo primitivo con el que se va a correlacionar. En el caso de una correlación de entrada de tipo básico, esto es el tipo de la definición de dominio especificada como parámetro formal en el método.

wdo-attribute

Contiene los detalles del atributo de objeto de datos de flujo de trabajo (consulte Capítulo 4, "Objetos de datos de flujo de trabajo", en la página 17) que contiene los datos que se van a utilizar para informar el parámetro de tipo básico asociado cuando se invoca el método de negocio de la actividad automática. Los atributos obligatorios se describen a continuación:

wdo-name

Describe el nombre del objeto de datos de flujo de trabajo utilizado en la correlación de entrada.

name Describe el nombre del atributo en el objeto de datos de flujo de trabajo especificado que se utiliza en la correlación de entrada.

7.4.1.2 Correlaciones de entrada para parámetros de estructura

Las estructuras pueden especifcarse como parámetros de los métodos de un objeto de proceso de negocio. En esta sección se describen los metadatos de las correlaciones de entrada asociadas a dichos parámetros.

```
<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 Contiene los detalles de una correlación de entrada de estructura, incluido el tipo de la estructura y las correlaciones de cada campo definido en dicha estructura. Una correlación de entrada de estructura contiene el siguiente atributo obligatorio:

type Describe el tipo de la estructura especificado como parámetro formal en el método. Se representa como el nombre completo de la estructura especificada como parámetro formal.

field Contiene los detalles de la correlación de entrada para uno de los campos definidos en el parámetro de estructura. Un campo contiene los detalles de la correlación de entrada para el tipo primitivo básico asociado a dicho campo, así como el atributo obligatorio siguiente:

name Describe el nombre del campo tal y como se ha definido en la estructura especificada como parámetro formal.

base-type
Contiene los detalles de una correlación de entrada de tipo básico para el campo especificado. Una correlación de entrada de tipo básico contiene el siguiente atributo obligatorio:

type Describe el tipo del campo primitivo con el que se va a correlacionar.

wdo-attribute
Contiene los detalles del atributo de objeto de datos de flujo de trabajo (consulte Capítulo 4, “Objetos de datos de flujo de trabajo”, en la página 17) que contiene los datos que se van a utilizar para informar el campo de tipo básico asociado cuando se invoca el método. Este elemento no estará presente si el usuario no ha especificado una correlación de entrada para este parámetro de método. Este elemento, cuando se especifica, contiene los siguientes atributos obligatorios:

wdo-name
Describe el nombre del objeto de datos de flujo de trabajo utilizado en la correlación de entrada.

name Describe el nombre del atributo en el objeto de datos de flujo de trabajo especificado que se utiliza en la correlación de entrada.

7.4.1.3 Correlaciones de entrada para parámetros de estructura agregada

Pueden especificarse estructuras agregadas (consulte la *Guía de referencia de modelado de Cúram* para obtener detalles sobre la agregación de estructuras) como parámetros de métodos de negocio. En esta instancia, los metadatos son similares a los descritos anteriormente para los parámetros formales de estructura (consulte 7.4.1.2, "Correlaciones de entrada para parámetros de estructura", en la página 37). Sin embargo, hay una diferencia sutil, y es que un campo del parámetro estructura definido se puede resolver a otra estructura y no a un tipo primitivo, tal y como se ve en el ejemplo de correlaciones de estructuras. En este escenario, el nombre de campo no es el nombre del campo que se está correlacionando asociado al parámetro de estructura, sino que es el nombre del rol contenido en la asociación establecida entre la estructura especificada y la estructura que agrega. El fragmento de metadatos siguiente proporciona un ejemplo de tales correlaciones de entrada. Los elementos de metadatos se han descrito anteriormente en la sección de correlaciones de entrada de estructuras.

```
<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 Correlaciones de entrada para parámetros de estructura de lista

Ahora también pueden especificarse correlaciones de entrada para parámetros de estructura de lista. En esta instancia, los metadatos son similares a los descritos anteriormente para los parámetros formales agregados (consulte 7.4.1.3, "Correlaciones de entrada para parámetros de estructura agregada"). El tipo

de la estructura especificada en los metadatos para un parámetro de estructura de lista es el nombre de la estructura de lista. El nombre del primer campo especifica el nombre del rol contenido en la asociación establecida entre la estructura de lista especificada y la estructura hija que agrega. Normalmente, este campo se resuelve a otra estructura (la estructura hija contenida en la estructura de lista). El objeto de datos de flujo de trabajo especificado en una correlación así es un objeto de datos de flujo de trabajo de lista. El fragmento de metadatos siguiente proporciona un ejemplo de tales correlaciones de entrada. Los elementos de metadatos se han descrito anteriormente en la sección de correlaciones de entrada de estructuras.

```
<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 Correlaciones de entrada y elementos indexados de objetos de datos de flujo de trabajo de lista

En las actividades contenidas en bucles, un elemento de un objeto de datos de flujo de trabajo de lista puede utilizarse en una correlación de entrada para informar un campo de parámetro formal. Cuando se utiliza este tipo de correlación de entrada, cada vez que se itera el bucle que contiene la actividad, el campo de parámetro formal se informará con el siguiente valor de ese objeto de datos de flujo de trabajo de lista. Esto se enfatiza aquí porque la sintaxis de metadatos de una correlación así es ligeramente distinta de la de otros tipos de correlación de entrada.El fragmento de metadatos proporciona un ejemplo de tales correlaciones de entrada. El nombre del objeto de datos de flujo de trabajo de lista se utiliza para informar el campo de parámetro formal se cualifica con la sintaxis `[Context_Loop.loopCount]`. El motor de flujo de trabajo utiliza esto en tiempo de ejecución para determinar qué ejecución del bucle se está ejecutando y, por tanto, de qué elemento del objeto de datos de flujo de trabajo de lista hay que recuperar los datos con los que informar el campo de parámetro formal.

```
<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 Validaciones

- Los atributos de objeto de datos de flujo de trabajo especificados en las correlaciones de entrada deben ser válidos. El criterio de definición de un atributo de objeto de datos de flujo de trabajo válido puede consultarse en 4.3, “Validaciones”, en la página 21
- El tipo de parámetro formal al que se correlaciona y el tipo del atributo de objeto de datos de flujo de trabajo que se utiliza en esa correlación de entrada deben ser compatibles. Por ejemplo, si la correlación de entrada que se crea es un campo de estructura de tipo STRING, el atributo de objeto de datos de flujo de trabajo que se utiliza para esa correlación también deberá ser de tipo STRING.
- El objeto de datos de flujo de trabajo `Context_Task` no podrá utilizarse en una correlación de entrada si la actividad asociada no es una actividad manual o de decisión.
- El objeto de datos de flujo de trabajo `Context_Loop` no podrá utilizarse en una correlación de entrada si la actividad asociada no está contenida en un bucle.
- Se mostrará un aviso de validación si ningún parámetros de estructura definidos en el método del objeto de proceso de negocio contiene una correlación de entrada asociada.
- Todos los parámetros formales de tipo básico primitivo definidos en el método del objeto de proceso de negocio que deben contener una correlación de entrada asociada.
- Si el campo de parámetro formal que se correlaciona es un parámetro de tipo básico, no podrá utilizarse un atributo de un objeto de datos de flujo de trabajo de lista.
- Si el campo de parámetro formal que se correlaciona procede de una estructura de lista, deberá correlacionarse con un atributo de un objeto de datos de flujo de trabajo.
- Si el elemento indexado de un objeto de datos de flujo de trabajo de lista (es decir, `ClaimantDependent[Context_Loop.loopCount]`) se está utilizando en una correlación de entrada, el objeto de datos de flujo de trabajo asociado deberá ser un objeto de datos de flujo de trabajo de lista y la actividad que contiene las correlaciones de entrada deberá estar contenida en un bucle.

7.4.3 Información de entorno de ejecución

Los valores de los atributos de objeto de datos de flujo de trabajo definidos en las correlaciones de parámetro de entrada se proporcionan como datos de entrada al método especificado antes de que se invoque cuando se ejecute la actividad automática asociada.

7.5 Correlaciones de salida

Los objetos de datos de flujo de trabajo (consulte Capítulo 4, “Objetos de datos de flujo de trabajo”, en la página 17) son el almacén de datos de los motores de flujo de trabajo. Algunos atributos de los objetos de datos de flujo de trabajo especificados se rellenan cuando el proceso se promulga. Es útil, sin embargo, actualizar o establecer los valores de los atributos de un objeto de datos de flujo de trabajo cuando el proceso de flujo de trabajo se ejecuta. Para soportar esto, algunos tipos de actividad pueden devolver correlaciones de datos al motor de flujo de trabajo. Esto es de especial utilidad en el caso de las actividades automáticas, porque cabe suponer que los métodos de negocio que invocan puedan acceder a datos almacenados en cualquier entidad de la aplicación y devolverlos para su uso en actividades posteriores del proceso de flujo de trabajo. Dichas correlaciones de retorno procedentes de un método de objeto de proceso de negocio asociado a una actividad automática son opcionales.

7.5.1 Metadatos

Igual que en las correlaciones de entrada (consulte 7.4, “Correlaciones de entrada”, en la página 36), en las correlaciones de salida se soportan tipos de retorno primitivos, tipos de retorno de estructura, tipos de retorno de estructura anidada (agregada) y tipos de retorno de estructura de lista. Si el tipo de retorno es un tipo primitivo, podrá especificarse una entrada de correlación de retorno. Si el tipo de retorno es una estructura, una estructura agregada o una estructura de lista, podrán crearse las correlaciones de retorno para uno o más de los campos de la estructura especificada. Los siguientes fragmentos de metadatos proporcionan ejemplos de tales correlaciones:

7.5.1.1 Tipo de retorno primitivo

```
<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 Tipo de retorno de estructura

```
<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 Tipo de retorno de estructura agregada

```
<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 Tipo de retorno de estructura de lista

```
<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 Contiene los detalles de las correlaciones de salida especificadas para el método de negocio asociado a la actividad automática. En el caso de un tipo de retorno primitivo, estará presente una entrada de los metadatos de tipo básico como se muestra en el ejemplo anterior (consulte 7.5.1.1, "Tipo de retorno primitivo", en la página 42). En el caso de los tipos estructura, estructura agregada y lista, se especifica la etiqueta de metadatos struct y contiene campos cuyos tipos básicos se correlacionan utilizando atributos del objeto de datos de flujo de trabajo.

struct Contiene los detalles de la correlación de salida de estructura. Una correlación de salida de estructura contiene el siguiente atributo obligatorio.

field Contiene los detalles de la correlación de salida para uno de los campos definidos en el tipo de retorno de estructura. Un campo contiene los detalles de la correlación de salida para el tipo primitivo básico asociado a dicho campo, así como el atributo obligatorio siguiente:

name Representa el nombre del campo tal y como se ha definido en la estructura especificada como tipo de retorno. En el caso de los tipos de retorno de estructura no agregada, esto no representa más que el nombre del campo en la estructura de retorno especificada que está correlacionando. En el caso de los tipos de retorno de estructura agregada y estructura de lista, el nombre del campo representa el nombre del rol contenido en la asociación establecida entre la estructura especificada y la estructura que agrega.

base-type
Contiene los detalles de una correlación de salida de tipo básico para el campo especificado o un tipo de retorno primitivo.

wdo-attribute
Contiene los detalles del atributo del objeto de datos de flujo de trabajo (consulte Capítulo 4, "Objetos de datos de flujo de trabajo", en la página 17) con el que se van a correlacionar y persistir los datos presentes en el campo de tipo de retorno asociado. Los atributos obligatorios se describen a continuación:

wdo-name
Representa el nombre del objeto de datos de flujo de trabajo utilizado en la correlación de salida.

name Representa el nombre del atributo del objeto de datos de flujo de trabajo utilizado en la correlación de salida.

7.5.2 Validaciones

- No se permite ninguna correlación de parámetros de salida duplicada. En otras palabras, un atributo de objeto de datos de flujo de trabajo solo puede especificarse una vez en cualquier lista de correlaciones de retorno de salida.
- Todos los atributos de objeto de datos de flujo de trabajo especificados en las correlaciones de salida deberán ser atributos de objeto de datos de flujo de trabajo válidos en el contexto de la definición de proceso de flujo de trabajo contenedora.
- El tipo de campo de retorno desde el que se correlaciona y el tipo del atributo de objeto de datos de flujo de trabajo que se está correlacionando deben ser compatibles.
- Las correlaciones de salida no pueden crearse para atributos de objeto de datos de flujo de trabajo que se hayan marcado como atributos de objeto de datos de flujo de trabajo constantes. Los atributos de objeto de datos de flujo de trabajo constantes representan datos que deben permanecer constantes durante el tiempo de vida de la instancia de proceso (consulte 4.2, "Metadatos", en la página 18). Si se permitiese utilizar estos atributos en las correlaciones de salida, estos datos se sobrescribirían con los especificados en las correlaciones de salida.
- Si la estructura de retorno es una estructura de retorno de lista, el objeto de datos de flujo de trabajo utilizado en la correlación de retorno deberá ser un objeto de datos de flujo de trabajo de lista.

7.5.3 Información de entorno de ejecución

Los valores de los campos de tipo de retorno definidos en las correlaciones de parámetros de salida se persisten utilizando los atributos de objeto de datos de flujo de trabajo especificados después de que haya ejecutado la actividad automática asociada.

7.6 Descripción de WDO de contexto

Hay dos objetos de datos de flujo de trabajo de contexto que están disponibles al crear condiciones de función y de elemento de datos para transiciones desde una actividad automática. Se describen a continuación.

Objeto de datos de flujo de trabajo Context_Result
El objeto de datos de flujo de trabajo Context_Result está disponible para su uso en una condición de elemento de datos o de función (consulte Capítulo 16, "Condiciones", en la página 113) para una transición desde una actividad automática. Esto permite utilizar el valor de retorno del método invocado en dichas condiciones. Los convenios de los atributos disponibles para el objeto de datos de flujo de trabajo Context_Result son los siguientes:

- Si el tipo de retorno es un tipo básico, el atributo disponible se denominará `value` (es decir, `Context_Result.value`).

- Si el valor de retorno es una estructura, los valores de atributo de Context_Result disponibles son todos los campos presentes en la clase de la estructura de retorno (es decir, `Context_Result.description`, etc.).
- Si el valor de retorno está anidado (estructura agregada), los valores de atributo de Context_Result disponibles serán los campos disponibles en la estructura contenedora (es decir, `Context_Result.description`, etc.) y también los nombres completos de los campos de las estructuras anidadas (es decir, `Context_Result.dtls:concernRoleID`, etc.). Independientemente de la profundidad de la anidación del valor de la estructura de retorno, sólo hay un objeto de datos de flujo de trabajo Context_Result disponible en el que los nombres de las estructuras anidadas forman parte de los nombres de atributo. El separador entre una estructura anidada y sus campos es un signo de dos puntos como se aprecia en el ejemplo anterior.
- Si el tipo de retorno es una estructura de lista, el objeto de datos de flujo de trabajo `Context_Result` no estará disponible.

Objeto de datos de flujo de trabajo Context_Error

A veces, un método BPO invocado por una actividad automática puede fallar (es decir, lanzar una excepción que provoque una retrotracción de la transacción de la actividad). Cuando esto ocurre, puede ser útil poder modelar las acciones de seguimiento que tengan lugar después de la anomalía. El objeto de datos de flujo de trabajo `Context_Error` permite este tipo de modelado de "ruta de error". Está disponible para su uso en condiciones de elemento de datos o de función (consulte Capítulo 16, “Condiciones”, en la página 113) para una transición desde una actividad automática. El objeto de datos de flujo de trabajo de `Context_Error` tiene un atributo `exceptionOccurred` que se describe a continuación:

- El atributo `exceptionOccurred` es un valor booleano que indica que el método BPO asociado a la actividad automática ha fallado. Su valor predeterminado es false y se establece a true cuando falla el método BPO asociado.

En tiempo de ejecución, si falla el método BPO llamado en una actividad automática (y se ha reintentado el número de veces indicado en los requisitos previos y sigue fallando), el motor de flujo de trabajo establecerá a true el atributo `exceptionOccurred` de `Context_Error`. Las transiciones que utilizan el objeto de datos de flujo de trabajo `Context_Error` se evalúan y se siguen si el resultado es true. Esto permite que una instancia de proceso de flujo de trabajo continúe a lo largo de la ruta de error definida aunque haya fallado la actividad automática.

Si el método BPO llamado falla y no hay *ninguna* transición que utilice el objeto de datos de flujo de trabajo `Context_Error`, la actividad se detendrá y se creará una entrada en la consola de administración de mensajes de error.

Nota: El objeto de datos de flujo de trabajo `Context_Error` no da cuenta del *motivo* del error, solo indica si se ha producido uno o no.

Capítulo 8. Espera de suceso

8.1 Requisitos previos

- Los detalles básicos comunes a todos los tipos de actividades soportadas en un flujo de trabajo Cúram se describen en Capítulo 6, "Actividad base", en la página 31 y son aplicables a la actividad de espera de suceso descrita aquí.

8.2 Descripción general

La aplicación de Cúram tiene la capacidad de generar sucesos en diversos puntos que informan a los escuchas registrados de lo que ha sucedido. Pueden registrarse varios escuchas de suceso distintos para escuchar un suceso determinado. Estos escuchas de sucesos son funciones de aplicación que implementan la interfaz `curam.util.events.impl.EventHandler`. Cuando se genera un suceso especificado, el motor de flujo de trabajo invoca la función de manejador de sucesos asociada (consulte la *Guía del desarrollador del servidor Cúram* para obtener más detalles sobre sucesos y manejadores de sucesos).

El flujo de trabajo utiliza este recurso de una forma ligeramente distinta a través de actividades de espera de suceso. Una actividad de espera de suceso detiene la ejecución de una rama concreta de una instancia de proceso hasta que produzca un determinado suceso.

8.3 Lista de sucesos

No es totalmente correcto afirmar que una actividad de espera de suceso detiene un proceso de flujo de trabajo hasta que se genera un determinado suceso. De hecho, una espera de suceso puede especificar cualquier número de sucesos a los que esperar. Si se ha especificado que no espere a que se generen todos esos sucesos para completar la actividad, el primer suceso que coincida con una de las esperas de suceso especificadas finalizará la actividad y hará avanzar el flujo de trabajo. En este escenario, que se genere o no el resto de los sucesos no afecta al proceso. También es posible especificar que los sucesos generados asociados emparejen todas las esperas de suceso antes de que la actividad complete y prosiga el proceso de flujo de trabajo.

8.3.1 Metadatos

```
<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
Contiene los detalles de la espera de suceso asociada a la actividad especificada. Esto incluye los detalles de todos los sucesos para la espera de suceso.

wait-on-all-events
El valor de este distintivo indica al motor de flujo de trabajo si debe esperar a que se generen los sucesos para todas las esperas de suceso especificadas antes de que se complete la actividad asociada. Si se establece a false, el primer suceso que coincida con una de las esperas de suceso especificadas dará lugar a la finalización de la actividad asociada y a que avance el flujo de trabajo. Cuando se establece a true, deberá generarse un suceso por cada una de las esperas de suceso especificadas para la actividad antes de que la actividad se complete y avance el flujo de trabajo.

events Contiene los detalles de todos los sucesos a los que espera la actividad especificada.

event Contiene los detalles de un suceso específico al que espera esta actividad. Los detalles del suceso contienen los siguientes atributos obligatorios:

event-class
Representa la clase de suceso de negocio al que espera este proceso.

event-type
Representa el tipo de suceso de negocio al que espera este proceso. La combinación de event-class y event-type denotará el suceso de negocio requerido.

identifier
Representa el identificador exclusivo de este suceso. El identificador solo debe ser exclusivo dentro de la lista de sucesos de esta actividad.

event-match-attribute
Esto representa el atributo de objeto de datos de flujo de trabajo (consulte Capítulo 4, “Objetos de

datos de flujo de trabajo", en la página 17) que se utiliza para emparejar con la instancia requerida del suceso específico. Por ejemplo, en el primer suceso especificado en los metadatos de arriba, el atributo de objeto de datos de flujo de trabajo haría referencia al identificador de tarea asociado al cierre de una tarea específica. Cuando se genera este suceso, el motor de flujo de trabajo utilizará los datos del atributo de coincidencia de suceso para identificar de forma exclusiva la tarea a cerrar.

multiple-occurring-event
Indica que este suceso representa un evento de ocurrencia múltiple. Esto significa que si se especifican estos metadatos para un suceso, el motor de flujo de trabajo creará un registro de suceso por cada elemento del objeto de datos de flujo de trabajo de lista especificado como suceso de ocurrencia múltiple cuando se ejecute dicha actividad. Esto permite que el motor de flujo de trabajo espere a varias apariciones del mismo suceso.

Debe tenerse en cuenta que cuando el suceso de ocurrencia múltiple se especifica para un suceso, deberá utilizarse un atributo del objeto de datos de flujo de trabajo de lista asociado como datos de coincidencia del suceso. Esto garantizará que cada suceso generado por el motor de flujo de trabajo para el suceso de ocurrencia múltiple será exclusivo.

list-wdo-name
Representa el nombre del objeto de datos de flujo de trabajo de lista que se utiliza como suceso de ocurrencia múltiple.

8.3.2 Validaciones

- Deberá definirse al menos un suceso para la información de espera de suceso asociada a una actividad de espera de suceso.
- La clase y tipo de suceso especificados para cada suceso de negocio deberán ser entradas válidas en las tablas de base de datos de sucesos relevantes.
- Un suceso y el atributo de coincidencia de suceso asociado solo podrá definirse una vez en una actividad de espera de suceso. Es decir, una misma combinación de clase de suceso, tipo de suceso y atributo de coincidencia de suceso sólo podrá utilizarse una vez como suceso específico al que se espera en una actividad de espera de suceso.
- El atributo de objeto de datos de flujo de trabajo correlacionado con el atributo de coincidencia de suceso para un suceso deberá ser válido, porque se utiliza como un identificador exclusivo en el mecanismo de emparejamiento de sucesos y deberá ser de tipo LONG para ajustarse a los identificadores de 64 bits utilizados en Cúram.
- El objeto de datos de flujo de trabajo Context_Task sólo podrá utilizarse como atributo de objeto de datos de flujo de trabajo de emparejamiento de sucesos si si la actividad es manual o paralela y el suceso no es de ocurrencia múltiple.
- Si un elemento indexado de un objeto de datos de flujo trabajo de lista (es decir, ParentList[Context_Loop.loopCount]) se utiliza como datos de emparejamiento de sucesos, el objeto de datos de flujo de trabajo deberá ser un objeto de datos de flujo de trabajo de lista y la actividad que contiene la correlación de sucesos deberá estar contenida en un bucle.
- Si un elemento indexado procedente del de objeto de datos de flujo de trabajo de lista Parallel se utiliza como datos de emparejamiento de sucesos, la actividad que contenga la correlación deberá ser una actividad paralela (es decir, ParallelListWDO[Context_Parallel.occurrenceCount]). El objeto de datos de flujo de trabajo indexado por el el objeto objeto de datos de flujo de trabajo Context_Parallel deberá ser el objeto de datos de flujo de trabajo de lista de la actividad paralela.
- Si el objeto de datos de flujo de trabajo de lista de ocurrencia múltiple no se ha especificado para el suceso y la actividad que contiene la correlación de sucesos no es paralela, no podrá utilizarse un atributo de un objeto de datos de flujo de trabajo de lista como datos de emparejamiento de sucesos para dicho suceso.
- Si el objeto de datos de flujo de trabajo de lista del suceso de ocurrencia múltiple se ha especificado para el suceso, deberá utilizarse un atributo de este objeto de datos de flujo de trabajo de lista como datos de emparejamiento de sucesos de dicho suceso.

- El atributo de objeto de datos de flujo de trabajo correlacionado como suceso de ocurrencia múltiple deberá ser válido. También deberá ser un objeto de datos de flujo de trabajo de lista.

8.3.3 Código

En Cúram se proporciona un manejador de sucesos de flujo de trabajo y se registra de forma automática para que escuche sucesos generados en la aplicación. Pueden registrarse varias esperas de suceso para una determinada instancia de actividad en un proceso de flujo de trabajo. Si el distintivo *waitOnAllEvents* se establece a false para los datos de espera de suceso especificados, sólo será necesario que empareje una de estas esperas de suceso para completar esa instancia de actividad. El manejador de sucesos del flujo de trabajo procesará dicho suceso completando la instancia de actividad especificada e haciendo avanzar el proceso iniciando el siguiente conjunto de actividades del proceso. Luego se eliminarán todos los demás registros de suceso que se registraron para la instancia de actividad completada. Si se han especificado correlaciones de salida (consulte 8.5, "Correlaciones de salida", en la página 56) para la espera de suceso, el motor de flujo de trabajo las persistirá y podrán utilizarse en actividades y transiciones posteriores del proceso.

Cuando *waitOnAllEvents* está establecido a true, todas las esperas de suceso especificadas para la instancia de actividad deberán ser emparejadas por sucesos generados para poder completar la actividad y proseguir con el flujo. Por cada suceso generado que empareje con una espera de suceso asociada para la instancia de actividad, el manejador de sucesos del flujo de trabajo procesará el suceso suprimiendo el registro de escucha de suceso asociado y persistiendo las correlaciones de salida (consulte 8.5, "Correlaciones de salida", en la página 56) que se hayan especificado para la espera de suceso. Este proceso continúa hasta que todas las esperas de suceso asociadas hayan sido emparejadas por sucesos generados. Solo en ese momento el manejador de sucesos del flujo de trabajo completará la actividad especificada y hará avanzar el proceso iniciando el siguiente conjunto de actividades del proceso.

8.3.4 Información de entorno de ejecución

Un suceso generado en la aplicación sólo podrá hacer que continúe una instancia de proceso si dicho suceso coincide con el que se está esperando y el atributo de coincidencia de sucesos especificado para la espera de suceso coincide con los datos de suceso primarios del suceso.

8.4 Fecha límite

Una espera de suceso detiene un proceso de flujo de trabajo en vez de generar un suceso. Sin embargo, en muchos casos no es deseable que un proceso espere indefinidamente. Es posible que que nunca se genere el suceso que el proceso está esperando. Por ejemplo, el suceso podría generarse por casualidad antes de que el proceso alcance la actividad de espera de suceso. Para mitigar este riesgo es posible especificar de forma opcional una fecha límite para la generación de un suceso transcurrida la cual se invoque un manejador de fecha límite.

8.4.1 Requisitos previos

- Los métodos de manejador de fecha límite especificados para una fecha límite de espera de suceso son métodos de objeto de proceso de negocio de Cúram. Las correlaciones de entrada para los parámetros formales de estos métodos y sus metadatos asociados se describen en Capítulo 7, "Automática", en la página 35. Por tanto, este capítulo debería consultarse para obtener una descripción de estas correlaciones.

8.4.2 Metadatos

```
<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
: Representa un distintivo booleano que indica si la actividad debe finalizar cuando caduque la duración de la fecha límite. El valor predeterminado para este distintivo es false.

duration
: Representa la cantidad de tiempo que puede transcurrir antes de que se invoque el método manejador de fechas límite. La duración puede representarse en cualquiera de los formatos siguientes, que se utilizará posteriormente para calcular el tiempo de fecha límite de la espera de sucesos:

seconds
El número de segundos que pueden transcurrir antes de que se invoque el manejador de fecha límite

mapped-duration
El atributo de un objeto de datos de flujo de trabajo que puede correlacionarse como representación del número de segundos que pueden transcurrir antes de que se invoque el manejador de fecha límite.

deadline-handler
Representa el método que debe invocarse una vez que haya transcurrido la duración de la fecha límite. Deben especificarse los siguientes metadatos para un manejador de fecha límite:

interface-name
Representa el nombre completo del nombre de clase de la interfaz del manejador de fecha límite.

method-name
Representa el método de la interfaz del manejador de fecha límite que debe invocarse cuando vence la fecha límite.

formal-parameters
Contiene una lista de los parámetros del método del manejador de fecha límite y de los atributos de objeto de datos de flujo de trabajo que están correlacionados con esos parámetros cuando se invoca el manejador de fecha límite. Puede obtener detalles sobre la correlación de parámetros de método consultando 7.4, “Correlaciones de entrada”, en la página 36.

deadline-output-mappings
Contiene los datos de salida de fecha límite que pueden correlacionarse de forma opcional con los atributos del objeto de datos de flujo de trabajo. Estos datos indican si ha caducado o no la duración de la fecha límite, y la fecha y hora de caducidad de la duración de la fecha límite.

8.4.3 Validaciones

- Si se especifica un manejador de fecha límite, deberá referenciar un método de negocio de Cúram válido que exista en la vía de acceso de clases de la aplicación.
- Los atributos de objeto de datos de flujo de trabajo especificados en las correlaciones de entrada deberán ser válidos. El criterio de definición de un atributo de objeto de datos de flujo de trabajo válido puede consultarse en 4.3, “Validaciones”, en la página 21
- El tipo de parámetro formal al que se correlaciona y el tipo del atributo de objeto de datos de flujo de trabajo que se utiliza en esa correlación de entrada deberán ser compatibles. Por ejemplo, si la correlación de entrada que se crea es un campo de estructura de tipo STRING, el atributo de objeto de datos de flujo de trabajo que se utiliza para esa correlación también deberá ser de tipo STRING.
- Si el elemento indexado de un objeto de datos de flujo de trabajo de lista (es decir, `ClaimantDependent[Context_Loop.loopCount]`) se está utilizando en una correlación de entrada, el objeto de datos de flujo de trabajo asociado deberá ser un objeto de datos de flujo de trabajo de lista y la actividad que contiene las correlaciones de entrada deberá estar contenida en un bucle.
- Si el objeto de datos de flujo de trabajo `Context_Parallel` se utiliza en una correlación de entrada, la actividad que contenga las correlaciones de entrada deberá ser una actividad `Paralela`.
- Si un elemento indexado procedente del de objeto de datos de flujo de trabajo de lista Parallel (paralela) se utiliza como correlación de entrada, la actividad que contenga la correlación deberá ser una actividad paralela (es decir, ParallelListWDO[Context_Parallel.occurrenceCount]). El objeto de datos de flujo de trabajo indexado por el el objeto objeto de datos de flujo de trabajo Context_Parallel deberá ser el objeto de datos de flujo de trabajo de lista de la actividad paralela.
- La duración de fecha límite podrá especificarse mediante una duración de fecha límite en segundos o una correlación de atributo de objeto de datos de flujo de trabajo, pero no ambos.

- Si se ha especificado la duración de fecha límite mediante un atributo de objeto de datos de flujo de trabajo, el atributo deberá ser válido y de tipo INTEGER.
- Si se ha especificado una fecha límite para una actividad, deberá especificarse una función de manejador de fecha límite y/o el indicador de actividad completada deberá establecerse a true. Si este no es el caso, el flujo de trabajo no hará nada cuando se alcance la fecha límite.
- Si el valor de duración caducada de las correlaciones de salida de la fecha límite se ha correlacionado con un atributo de objeto de datos de flujo de trabajo, el atributo deberá ser válido y de tipo BOOLEAN.
- Si el valor de hora de caducidad de la fecha límite las correlaciones de salida de la fecha límite se ha correlacionado con un atributo de objeto de datos de flujo de trabajo, el atributo deberá ser válido y de tipo DATETIME.
- El indicador de actividad completada no se podrá establecer a `true` si la actividad que contiene la fecha límite es paralela. Esto se debe al hecho de que las actividades paralelas no soportan fechas límite modeladas.

8.4.4 Código

- Los parámetros de retorno asociados al método manejador de fecha límite no se utilizan en el motor de flujo de trabajo y por lo tanto son irrelevantes.
- Se proporciona la función del API explorador de fechas límite de flujo de trabajo `DeadlineScanner.scanDeadlines()` para permitir la exploración de fechas límite de esperas de suceso que hayan excedido su duración especificada. Se procesarán dichas esperas de suceso y se invocará su función manejadora asociada o se completará la actividad asociada.

8.4.5 Información de entorno de ejecución

Cuando el motor de flujo de trabajo ejecuta una actividad que contiene metadatos de fecha límite, crea el tiempo de fecha límite como se indica a continuación:

- Si la duración se ha especificado en segundos, el cálculo es fecha y hora actual + segundos definidos en los metadatos = tiempo de fecha límite.
- Si la duración se ha especificado como un atributo de objeto de datos de flujo de trabajo, el cálculo es fecha y hora actual + valor definido en el atributo de objeto de datos de flujo de trabajo = tiempo de fecha límite.

Los plazos que han caducado se procesan invocando el trabajo por lotes `ScanTaskDeadlines`. Este trabajo por lotes invoca a su vez el API de exploración de fechas límite del flujo de trabajo descrita anteriormente, que recupera una lista de todas las fechas límite que han caducado y las procesa. Si se ha especificado un método manejador de fecha límite para la fecha límite, los valores de los atributos de objeto de datos de flujo de trabajo de definidos en las correlaciones de parámetros se proporcionan como parámetros de entrada al método de manejador de fecha límite y se invoca. Si el indicador de actividad completada se ha establecido a true, se completará la actividad asociada. Las correlaciones de salida de fecha límite (duración caducada y tiempo de caducidad de la fecha límite) que puedan haberse especificado se persisten aquí. Los atributos del objeto de datos de flujo de trabajo `Context_Deadline` también se persisten durante este proceso para que puedan utilizarse en las transiciones que surjan de la actividad que contiene la fecha límite.

8.4.6 Descripción de WDO de contexto

El objeto de datos de flujo de trabajo Context_Deadline está disponible para su uso en una condición de elemento de datos o de función (consulte Capítulo 16, “Condiciones”, en la página 113) para una transición procedente de una actividad con una espera de suceso que tiene una fecha límite. Los atributos disponibles del objeto de datos de flujo de trabajo de Context_Deadline son:

Context_Deadline.durationExpired
Representa un booleano que indica si ha caducado la duración de la fecha límite asociada a la actividad.

Context_Deadline.expiryTime
Un atributo que contiene la fecha y hora en que caduca la duración de la fecha límite.

8.5 Correlaciones de salida

El suceso generado contiene información que puede merecer la pena devolver en una correlación al motor de flujo de trabajo. El suceso tiene datos tanto primarios como secundarios. Los datos de suceso primarios son lo que se ha utilizado para emparejar el suceso, de modo que no tiene mucho sentido devolver esto en una correlación al proceso. Sin embargo, puede que el motor de flujo de trabajo desconozca los datos de suceso secundarios, de modo que pueden correlacionarse en él. Asimismo, dado que una actividad de espera de suceso puede esperar a cualquier número de sucesos, puede que el suceso generado sea de interés y, por tanto, también puede correlacionarse en el motor de flujo de trabajo. Por último, el usuario de Cúram que genera el suceso también puede ser de interés y, por tanto, también puede correlacionarse en el motor de flujo de trabajo.

Debe tenerse en cuenta que si una instancia de actividad debe esperar a que se emparejen todas sus esperas de suceso asociadas, las correlaciones de salida de suceso que existan para la instancia de actividad se procesarán cada vez que se genere un suceso que empareje con una de las esperas de suceso.

8.5.1 Metadatos

```
<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
Contiene los datos que se pueden correlacionar de forma opcional con el motor de flujo de trabajo desde el suceso que se ha generado.

event-type
Contiene el suceso de negocio generado al que estaba esperando la instancia de actividad.

output-data
Contiene los datos de suceso secundario que hay que correlacionar en el motor de flujo de trabajo.

raised-by
Contiene el nombre de usuario del usuario de Cúram que ha causado que se genere el suceso.

time-raised
Contiene la fecha y hora de generación del suceso.

8.5.2 Validaciones

- Si se especifica una correlación de salida de suceso de tipo suceso, deberá se un atributo de objeto de datos de flujo de trabajo válido y deberá ser de tipo STRING.

- Si se especifica la correlación de salida de suceso generado por nombre de usuario, deberá se un atributo de objeto de datos de flujo de trabajo válido y deberá ser de tipo STRING.
- Si se especifica una correlación de salida de suceso de datos de salida, deberá se un atributo de objeto de datos de flujo de trabajo válido y deberá ser de tipo LONG.
- Si se especifica una correlación de salida de hora de generación, deberá se un atributo de objeto de datos de flujo de trabajo válido y deberá ser de tipo DATETIME.

8.5.3 Información de entorno de ejecución

Cuando se genera un suceso en la aplicación al que está esperando una instancia de actividad, los atributos de objeto de datos de flujo de trabajo contenidos en las correlaciones de salida definidas para la espera de suceso se rellenan con los datos relevantes procedentes del suceso y se persisten.

8.5.4 Descripción de WDO de contexto

El objeto de datos de flujo de trabajo Context_Event está disponible para su uso en una condición de elemento de datos o de función (consulte Capítulo 16, “Condiciones”, en la página 113) para una transición procedente de una actividad con una espera de suceso. Los atributos disponibles del objeto de datos de flujo de trabajo de Context_Event son:

Context_Event.raisedByUserName
El nombre del usuario de Cúram que ha generado el suceso.

Context_Event.timeRaised
La hora en la que el suceso se ha generado.

Context_Event.fullyQualifiedEventType
El nombre completo (clase de suceso y tipo de suceso) del suceso de negocio generado.

Context_Event.outputData
Los datos de suceso secundario asociados al suceso generado.

8.6 Recordatorios

Un recordatorio se puede establecer en cualquier fecha límite asociada a actividad manual, de decisión, de espera de suceso, manual paralela o de decisión paralela. Se puede especificar un número indeterminado de recordatorios. Los recordatorios utilizan los metadatos de notificación descritos en el capítulo de notificación de actividades (consulte Capítulo 14, “Notificaciones de actividad”, en la página 101). Esto significa que para un recordatorio se podrán especificar el asunto, el cuerpo, la estrategia de asignación y las acciones habituales en una notificación.

8.6.1 Metadatos

```
<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
Es opcional y encapsula todas las etiquetas de recordatorio de la fecha límite.

reminder
Contiene todos los metadatos de recordatorio de la fecha límite incluyendo los metadatos de la notificación asociada.

delivery-offset
Hace referencia a un valor de la tabla de códigos `ReminderDeliveryOffset` que indica cuál será el desplazamiento respecto de seconds o de mapped-delivery-time. En el caso de una fecha límite, es el desplazamiento desde la hora de vencimiento de la fecha límite. Este es el único desplazamiento soportado en la actualidad.

delivery-time
Contiene las etiquetas seconds o mapped-delivery-time dependiendo de cuál se haya especificado.

seconds
Esta etiqueta representa los segundos previos a la hora de vencimiento de la fecha límite en que se enviará el recordatorio.

mapped-delivery-time
Esta etiqueta representa un objeto de datos de flujo de trabajo que contiene los segundos previos a la hora de vencimiento de la fecha límite en que se enviará el recordatorio.

8.6.2 Validaciones

- No se podrá crear un recordatorio si no se asocia una fecha límite a la actividad relevante. Además, si existe una fecha límite, pero no se ha establecido el manejador de fecha límite o el indicador de actividad completada se ha establecido a false, no se podrá crear un recordatorio.
- Cada recordatorio tiene un identificador. Dicho identificador deberá ser exclusivo en la fecha límite a la que está asociado.
- Deberá especificarse un mapped-delivery-time o un seconds para un recordatorio.
- Si se especifica un seconds, deberá ser anterior a la hora de caducidad de la fecha límite.
- El atributo de objeto de datos de flujo de trabajo referenciado por mapped-delivery-time deberá ser de tipo INTEGER.

- Todas las validaciones existentes para notificaciones de actividad (consulte Capítulo 14, “Notificaciones de actividad”, en la página 101) son aplicables a los metadatos de notificación asociados a los recordatorios.

8.6.3 Código

La función del API de exploración de fechas límite del flujo de trabajo `DeadlineScanner.scanDeadlines()` incluye una llamada a la función `deliverReminders()`, que procesa y entrega los recordatorios que hayan alcanzado su hora de entrega.

8.6.4 Información de entorno de ejecución

Cuando se ejecuta una actividad que contiene recordatorios, dichos recordatorios se persisten en la entidad `Recordatorios`. La hora en la que debe enviarse un recordatorio se calcula como sigue:

- Se recupera la duración de entrega del recordatorio en segundos. Puede especificarse directamente en segundos o en un atributo de objeto de datos de flujo de trabajo.
- Se recupera la duración en segundos de la fecha límite asociada al recordatorio. Puede especificarse directamente en segundos o en un atributo de objeto de datos de flujo de trabajo.
- Si la duración de entrega del recordatorio es un número positivo y este número es menor que la duración de la fecha límite (no se pueden especificar entregas de recordatorio con horas que sean mayores que la fecha límite por motivos obvios), la hora de entrega de la notificación de recordatorio se calcula como la duración de fecha límite menos la duración de la entrega de recordatorio. Esta duración en segundos se convierte a fecha y hora, y se suma a la fecha y hora en que se crea el recordatorio. Esto se almacena entonces en el registro de recordatorio como la fecha y hora en que debe enviarse la notificación de recordatorio.

Los recordatorios configurados para las fechas límite se procesan y envían invocando el trabajo por lotes `ScanTaskDeadlines`. Dicho trabajo por lotes invoca la función `DeadlineScanner.scanDeadlines()`, que busca los recordatorios pendientes y y envía las notificaciones de recordatorio asociadas (utilizando la estrategia de asignación de notificación de recordatorios para determinar los usuarios a quienes hay que enviar las notificaciones). Los recordatorios enviados se eliminan de la entidad `Reminders` para garantizar que no se vuelvan a enviar. Cuando la actividad finaliza, se eliminan los recordatorios configurados para esa actividad que no se han enviado.

Capítulo 9. Manual

9.1 Requisitos previos

- Los detalles básicos comunes a todos los tipos de actividades soportadas en un flujo de trabajo Cúram se describen en Capítulo 6, “Actividad base”, en la página 31 y son aplicables a la actividad manual descrita aquí.

9.2 Descripción general

En cualquier proceso de negocio automatizado existe una necesidad de interactuar con agentes humanos para que tomen ecisiones, proporcionen datos adicionales o realicen tareas en el mundo real como, por ejemplo, telefonear a un cliente. En el flujo de trabajo de Cúram, tales pasos de un proceso se modelan utilizando actividades manuales. Una actividad manual especifica dónde es necesaria una intervención humana en proceso de negocio. También especifica la información que el usuario recibe cuando se le notifica que debe realizar una tarea y también la selección de los agentes a los que se asignará el trabajo.

9.3 Detalles de tarea

Para notificar a un usuario que tiene que realizar un trabajo como parte de un proceso de negocio automatizado, se le asigna una tarea. Una tarea es un mensaje que aparece en la bandeja de entrada del usuario. Esta bandeja de entrada especifica el trabajo que se espera que realice el usuario. La tarea también puede tener una lista de acciones asociadas. Las acciones son enlaces a páginas de solicitud de Cúram donde puede realizarse el trabajo necesario para llevar a cabo la tarea.

9.3.1 Metadatos

```
<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="es">
                    Página de inicio del caso: %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="es">
                    Confirme los detalles de persona de la
                      persona: %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 Contiene todos los detalles de una tarea, incluido el mensaje y los detalles de las acciones asociadas. Los diversos metadatos asociados a una tarea se describen a continuación

message
Contiene los detalles del mensaje parametrizado. Cuando se ejecuta una actividad manual, se crea una tarea. Cuando un usuario visualiza sus tareas en la bandeja de entrada, este mensaje representa el asunto de esa tarea.

message-text
Contiene los detalles del texto del mensaje. El texto del asunto puede contener cadenas sustituibles (%k), que se sustituirán con los parámetros de texto asociados. Un parámetro de texto es una correlación con un atributo de objeto de datos de flujo de trabajo. El parámetro k de la lista sustituirá a %k en la cadena de texto, donde k es el orden del parámetro en la lista. %k puede repetirse dentro de la cadena, de modo que cada atributo de objeto de datos de flujo de trabajo solo debe correlacionarse una vez. Puede especificarse de forma opcional un formato para las cadenas sustituibles colocando otra letra detrás de la cadena sustituible como, por ejemplo, %1d, donde d dará al valor un formato de fecha.

Tabla 3. Conversión de datos del texto del asunto

Letra de formateo	Formatear como
s	cadena
n	numérico
d	fecha
z	fecha/hora
t	hora

localized-text
Contiene detalles del texto del mensaje de tarea localizable o traducible. Para obtener más detalles del texto localizado y de los metadatos asociados, consulte 6.2.1, “Texto localizado”, en la página 32.

message-parameters
Un mensaje de tarea puede tener parámetros asociados a él. Este elemento contiene los detalles de los parámetros de atributo de objeto de datos de flujo de trabajo utilizados para sustituir los marcadores de posición en el texto asociado. Puede obtener detalles sobre los objetos de datos de flujo de trabajo y los atributos de objeto de datos de flujo de trabajo consulando Capítulo 4, “Objetos de datos de flujo de trabajo”, en la página 17.

actions
Contiene los detalles de todas las acciones asociadas a la tarea de actividad manual. Dichas acciones son enlaces a páginas de solicitud de Cúram donde puede realizarse el trabajo necesario para llevar a cabo la tarea.

action Contiene la definición de un hiperenlace a una página de Cúram en la que puede realizarse una tarea. A continuación se describen los campos asociados a la acción de tarea:

page-id
Representa el identificador de la página de Cúram de destino en la que un usuario puede empezar a realizar la acción necesaria.

principal-action
Las acciones pueden definirse como principales o como secundarias. Las acciones principales suelen contener los enlaces a las páginas de Cúram en las que un usuario puede empezar a realizar el trabajo necesario real. Las acciones secundarias suelen contener enlaces a información de apoyo a las que puede referirse el usuario asignado para hacer el trabajo mientras lleva a cabo la tarea asignada.

open-modal

Puede especificarse que las páginas enlazadas desde una acción de tarea se abran en un diálogo modal. Si este indicador se establece a true, la página especificada por el enlace de acción se abrirá en un diálogo modal. Si se establece a false (el valor predeterminado), la infraestructura cliente decidirá cómo abrir el enlace de la misma forma que lo hace con cualquier otro enlace de la aplicación (es decir, si la página forma parte de una configuración de pestañas, abrirá la correspondiente pestaña; si no, se limitará a sustituir la página inicial de enlace de acción en el área de contenido de la pestaña actual).

message

Contiene los detalles del mensaje parametrizado asociado a la acción que debe realizarse, incluido el texto del mensaje y los parámetros opcionales que puedan estar asociados al texto.

link-parameter

Los enlaces a las páginas de Cúram donde se va a realizar el trabajo real de la tarea deben contener un identificador de página (descrito anteriormente) y parámetros de página opcionales. Dichos parámetros se describen mediante estos metadatos y representan un par nombre/valor donde el atributo nombre es el nombre de un parámetro de enlace (el nombre del parámetro de página en la página cliente de Cúram asociada) y el valor lo proporciona un atributo de objeto de datos de flujo de trabajo. A continuación se describe el campo asociado al parámetro de enlace (link-parameter):

name El nombre del parámetro de enlace.

multiple-occurring-action

Significa que esta acción ocurre varias veces. Esto significa que si se especifican estos metadatos para una acción, el motor de flujo de trabajo creará un registro de acción por cada elemento del objeto de datos de flujo de trabajo de lista especificado como acción con varias ocurrencias cuando se ejecute dicha actividad.

Debe tenerse en cuenta que cuando la acción de ocurrencia múltiple se especifica para una acción, deberá utilizarse un atributo del objeto de datos de flujo de trabajo de lista asociado como parámetro de enlace de la acción.

list-wdo-name

Nombre del objeto de datos de flujo de trabajo de lista para su uso con la acción de ocurrencia múltiple.

wdo-attribute

La correlación de atributos del objeto de datos de flujo de trabajo especificada en estos metadatos proporciona el valor utilizado en el parámetro de enlace de acción.

task-priority

Una tarea puede contener de forma opcional una prioridad, y estos metadatos contienen esos detalles. La prioridad de una tarea se representará en cualquiera de los formatos siguientes:

priority

En esta instancia, la prioridad se selecciona en la herramienta de definición de procesos y se toma de la tabla de códigos de `TaskPriority`.

mapped-priority

La prioridad de una tarea manual puede correlacionarse utilizando un atributo de objeto de datos de flujo de trabajo. El fragmento de metadatos siguiente proporciona un ejemplo de cómo se puede hacerse esto:

```
<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
Representa un distintivo booleano utilizado para indicar si la fecha límite (consulte 8.4, "Fecha límite", en la página 52) asociada a la tarea de actividad manual puede sustituirse. Si se establece el valor de este distintivo a `true` (el valor predeterminado es `false`), se indica que la hora límite puede modificarse después de que el motor de flujo de trabajo haya creado la tarea.

allow-task-forwarding
Esto es un distintivo booleano que se utiliza para indicar si la tarea generada como consecuencia de la ejecución de la actividad manual asociada se puede reenviar a otro usuario. Cuando se genera una tarea, se asigna a un agente para que lleve a cabo el trabajo. Si se establece este distintivo a `true` (el valor predeterminado es `true`), se permite a ese agente reenviar la tarea a otro agente para que lleve a cabo el trabajo especificado.

administration-sid
Este campo permite especificar un identificador de seguridad de administración para una tarea manual. Esto permite a un usuario en un grupo asociado al identificador de seguridad especificado modificar los detalles de la tarea, aunque la tarea pueda estar reservada por otro usuario en la solicitud.

initial-comment
Permite especificar una correlación de comentarios inicial especificada para la tarea manual. El valor del atributo de objeto de datos de flujo de trabajo utilizado en esta correlación se utiliza para colocar un registro en la tabla `TaskHistory` cuando se ejecuta la actividad manual asociada.

9.3.2 Validaciones

- Deberá definirse un asunto para una tarea de actividad manual. Dicho asunto es una cadena localizable en la definición de proceso, pero deberá existir una entrada para el entorno local predeterminado del servidor.
- Todos los objetos de datos de flujo de trabajo utilizados como parámetros de texto de asunto en el mensaje de asunto de la tarea de actividad manual deberán ser atributos de objeto de datos de flujo de trabajo válidos en el contexto de la definición de proceso de flujo de trabajo contenedora.
- Si un elemento indexado de un objeto de datos de flujo trabajo de lista (es decir, PersonDetailsList[Context_Loop.loopCount]) se utiliza como parámetro de texto de asunto, el objeto de datos de flujo de trabajo deberá ser un objeto de datos de flujo de trabajo de lista y la actividad que contiene la correlación deberá estar contenida en un bucle.
- Si el objeto de datos de flujo de trabajo `Context_Parallel` se utiliza como parámetro del texto de asunto, la actividad que contenga la correlación deberá ser una actividad manual `paralela`.

- Si un elemento indexado procedente del de objeto de datos de flujo de trabajo de lista Parallel se utiliza como parámetro del texto de asunto, la actividad que contenga la correlación deberá ser una actividad paralela (es decir, ParallelListWDO[Context_Parallel.occurrenceCount]). El objeto de datos de flujo de trabajo indexado por el el objeto objeto de datos de flujo de trabajo Context_Parallel deberá ser el objeto de datos de flujo de trabajo de lista de la actividad paralela.
- Si se han especificado acciones para la tarea de actividad manual, los atributos de objeto de datos de flujo de trabajo utilizados como correlaciones para los parámetros de texto de acción deberán ser válidos en el contexto de la definición de proceso de flujo de trabajo contenedora.
- Si se han especificado acciones para la tarea de actividad manual, los atributos de objeto de datos de flujo de trabajo utilizados en las correlaciones de parámetro de enlace de acción de una acción de actividad manual deberán ser válidos en el contexto de la definición de proceso de flujo de trabajo contenedora.
- Si un elemento indexado de un objeto de datos de flujo trabajo de lista (es decir, PersonDetailsList[Context_Loop.loopCount]) se utiliza en las correlaciones de parámetro de texto o enlace de acción, el objeto de datos de flujo de trabajo deberá ser un objeto de datos de flujo de trabajo de lista y la actividad que contiene la correlación deberá estar contenida en un bucle.
- Si el objeto de datos de flujo de trabajo `Context_Parallel` se utiliza en las correlaciones de parámetro de enlace o texto de acción, la actividad que contenga la correlación deberá ser una actividad manual `paralela`.
- Si un elemento indexado procedente del de objeto de datos de flujo de trabajo de lista Parallel se utiliza en las correlaciones de parámetro de enlace o texto de acción, la actividad que contenga la correlación deberá ser una actividad paralela (es decir, ParallelListWDO[Context_Parallel.occurrenceCount]). El objeto de datos de flujo de trabajo indexado por el el objeto objeto de datos de flujo de trabajo Context_Parallel deberá ser el objeto de datos de flujo de trabajo de lista de la actividad paralela.
- El número de marcadores de posición utilizados en el texto del asunto y en el texto de la acción de la tarea de actividad manual deberá ser igual al número de atributos de objeto de datos de flujo de trabajo correlacionados para todos los entornos locales definidos.
- La prioridad de una tarea manual puede especificarse mediante un valor de código de tabla de códigos o una correlación de atributo de objeto de datos de flujo de trabajo, pero no ambos.
- Si se ha especificado una prioridad correlacionada para la tarea de actividad manual, el atributo de objeto de datos de flujo de trabajo especificado para ella deberá ser válido en el contexto de la definición de proceso de flujo de trabajo contenedora. También deberá ser de tipo STRING.
- Si se ha especificado una correlación de comentarios inicial para la tarea de actividad manual, el atributo de objeto de datos de flujo de trabajo especificado para ella deberá ser válido en el contexto de la definición de proceso de flujo de trabajo contenedora. También deberá ser de tipo STRING.
- El objeto de datos de flujo de trabajo especificado para su uso en la acción de ocurrencia múltiple deberá ser un objeto de datos de flujo de trabajo válido en el contexto de la definición de proceso de flujo de trabajo contenedora. También deberá ser un objeto de datos de flujo de trabajo de lista.
- Al menos un atributo del objeto de datos de flujo de trabajo de lista de acción de ocurrencia múltiple deberá utilizarse en los parámetros de enlace especificados para una acción de ocurrencia múltiple.

9.3.3 Código

Páginas de acción y parámetros de una página de acción

Las acciones especificadas para la tarea de actividad manual son enlaces a páginas de solicitud de Cúram donde puede realizarse el trabajo necesario para llevar a cabo la tarea. Las páginas especificadas en las acciones de tarea deben ser páginas Cúram válidas y deben estar disponibles en la aplicación Cúram. Los parámetros de estas páginas deben coincidir con los parámetros especificados como parámetros de enlace de acción en las acciones de la tarea asociada.

API LocalizableStringResolver.TaskStringResolver

El asunto de tarea y los mensajes de acción de tarea asociada se muestran en la bandeja de entrada del usuario para informarle del trabajo necesario para completar la tarea asociada. El API

`LocalizableStringResolver.TaskStringResolver` contiene las funciones que resuelven los asuntos de tarea y los mensajes de acción a la configuración regional del usuario correcta. Como parte de dichas funciones, también se lleva a cabo la sustitución de los marcadores de posición con los valores de atributo de objeto de datos de flujo de trabajo asociados especificados en las correlaciones asociadas.

API de administración de tareas
En la clase `TaskAdmin` se proporcionan una serie de funciones que permiten la manipulación de tareas. Para obtener más detalles de las funciones disponibles, consulte la especificación Javadoc asociada a la clase `TaskAdmin`.

API de administración del historial de tareas
En la tabla `TaskHistory`, durante el tiempo de vida de una tarea, se escriben diversos sucesos de ciclo de vida de una tarea (p.ej. cuando se crea una tarea, cuando se asigna una tarea, cuando se cierra una tarea). En esta clase del API se proporcionan una serie de funciones de búsqueda que permiten examinar esas entradas. Para obtener más detalles de las funciones disponibles, consulte la especificación Javadoc asociada a la entidad `TaskHistoryAdmin`.

API de administración de la fecha límite de un flujo de trabajo
En la clase `WorkflowDeadlineAdmin` se proporcionan una serie de funciones que permiten la manipulación de fechas límite de flujo de trabajo. Para obtener más detalles de las funciones disponibles, consulte la especificación Javadoc asociada a la clase `WorkflowDeadlineAdmin`.

9.3.4 Información de entorno de ejecución

Cuando el motor de flujo de trabajo ejecuta una actividad manual, se crea una tarea y se asigna a un agente para que realice ese trabajo (consulte 9.4, “Estrategia de asignación”).

9.3.5 Descripción de WDO de contexto

El objeto de datos de flujo de trabajo Context_Task permite que el identificador exclusivo de la tarea creada como parte de la ejecución de la actividad manual asociada esté disponible para su uso en las diversas correlaciones de metadatos asociadas a la actividad manual. Como ejemplos de algunas de estas correlaciones cabe mencionar las correlaciones de datos de coincidencia de eventos (consulte 8.3, “Lista de sucesos”, en la página 49) y las correlaciones de entrada de la función de fecha límite (consulte 8.4, “Fecha límite”, en la página 52). El único atributo disponible en este objeto de datos de flujo de trabajo es:

Context_Task.taskID
El atributo `taskID` representa el identificador exclusivo de la tarea que se crea cuando se ejecuta la actividad manual asociada.

9.4 Estrategia de asignación

Una organización suele tener muchos agentes humanos con distintos niveles de responsabilidad que puedan realizar el trabajo de una determinada definición de proceso. Para seleccionar un determinado agente o grupo de agentes que puedan realizar el trabajo de una determinada actividad manual, se asigna una estrategia de asignación a la actividad. Los cuatro tipos de estrategias de estrategias de asignación actualmente soportados por el flujo de trabajo de Cúram son: función, reglas clásicas, reglas CER y destino. Cuando se selecciona una estrategia de asignación de tipo destino, el agente o grupo de agentes a los que se asigna el trabajo se nombran directamente. La selección de una estrategia de asignación de funciones da lugar a la invocación de la función de asignación especificada cuando el motor de flujo de trabajo ejecuta la actividad asociada. Por último, si se ha seleccionado una estrategia de asignación de reglas clásicas o CER (Cúram Express Rules), el conjunto de reglas especificado se ejecuta cuando se ejecuta la actividad asociada.

9.4.1 Requisitos previos

- Si la estrategia de asignación asociada a una actividad manual es de tipo `Function` (función), estas funciones de asignación son los métodos de negocio de Cúram con una signatura específica. Las correlaciones de entrada para los parámetros formales de estos métodos y sus metadatos asociados se describen en Capítulo 7, “Automática”, en la página 35. Por tanto, este capítulo debería consultarse para obtener una descripción de estas correlaciones.

9.4.2 Metadatos

Como se ha descrito anteriormente, existen cuatro tipos de estrategias de asignación. Los metadatos necesarios para cada uno de estos tipos se describen en las secciones siguientes.

allocation-strategy

Este documento contiene los detalles de la estrategia de asignación definida para la estrategia manual. A continuación se describen los campos asociados a una estrategia de asignación:

type Contiene el tipo de la estrategia de asignación. Los cuatro tipos de estrategias de asignación actualmente soportados por el flujo de trabajo de Cúram son función, reglas clásicas, reglas CER y destino.

identifier

Representa el identificador de la estrategia de asignación. En una estrategia de asignación de tipo función, este identificador representa el nombre completo de la función de asignación que se utiliza. Para una estrategia de asignaciónde tipo regla o regla de Curam Express, este identificador representa el identificador del conjunto de reglas que se está utilizando. Por último, cuando se selecciona una estrategia de asignación de tipo destino, este identificador representa el identificador del destino de asignación utilizado.

9.4.2.1 Estrategia de asignación de funciones

```
<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

Contiene los detalles de las correlaciones de entrada para los parámetros formales de la función de asignación especificada. La funciones de asignación son métodos de negocio de Cúram (similares a los que se han especificado para las actividades automáticas) que tienen una signatura de retorno característica (las funciones de asignación deben tener un tipo de retorno de `curam.util.workflow.struct.AllocationTargetList`). Por lo tanto, los metadatos utilizados en estas correlaciones son los mismos que los utilizados en las correlaciones de entrada de los métodos de objeto de proceso de negocio asociados a las actividades automáticas. El lector debe consultar la sección 7.4, "Correlaciones de entrada", en la página 36 del capítulo de actividades automáticas para obtener más detalles sobre estos metadatos y su significado.

9.4.2.2 Asignación de reglas CER

```
<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

Contiene los detalles de todas las correlaciones para el *conjunto de reglas* especificado en el identificador de asignación. No es necesario correlacionar todos los atributos de objeto de datos de regla especificados en el conjunto de reglas (pueden crearse correlaciones para un subconjunto de ellos).

rdo-mapping

Contiene los detalles de una correlación entre un atributo de objeto de datos de regla especificado en el conjunto de reglas de asignación y su atributo de objeto de datos de flujo de trabajo asociado. Los metadatos siguientes constituyen una correlación válida:

source-attribute

Contiene los detalles del atributo de origen en la correlación (es decir, el lugar desde donde se proporcionan los datos en tiempo de ejecución). Un atributo de origen consta de un nombre de objeto de datos de flujo de trabajo y su nombre de atributo asociado (consulte Capítulo 4, "Objetos de datos de flujo de trabajo", en la página 17).

target-attribute

Contiene los detalles del atributo de destino en la correlación (es decir, el lugar al que se correlacionan los datos en tiempo de ejecución). Un atributo de destino consta de un nombre de objeto de datos de reglas y su nombre de atributo asociado.

9.4.2.3 Asignación de reglas CER

```
<manual-activity id="1" category="AC1">
      ...
      <task>
        ...
      </task>
      <allocation-strategy type="curam express rule"
                           identifier="Sample allocation Rules">
        <cer-set-mappings primary-class="sampleAllocationClass">
          <cer-class-mapping>
            <source-attribute wdo-name="TaskCreateDetails"
                              name="caseID" />
            <target-attribute cer-class-name="SampleAllocationClass"
                              name="caseID" />
          </cer-class-mapping>
          <cer-class-mapping>
            <source-attribute wdo-name="TaskCreateDetails"
                              name="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

Contiene los detalles de todas las correlaciones para el *Conjunto de reglas CER* especificado en el identificador de asignación. El parámetro de clase primaria principal debería apuntar a una clase de regla que contenga el atributo *targets* para esta estrategia de asignación. Se recomienda crear las correlaciones para todos los atributos especificados en la clase de reglas seleccionada.

cer-class-mapping

Contiene los detalles de una correlación entre un atributo de clase de regla especificado en el conjunto de reglas CER y su atributo de objeto de datos de flujo de trabajo asociado. Los metadatos siguientes constituyen una correlación válida:

source-attribute

Contiene los detalles del atributo de origen en la correlación (es decir, el lugar desde donde se proporcionan los datos en tiempo de ejecución). Un atributo de origen consta de un nombre de objeto de datos de flujo de trabajo y su nombre de atributo asociado (consulte Capítulo 4, “Objetos de datos de flujo de trabajo”, en la página 17).

target-attribute

Contiene los detalles del atributo de destino en la correlación (es decir, el lugar al que se correlacionan los datos en tiempo de ejecución). Un atributo de destino consta de un nombre de clase CER y su nombre de atributo asociado.

9.4.2.4 Estrategia de asignaciones de destinos

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

No es necesario describir metadatos adicionales en una estrategia de asignación de tipo *destino*. Como se ha indicado anteriormente, el identificador en este caso es el identificador del destino de asignación que contiene el agente o grupo de agentes a los que se va a asignar la tarea.

9.4.3 Validaciones

- Deberá definirse una estrategia de asignación para una tarea manual.
- Si la estrategia de asignación es de tipo función, la función especificada deberá ser válida y deberá existir en la vía de acceso de clases de Cúram.
- Si la estrategia de asignación es de tipo función, el tipo de retorno de la función deberá ser `curam.util.workflow.struct.AllocationTargetList`.
- Si la estrategia de asignación es de tipo función, los parámetros de entrada de la función especificada que se correlacionen deberán ser atributos de objeto de datos de flujo de trabajo válidos y el tipo del atributo de objeto de datos de flujo de trabajo deberá coincidir con el tipo del campo del parámetro de entrada.
- Si la estrategia de asignación es de tipo función y se utiliza un elemento indexado de un objeto de datos de flujo de trabajo de lista en una correlación de entrada, el objeto de datos de flujo de trabajo deberá ser un objeto de datos de flujo de trabajo de lista y la actividad que contenga la correlación deberá estar contenida en un bucle.
- Si la estrategia de asignación es de tipo regla, el conjunto de reglas especificado deberá ser válido.
- Si la estrategia de asignación es de tipo regla, todos los atributos de origen especificados en las correlaciones deberán ser atributos de objeto de datos de flujo de trabajo válidos en el contexto de la definición de proceso de flujo de trabajo contenedora. Todos los atributos de destino deberán ser atributo de objeto de datos de reglas válidos en el contexto del conjunto de reglas especificado. El tipo del atributo de objeto de datos de flujo de trabajo especificado como atributo de origen deberá coincidir con el tipo del atributo de objeto de datos de reglas especificado como atributo de destino en la correlación.
- No se permiten las correlaciones de atributos de destino duplicadas. En otras palabras, un atributo de objeto de datos de reglas solo puede especificarse una vez en cualquier lista de correlaciones de conjunto de reglas.
- Si un elemento indexado de un objeto de datos de flujo trabajo de lista (es decir, PersonDetailsList[Context_Loop.loopCount]) se utiliza en las correlaciones de estrategia de asignación de función o regla, el objeto de datos de flujo de trabajo deberá ser un objeto de datos de flujo de trabajo de lista y la actividad que contiene la correlación deberá estar contenida en un bucle.
- Si el objeto de datos de flujo de trabajo `Context_Parallel` se utiliza en las correlaciones de estrategia de asignación de función o regla, la actividad que contenga la correlación deberá ser una actividad `paralela`.
- Si un elemento indexado procedente del de objeto de datos de flujo de trabajo de lista Parallel se utiliza en las correlaciones de estrategia de asignación de función o regla, la actividad que contenga la correlación deberá ser una actividad paralela (es decir,

ParallelListWDO[Context_Parallel.occurrenceCount]). El objeto de datos de flujo de trabajo indexado por el el objeto objeto de datos de flujo de trabajo Context_Parallel deberá ser el objeto de datos de flujo de trabajo de lista de la actividad paralela.

9.4.4 Código

Como se ha indicado anteriormente, un método de objeto de proceso negocio especificado como una función de asignación deberá devolver una estructura de tipo `curam.util.workflow.struct.AllocationTargetList`.

Como ocurre con los métodos de negocio asociados a actividades automáticas, un error en la función de asignación cuando se ejecuta una actividad manual provocará que se invoque la estrategia de manejo de errores de flujo de trabajo. Esto puede provocar, por ejemplo, que la actividad asociada al método fallido se reintente una serie de veces. Por este motivo, las funciones de asignación asociadas a las estrategias de asignación de actividades manuales o de decisión no deberán, en general, lanzar excepciones a menos que se produzca una situación irrecuperable.

La aplicación debe implementar la interfaz de devolución de llamada `curam.util.workflow.impl.WorkResolver` para determinar qué tareas se asignan en la aplicación. La propiedad de aplicación curam.custom.workflow.workresolver debe hacer referencia a la implementación de la clase `curam.util.workflow.impl.WorkResolver` de la aplicación, ya que el motor de flujo de trabajo utilizará esta propiedad para determinar la función correcta para asignar la tarea.

La clase `curam.util.workflow.impl.WorkResolver` tiene sobrecargado el método `resolveWork` porque los diversos tipos de estrategia de asignación devuelven los destinos de asignación en distintos formatos. No obstante, esto es un detalle de implementación del que no tienen que preocuparse los desarrolladores de clases personalizadas de resolución de trabajo, sobre todo porque el procesamiento de negocio de todas las versiones del método debería ser el mismo.

```
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);

            ...
      }
```

Para paliar este problema, `curam.core.sl.impl.DefaultWorkResolverAdapter` proporciona un mecanismo más cómodo de implementar un programa de resolución de trabajo. Esta clase implementa los distintos métodos y convierte sus parámetros de entrada en listas de destino de asignación, permitiendo a los desarrolladores de lógica de resolución de trabajo personalizada extender esta clase e implementar un método que se invoca independientemente del origen de los destinos de asignación.

```
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);

            ...
      }
```

Además de esta clase de adaptador, la aplicación se distribuye con una implementación de resolución de trabajo que ya va lista para utilizarse. Dicha clase se denomina `curam.core.sl.impl.DefaultWorkResolver` y también sirve de ejemplo de cómo extender el adaptador.

9.4.5 Información de entorno de ejecución

Cuando se ejecuta una actividad manual, el motor de flujo de trabajo procesa la estrategia de asignación definida en los metadatos para recuperar la lista de destinos de asignación de esa tarea. Si la estrategia de asignación es de tipo función, el motor de flujo de trabajo procesa las correlaciones de entrada definidas para la función de asignación asociada y la invoca para recuperar la lista de destinos de asignación. Si la estrategia de asignación es de tipo regla, el motor de flujo de trabajo procesa las correlaciones del conjunto de reglas especificado y llama al motor de reglas para que ejecute el conjunto de reglas para recuperar la lista de destinos de asignación. Si la estrategia de asignación es de tipo destino, el destino de asignación es simplemente el que se especifica en los metadatos y no se requiere proceso adicional.

Tal como se describe en los metadatos de un proceso de flujo de trabajo (consulte Capítulo 3, “Metadatos de una definición de proceso”, en la página 13), puede especificarse una estrategia de asignación de errores para un proceso. Esta estrategia se procesa y se utiliza cuando la invocación de la estrategia de asignación asociada a la tarea no devuelve ningún destino de asignación.

El motor de flujo de trabajo utiliza después la propiedad curam.custom.workflow.workresolver para determinar la implementación de la función utilizada para asignar tareas en la aplicación. Luego el motor de flujo de trabajo invoca esta función pasándole la lista de destinos de asignación que determine la estrategia de asignación y también los detalles de la tarea que se va a asignar.

Una vez invocado el programa de resolución de trabajos para la tarea, el motor de flujo de trabajo hace una llamada al método `checkTaskAssignment` de la clase `curam.core.sl.impl.TaskAssignmentChecker`. Esta función comprueba el estado de la asignación de la tarea (es decir, para asegurarse de que se ha asignado a al menos un usuario u objeto organizativo (organización, puesto o trabajo) o a una cola de trabajos). Si la tarea no se ha asignado, la propiedad de aplicación curam.workflow.defaultworkqueue se examina para ver qué se ha especificado como cola de trabajos predeterminada para el flujo de trabajo. Entonces la tarea se asigna a esa cola de trabajos.

Si la tarea se ha asignado a un único usuario después de haberse resuelto el trabajo, el sistema comprobará el valor de la propiedad de aplicación curam.workflow.automaticallyaddtasktousertasks. Este distintivo controla si el sistema va a añadir automáticamente la tarea especificada que se está procesando a la lista de las tareas de ese usuario para permitirle que trabaje en ella. El valor predeterminado de la propiedad es `NO`, pero si se ha especificado `YES`, el sistema añadirá automáticamente esta tarea a la lista Mis tareas del usuario en su bandeja de entrada para permitirle que trabaje en ella.

9.4.6 Descripción de WDO de contexto

El objeto de datos de flujo de trabajo Context_Task está disponible a las correlaciones de función de asignación y de conjunto de reglas de asignación. Este objeto de datos de flujo de trabajo de contexto y su atributo ya se han descrito anteriormente (consulte 9.3.5, "Descripción de WDO de contexto", en la página 67).

9.5 Asociaciones de objeto de negocio

Las actividades manuales y el flujo de trabajo en general realizan operaciones sobre entidades que existen en la aplicación. Por este motivo, puede ser útil asociar una tarea con las entidades relacionadas para dicho proceso. Las asociaciones de objeto de negocio proporcionan en esencia enlaces entre una tarea y cualquier entidad de aplicación que resulte de interés para dicho proceso. Entre los ejemplos más representativos de ello en Cúram se incluyen las entidades Caso y Asunto.

9.5.1 Metadatos

```
<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
Contiene los detalles de todas las asociaciones de objeto de negocio que se han especificado para la actividad manual.

biz-object-association
Contiene los detalles de una asociación de objeto de negocio que se ha especificado para esa actividad manual. Incluye el tipo de objeto de negocio y la correlación de atributo de objeto de datos de flujo de trabajo asociada a ese tipo. Esta correlación de atributo de objeto de datos de flujo de trabajo representa el identificador exclusivo del objeto de negocio en la asociación (es decir, para una asociación de objeto de negocio de tipo Case, esto representaría el identificador exclusivo del caso vinculado a la tarea).

biz-object-type
Detalla el tipo de objeto de negocio real para la asociación de objeto de negocio de la actividad manual. El tipo de objeto de negocio debe seleccionarse en la herramienta de definición de proceso y se obtiene de la tabla de códigos `BusinessObjectType`.

9.5.2 Validaciones

- El tipo de objeto de negocio especificado deberá ser un código válido de tabla de códigos contenido en la tabla de códigos `BusinessObjectType`.

- El atributo de objeto de datos de flujo de trabajo correlacionado con el tipo de objeto de negocio de una asociación de objeto de negocio de actividad manual deberá ser válido. Este tipo de atributo deberá poder asignarse a un tipo LONG, porque representa una correlación con un identificador exclusivo (p.ej. un identificador de caso o identificador de participante).
- Si un elemento indexado de un objeto de datos de flujo trabajo de lista (es decir, PersonDetailsList[Context_Loop.loopCount]) se utiliza en una correlación de asociación de objeto de negocio, el objeto de datos de flujo de trabajo deberá ser un objeto de datos de flujo de trabajo de lista y la actividad que contiene la correlación deberá estar contenida en un bucle.
- Si el objeto de datos de flujo de trabajo `Context_Parallel` se utiliza en una correlación de asociación de objeto de negocio, la actividad que contenga la correlación deberá ser una actividad manual `paralela`.
- Si un elemento indexado procedente del de objeto de datos de flujo de trabajo de lista Parallel se utiliza en una correlación de asociación de objeto de negocio, la actividad que contenga la correlación deberá ser una actividad paralela (es decir, ParallelListWDO[Context_Parallel.occurrenceCount]). El objeto de datos de flujo de trabajo indexado por el el objeto objeto de datos de flujo de trabajo Context_Parallel deberá ser el objeto de datos de flujo de trabajo de lista de la actividad paralela.

9.5.3 Código

API de administración de asociaciones de objetos de negocio

En la clase `BusinessObjectAssociationAdmin` se proporcionan una serie de funciones que permiten la manipulación de asociaciones de objetos de negocio. Para obtener más detalles de las funciones disponibles, consulte la especificación Javadoc asociada a la clase `BusinessObjectAssociationAdmin`.

9.5.4 Información de entorno de ejecución

Las asociaciones de objeto de negocio no tienen impacto funcional en la ejecución de una actividad manual. El motor de flujo de trabajo se limita a examinar los metadatos y colocar un registro en la entidad `BizObjAssociation` por cada asociación de objeto de negocio especificada. El tipo de objeto de negocio, el valor de la correlación de atributo de objeto de datos de flujo de trabajo y el identificador de la tarea recién creada asociada a la actividad manual se utilizan todos en la creación de este registro.

9.6 Espera de suceso

Puesto que una actividad manual requiere que un usuario lleve a cabo alguna acción antes de que pueda completarse y de que el proceso puede continuar, debe haber alguna forma de notificar al motor de flujo de trabajo cuándo se ha realizado el trabajo necesario. Puesto que esta semántica es similar a la de la actividad de espera de suceso, se aprovecha el mecanismo de espera de suceso en las actividades manuales.

9.6.1 Requisitos previos

- Los detalles de una espera de suceso y sus metadatos asociados (que también se aplican a una actividad manual) pueden encontrarse en Capítulo 8, “Espera de suceso”, en la página 49.

9.6.2 Descripción de WDO de contexto

El objeto de datos de flujo de trabajo Context_Task está disponible para su uso en las correlaciones de entrada para las funciones de fecha límite asociadas a la espera de suceso de una actividad manual. Está disponible para las correlaciones de entrada asociadas a las correlaciones de entrada de regla o de función de asignación. También está disponible para su uso como una correlación de datos de coincidencia de sucesos de una espera de suceso especificada asociada a una actividad manual. Este objeto de datos de flujo de trabajo de contexto y su atributo ya se han descrito anteriormente (consulte 9.3.5, “Descripción de WDO de contexto”, en la página 67).

Capítulo 10. Decisión

10.1 Requisitos previos

- Los detalles básicos comunes a todos los tipos de actividades soportadas en un flujo de trabajo Cúram se describen en Capítulo 6, "Actividad base", en la página 31 y son aplicables a la actividad de decisión descrita aquí.
- También hay construcciones de metadatos de flujo de trabajo que son comunes entre las actividades manuales y la actividades de decisión (es decir, estrategia de asignación, asunto de tarea, fecha límite de tarea, etc.). Los detalles de las mismas pueden encontrarse en Capítulo 9, "Manual", en la página 61.

10.2 Descripción general

Un requisito habitual en los procesos de negocio es hacer que un agente humano tome decisiones que tienen respuestas simples. Un ejemplo de ese tipo de decisiones es aprobar o rechazar un caso o proporcionar alguna información simple como la edad del demandante. La utilización de actividades manuales para solicitar tal información requeriría que en la aplicación hubiese una pantalla de interfaz de usuario distinta por cada pregunta. Esto sería engorroso y, puesto que las definiciones pueden cambiar con el tiempo, esas pantallas de interfaz de usuario serían en cierto modo temporales.

La actividad Decisión es una especialización de una actividad manual que controla una interfaz de usuario controlada por metadatos para formular preguntas simples. Las preguntas y las posibles respuestas están en los metadatos de la actividad, lo que permite utilizar una única interfaz de usuario para un amplio abanico de preguntas. En la actualidad se soportan dos tipos de pregunta. Son las preguntas de selección múltiple y las preguntas que requieren una respuesta que se puede proporcionar en un campo de la interfaz de usuario.

10.3 Detalles de tarea

De forma similar a como ocurre con una Capítulo 9, "Manual", en la página 61, las actividades de decisión notifican a los usuarios que tienen que realizar algún trabajo, y les asignan una tarea basándose en la estrategia de asignación definida. La tarea se enlazará automáticamente a una página de interfaz de usuario de la aplicación que ensambla la pregunta de decisión procedente de los metadatos de la pregunta de actividad de decisión y avanza el flujo una vez proporcionada la respuesta de decisión. Por tanto, una actividad de decisión solo podrá tener una acción de tarea asociada y no requerirá que se defina ninguna página de acción para esa acción.

Además de la acción de tarea, una actividad de decisión puede tener cero o más acciones secundarias asociadas a ella. Las acciones secundarias contienen un enlace a una página que puede proporcionar información complementaria para ayudar al usuario a responder a la pregunta de decisión.

10.3.1 Metadatos

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

    ...

      <allocation-strategy type="target" identifier="1" />
      <message>
        <message-text>
          <localized-text>
            <locale language="es">
            Decida la edad del usuario %1s para el caso %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="es">
              Página de inicio de participante %1n para el caso %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="es">Ver detalles del caso.</locale>
              </localized-text>
            </message-text>
          </message>
        </secondary-action>
        <secondary-action page-id="Case_viewUserDetails">
          <message>
            <message-text>
              <localized-text>
                <locale language="es">Ver detalles del usuario %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
Describe la estrategia de asignación que se utiliza para determinar el usuario asignado a la tarea asociada. Para obtener detalles sobre las estrategias de asignación, consulte 9.4, “Estrategia de asignación”, en la página 67.

message
Representa el mensaje de asunto parametrizado de la tarea creada. Puede obtener los detalles completos sobre los mensajes parametrizados consultando Capítulo 9, “Manual”, en la página 61.

decision-action
Representa el mensaje de texto de acción parametrizada asociado a la tarea. El usuario pulsará en este texto de acción para abrir la pantalla de decisión de interfaz de usuario generada automáticamente con la pregunta relevante.

deadline
Describe los detalles de fecha límite de la actividad de decisión. Si no se proporciona una respuesta a la actividad de decisión dentro del plazo especificado, se invocará el método manejador de fecha límite asociado. Para obtener más detalles sobre las fechas límite, consulte 8.4, “Fecha límite”, en la página 52

secondary-actions
Describe las acciones secundarias opcionales que pueden incluirse en la actividad de decisión.

secondary-action
Una acción secundaria contiene un mensaje parametrizado y un enlace parametrizado a información de apoyo para ayudar al usuario a responder a la pregunta de la decisión. Para obtener más detalles sobre mensajes y enlaces parametrizados en acciones, consulte 9.3.1, “Metadatos”, en la página 62

page-id
Representa el identificador de la página de Cúram de destino que contiene la información complementaria que se enlaza a la acción secundaria.

multiple-occurring-action
Indica que esta acción secundaria representa una acción que ocurre varias veces. Esto significa que si se especifican estos metadatos para una acción secundaria, el motor de flujo de trabajo creará un registro de acción secundaria por cada elemento del objeto de datos de flujo de trabajo de lista especificado como acción con varias ocurrencias cuando se ejecute dicha actividad.

Debe tenerse en cuenta que cuando la acción que se produce varias veces se especifica para una acción secundaria, a continuación debe utilizarse un atributo del objeto de datos de flujo de trabajo asociado como parámetro de enlace para la acción secundaria.

list-wdo-name
Nombre del objeto de datos de flujo de trabajo de lista para su uso con la acción de ocurrencia múltiple.

10.3.2 Validaciones

- Deberá definirse un asunto de actividad.
- Cada atributo de objeto de datos de flujo de trabajo correlacionado con el asunto de la actividad de decisión deberá ser un atributo de objeto de datos de flujo de trabajo válido.
- Si un elemento indexado de un objeto de datos de flujo trabajo de lista (es decir, CaseList[Context_Loop.loopCount]) se utiliza como parámetro de texto del asunto de la decisión, el objeto de datos de flujo de trabajo deberá ser un objeto de datos de flujo de trabajo de lista y la actividad que contiene la correlación deberá estar contenida en un bucle.

- Si el objeto de datos de flujo de trabajo `Context_Parallel` se utiliza como parámetro del texto de asunto de la decisión, la actividad que contenga la correlación deberá ser una actividad de decisión `paralela`.
- Si un elemento indexado procedente del de objeto de datos de flujo de trabajo de lista Parallel se utiliza como parámetro del texto de asunto de la decisión, la actividad que contenga la correlación deberá ser una actividad paralela (es decir, ParallelListWDO[Context_Parallel.occurrenceCount]). El objeto de datos de flujo de trabajo indexado por el el objeto objeto de datos de flujo de trabajo Context_Parallel deberá ser el objeto de datos de flujo de trabajo de lista de la actividad paralela.
- El número de marcadores de posición utilizado en el texto del asunto y en el texto de la acción deberá ser igual al número de atributos de objeto de datos de flujo de trabajo correlacionados (para todos los entornos locales).
- Si un elemento indexado de un objeto de datos de flujo trabajo de lista (es decir, CaseList[Context_Loop.loopCount]) se utiliza como parámetro de texto de la acción de tarea de la decisión, el objeto de datos de flujo de trabajo deberá ser un objeto de datos de flujo de trabajo de lista y la actividad que contiene la correlación deberá estar contenida en un bucle.
- Si el objeto de datos de flujo de trabajo `Context_Parallel` se utiliza como parámetro del texto de acción de la decisión, la actividad que contenga la correlación deberá ser una actividad de decisión `paralela`.
- Si un elemento indexado procedente del de objeto de datos de flujo de trabajo de lista Parallel se utiliza como parámetro del texto de acción de la decisión, la actividad que contenga la correlación deberá ser una actividad paralela (es decir, ParallelListWDO[Context_Parallel.occurrenceCount]). El objeto de datos de flujo de trabajo indexado por el el objeto objeto de datos de flujo de trabajo Context_Parallel deberá ser el objeto de datos de flujo de trabajo de lista de la actividad paralela.
- Deberá definirse una estrategia de asignación.
- El destino de asignación, la función o el conjunto de reglas especificados como estrategia de asignación deberán ser válidos. Si la asignación es de tipo función, deberá ser un método de negocio de Cúram válido, y deberá existir en la vía de acceso de clases de la aplicación. Si la asignación es de de tipo regla, deberá ser un conjunto de reglas de asignación válido.
- Si se especifica el manejador de fecha límite opcional, este deberá ser un método de negocio de Cúram válido.
- Todas las correlaciones de entrada del manejador de fecha límite deberán ser válidas. Esto significa que todos los campos de parámetro de entrada que necesita el método especificado se correlacionarán con atributos de objeto de datos de flujo de trabajo válidos del tipo correcto.
- Cada acción secundaria deberá tener especificado un enlace de página que no puede contener espacios en blanco.
- Cada acción secundaria deberá tener especificado un mensaje.
- El texto del mensaje de acción secundaria deberá contener un número de marcadores de posición igual al número de parámetros de mensaje especificados.
- Los parámetros de mensaje de acción secundaria deberán estar correlacionados con atributos de objeto de datos de flujo de trabajo válidos del tipo correcto.
- Los parámetros de enlace de página de acción secundaria deberán estar correlacionados con atributos de objeto de datos de flujo de trabajo válidos.
- Si un elemento indexado de un objeto de datos de flujo trabajo de lista (es decir, ChildDependents[Context_Loop.loopCount]) se utiliza en las correlaciones de texto o enlace de acción secundaria, el objeto de datos de flujo de trabajo deberá ser un objeto de datos de flujo de trabajo de lista y la actividad que contiene la correlación deberá estar contenida en un bucle.
- Si el objeto de datos de flujo de trabajo `Context_Parallel` se utiliza en las correlaciones de parámetro de enlace o texto de acción secundaria, la actividad que contenga la correlación deberá ser una actividad de decisión `paralela`.
- Si un elemento indexado procedente del de objeto de datos de flujo de trabajo de lista Parallel en las correlaciones de parámetro de enlace o texto de acción secundaria, la actividad que contenga las correlaciones deberá ser una actividad paralela (es decir,

ParallelListWDO[Context_Parallel.occurrenceCount]). El objeto de datos de flujo de trabajo indexado por el el objeto objeto de datos de flujo de trabajo Context_Parallel deberá ser el objeto de datos de flujo de trabajo de lista de la actividad paralela.

- El objeto de datos de flujo de trabajo especificado para su uso en la acción de ocurrencia múltiple deberá ser un objeto de datos de flujo de trabajo válido en el contexto de la definición de proceso de flujo de trabajo contenedora. También deberá ser de tipo List.
- Al menos un atributo del objeto de datos de flujo de trabajo de lista de acción de ocurrencia múltiple deberá utilizarse en los parámetros de enlace especificados para una acción de ocurrencia múltiple.

10.3.3 Información de entorno de ejecución

Cuando se ejecuta una actividad de decisión, el motor de flujo de trabajo crea la tarea asociada. Se recogerá y almacenará una instantánea de los datos de objeto de datos de flujo de trabajo necesarios para el asunto de la actividad de decisión y los parámetros de texto de la acción, y cualquier mensaje de texto de acción secundaria y parámetros de enlace. La estrategia de asignación asociada a la actividad de decisión se invoca para determinar el/los usuario(s) que quien(es) se asignará la tarea de decisión. El motor de flujo también crea una espera de suceso para el suceso `DECISION.MADE` con el identificador de tarea asociada como dato de coincidencia del suceso. El flujo de trabajo se detendrá en este punto a la espera de la generación de este suceso, que indicará el resultado de la decisión tomada.

10.4 Detalles de pregunta

La actividad de decisión soporta actualmente preguntas de selección múltiple y de texto libre como formatos de pregunta. La página de decisión generada automáticamente examina el formato de pregunta requerido y genera la pregunta relevante a partir de los metadatos de la pregunta una vez que el usuario pulsa en la acción asociada a la tarea.

10.4.1 Metadatos

10.4.1.1 Opción múltiple

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

    ...

      <question>
        <message>
          <message-text>
            <localized-text>
              <locale language="es">
              ¿El demandante %1s para el caso %2n es mayor de 18 años?
              </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="es">Sí</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="es">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
Representa la pregunta asociada a la actividad de decisión, que, en el caso de una pregunta de selección múltiple, contiene los metadatos descritos a continuación.

message
Representa los parámetros de texto de la pregunta a formular para todos los entornos locales.

answers
Representa una lista de respuestas que el usuario puede elegir para una pregunta de selección múltiple.

multiple-selection
Representa un distintivo que indica si el usuario puede seleccionar varias respuestas a partir de las proporcionadas, o si sólo se puede seleccionar una.

answer
Representa una respuesta que el usuario puede seleccionar. Debe proporcionarse como mínimo una respuesta a una pregunta de selección múltiple.

name Representa el nombre de la respuesta. Una vez que el usuario selecciona una respuesta o respuestas, los nombres de las respuestas seleccionadas se pasan al motor de flujo de trabajo y el proceso continúa. Puesto que el motor trata dichas respuestas de forma similar a como trata los atributos de un objeto de datos de flujo de trabajo, los nombres de las respuestas deberán ser identificadores Java válidos.

answer-text
Representa el texto de respuesta que el usuario puede seleccionar para todos los entornos locales.

choice-output-mapping
Este código incluye los metadatos que describen cómo se persiste la salida de una respuesta de selección múltiple.

wdo-attribute
El nombre del atributo de objeto de datos de flujo de trabajo utilizado para almacenar el valor de la respuesta de selección múltiple.

selected-value
Si se especifica, el valor de este elemento se persistirá en el atributo del objeto de datos de flujo de trabajo si esa respuesta ha sido seleccionada por el usuario. Si el atributo del objeto de datos de flujo de trabajo es un tipo booleano, no será necesario especificar este valor, pues se le asignará el valor predeterminado `true`.

not-selected-value
Si se especifica, el valor de este elemento se persistirá en el atributo del objeto de datos de flujo de trabajo si esa respuesta no ha sido seleccionada por el usuario. Si el atributo del objeto de datos de flujo de trabajo es un tipo booleano, no será necesario especificar este valor, pues se le asignará el valor predeterminado `false`.

10.4.1.2 Texto libre

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

    ...

      <question>
        <message>
          <message-text>
            <localized-text>;
              <locale language="es">
              ¿Qué edad tiene el demandante, %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
Representa la pregunta asociada a esta actividad de decisión, que, en el caso de una pregunta de texto libre, contiene los metadatos descritos a continuación.

message
Representa los parámetros de texto de la pregunta a formular para todos los entornos locales.

free-text
Contiene los detalles de la respuesta de texto libre que debe proporcionar el usuario.

type Representa el tipo de datos necesario de la respuesta de texto libre que se debe proporcionar.

wdo-attribute
Representa el atributo de objeto de datos de flujo de trabajo que devuelve la respuesta de texto libre correlacionada al motor de flujo de trabajo.

10.4.2 Validaciones

- Deberá especificarse el formato de la respuesta y el texto de la pregunta para una actividad de decisión.
- El número de marcadores de posición utilizados en el texto de la pregunta deberá ser igual al número de atributos de objeto de datos de flujo de trabajo correlacionados (para todos los entornos locales).
- Si un elemento indexado de un objeto de datos de flujo trabajo de lista (es decir, CaseList[Context_Loop.loopCount]) se utiliza como parámetro de texto de pregunta, el objeto de datos de flujo de trabajo deberá ser un objeto de datos de flujo de trabajo de lista y la actividad que contiene la correlación deberá estar contenida en un bucle.
- Si el objeto de datos de flujo de trabajo `Context_Parallel` se utiliza como parámetro de texto de pregunta, la actividad que contenga la correlación deberá ser una actividad de decisión `paralela`.
- Si un elemento indexado procedente del de objeto de datos de flujo de trabajo de lista Parallel se utiliza como parámetro de texto de pregunta, la actividad que contenga la correlación deberá ser una actividad paralela (es decir, ParallelListWDO[Context_Parallel.occurrenceCount]). El objeto de datos de

flujo de trabajo indexado por el el objeto objeto de datos de flujo de trabajo Context_Parallel deberá ser el objeto de datos de flujo de trabajo de lista de la actividad paralela.

- En el caso de una pregunta con un formato de respuesta de texto de formulario abierto, el tipo de datos de la respuesta deberá especificarse y el atributo de objeto de datos de flujo de trabajo correlacionado deberá ser válido y coincidir con el tipo de datos de la respuesta. El atributo de objeto de datos de flujo de trabajo correlacionado no podrá ser un atributo de objeto de datos de flujo de trabajo constante.
- En el caso de una pregunta con formato de respuesta de lista (List), deberá listarse al menos una opción de respuesta. Todos los nombres de respuesta deberán ser nombres de atributo Java válidos.

10.4.3 Información de entorno de ejecución

Cuando se proporciona una respuesta a una pregunta de actividad de decisión, se genera el suceso `DECISION.MADE` con el identificador de tarea de la tarea de actividad de decisión utilizada como dato de coincidencia del suceso. El manejador de sucesos de flujo de trabajo procesará de este suceso, lo que hará que proceso de flujo de trabajo avance.

Si la respuesta proporcionada no es una respuesta de texto libre, se correlacionará con el atributo de objeto de datos de flujo de trabajo especificado para su uso posterior en el proceso donde sea necesario.

10.4.4 Descripción de WDO de contexto

El objeto de datos de flujo de trabajo Context_Decision está disponible para su uso en una condición de elemento de datos o de función (consulte Capítulo 16, “Condiciones”, en la página 113) para una transición desde una actividad de decisión. Los atributos disponibles dependerán del formato de respuesta definido para la actividad.

Respuesta de texto libre

Si el formato de la respuesta es texto libre, el atributo disponible será:

Context_Decision.value

El valor de la respuesta de texto libre proporcionada. Puede utilizarse en condiciones de transición y puede correlacionarse con un atributo de objeto de datos de flujo de trabajo especificado.

Respuesta de opción múltiple

En este caso, el objeto de datos de flujo de trabajo Context_Decision se llenará con atributos para cada una de las respuestas disponibles, cada uno de tipo booleano. Esto indica si esa respuesta se ha seleccionado o no. En el fragmento de metadatos de respuesta de opción múltiple anterior, (10.4.1.1, “Opción múltiple”, en la página 82, si el usuario seleccionase la primera respuesta (Sí/Yes), esto se reflejaría estableciendo a `true` el siguiente atributo del objeto de datos de flujo de trabajo Context_Decision:

Context_Decision.yesAnswer

Representa un booleano que indica si se ha seleccionado la respuesta Yes (Sí) para la pregunta. Esto solo se puede utilizar en las condiciones de transición que parten de la actividad de decisión.

De forma alternativa, si el usuario seleccionase la segunda respuesta (No), esto se reflejaría estableciendo a `true` el siguiente atributo del objeto de datos de flujo de trabajo Context_Decision:

Context_Decision.noAnswer

Representa un booleano que indica si se ha seleccionado la respuesta No para la pregunta. Asimismo, esto solo se puede utilizar en las condiciones de transición que parten de la actividad de decisión.

Capítulo 11. Subflujo

11.1 Requisitos previos

- Los detalles básicos comunes a todos los tipos de actividades soportadas en un flujo de trabajo Cúram se describen en Capítulo 6, "Actividad base", en la página 31 y son aplicables a la actividad de subflujo descrita aquí.

11.2 Descripción general

Al diseñar procesos de negocio complejos, puede que se vuelvan demasiado grandes para gestionarlos como definiciones de proceso monolíticas. Una actividad de subflujo permite que otra definición de proceso se promulgue como parte de otro proceso.

Puede que sea una decisión prudente diseñar definiciones de proceso como un conjunto de subflujos independientemente de que preocupe el tamaño. Esto permitiría que las secciones del proceso de negocio cambiasen si afectar a otras. Además, los procesos de subflujo podrían actuar como componentes reutilizables que los clientes pueden reutilizar al crear sus propias definiciones de proceso de orden superior.

11.3 Proceso de subflujo

Para promulgar un proceso como subflujo, la actividad de subflujo debe identificar por nombre el proceso que se va a promulgar. Igual que con los otros mecanismos de promulgación de procesos, se promulgará la última versión publicada del proceso.

Los subflujos se pueden promulgar *de forma síncrona*. Esto significa que la rama del flujo de trabajo padre que contiene la actividad de subflujo que ha iniciado el proceso de subflujo esperará a que dicho proceso de subflujo finalice antes de continuar.

De forma alternativa, un subflujo puede promulgarse *de forma asíncrona*. Esto significa que, una vez que la actividad de subflujo inicia el proceso de subflujo, la rama que contiene la actividad de subflujo proseguirá de forma inmediata sin que el resultado del proceso de subflujo tenga efecto alguno sobre el proceso padre.

11.3.1 Metadatos

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

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

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

subflow

workflow-process
El nombre del proceso de flujo de trabajo a iniciar cuando se ejecuta la actividad. Los nombres de proceso distinguen entre mayúsculas y minúsculas, y el nombre de proceso especificado aquí debe coincidir exactamente con el del proceso para iniciar como subflujo.

synchronous

Un distintivo que indica si el subflujo debe ejecutarse de manera síncrona o no (consulte: 11.3, “Proceso de subflujo”, en la página 87) en relación con su proceso padre.

11.3.2 Validaciones

- Deberá especificarse un proceso de flujo de trabajo para la actividad de subflujo.
- El proceso de flujo de trabajo especificado como subflujo deberá tener al menos una versión publicada.

11.4 Correlaciones de entrada

Cuando un subflujo se promulga, se le proporcionan datos de los objetos de datos de flujo de trabajo del proceso padre. La actividad de subflujo define la correlación entre los objetos de datos de flujo de trabajo del proceso padre y los datos de promulgación del subflujo.

11.4.1 Metadatos

```
<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

Especifica cómo se correlacionan los datos desde el proceso que se está ejecutando actualmente a un subproceso como datos de promulgación cuando se ha iniciado el subproceso. El proceso especificado como subflujo no puede tener ningún atributo de objeto de datos de flujo de trabajo marcado como obligatorio en la promulgación, en cuyo caso no serán necesarias correlaciones de entrada.

mapping

Un elemento `mapping` (correlación) representa los datos que se envían desde un atributo de objeto de datos de flujo de trabajo a un atributo en el proceso que se promulga como subflujo. Si fuera necesaria una lista de datos para promulgar el proceso de subflujo, podrán utilizarse a tal fin los atributos de los objetos de datos de flujo de trabajo de lista. El número de correlaciones

especificadas viene determinado por cuántos atributos se marquen como obligatorios en la promulgación en el proceso de subflujo, ya que todos esos atributos deben informarse cuando el proceso se inicia.

source-attribute
Representa un atributo de objeto de datos de flujo de trabajo del proceso padre que se utiliza para llenar el atributo asociado en el subflujo cuando se promulga.

target-attribute
Representa un atributo de objeto de datos de flujo de trabajo del subflujo que debe llenarse con datos del atributo asociado del proceso padre en el momento de la promulgación.

source/target-attribute

wdo-name
Representa el nombre de un objeto de datos de flujo de trabajo de Cúram tal y como se describe en Capítulo 4, “Objetos de datos de flujo de trabajo”, en la página 17.

name Representa el nombre de un atributo de objeto de datos de flujo de trabajo de Cúram tal y como se describe en Capítulo 4, “Objetos de datos de flujo de trabajo”, en la página 17.

11.4.2 Validaciones

- Cada atributo de objeto de datos de flujo de trabajo marcado como *obligatorio para la promulgación* en el subflujo deberá especificarse en las correlaciones de entrada. Si no se ha marcado ningún objeto de datos de flujo de trabajo como obligatorio para la promulgación en el proceso de subflujo, no deberán especificarse correlaciones de entrada.
- El tipo de datos del atributo de objeto de datos de flujo de trabajo especificado por la etiqueta `target-attribute` deberá coincidir con o ser asignable desde el atributo especificado por la etiqueta `source-attribute`.
- Si un elemento indexado de un objeto de datos de flujo de trabajo de lista (es decir, PersonDetailsList[Context_Loop.loopCount]) se especifica en la etiqueta `source-attribute` de la correlación de entrada del subflujo, dicho objeto de datos de flujo de trabajo deberá ser un objeto de datos de flujo de trabajo de lista y la actividad de subflujo que contiene la correlación de entrada deberá estar contenida dentro de un bucle. El tipo de datos del atributo de objeto de datos de flujo de trabajo especificado por la etiqueta `target-attribute` deberá coincidir con o ser asignable desde el atributo especificado por la etiqueta `source-attribute`.
- Si la correlación de entrada del subflujo especificada utiliza un objeto de datos de flujo de trabajo de lista, los atributos del objeto de datos de flujo de trabajo tanto del `source-attribute` (atributo de origen) padre como del `target-attribute` (atributo de destino) del proceso de subflujo deberán ser objetos de datos de flujo de trabajo de lista.

11.5 Correlaciones de salida

Las correlaciones de salida sólo son aplicables a actividades de subflujo *síncrono*, ya que los subflujos síncronos pueden continuar sin completar la actividad. Los datos se proporcionan al proceso padre desde la actividad de subflujo cuando completa. La actividad de subflujo define la correlación entre un atributo de objeto de datos de flujo de trabajo de subflujo y el atributo de objeto de datos de flujo de trabajo del proceso padre.

11.5.1 Metadatos

```
<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
Especifica cómo se correlacionan los datos desde el subproceso invocado al proceso padre cuando el sub-proceso ha finalizado. Es posible que el proceso especificado como subflujo no tenga ninguna correlación de salida definida, en cuyo caso el subflujo finalizará de forma normal.

mapping
Representa los datos que se envían desde un atributo de objeto de datos de flujo de trabajo de subflujo a un atributo del proceso padre. Si se envía una lista de datos desde el proceso de subflujo al proceso padre, podrán utilizarse a este efecto los atributos de los objetos de datos de flujo de trabajo de lista. El número de correlaciones especificadas está determinado por el número de correlaciones de salida especificadas.

source-attribute
Representa un atributo de objeto de datos de flujo de trabajo del proceso de subflujo que se utiliza para llenar el atributo asociado del proceso padre tras la finalización.

target-attribute
Representa un atributo de objeto de datos de flujo de trabajo del padre que debe llenarse con datos del atributo asociado del proceso de subflujo cuando se ha completado.

source/target-attribute

wdo-name
Representa el nombre de un objeto de datos de flujo de trabajo de Cúram (tal y como se describe en Capítulo 4, “Objetos de datos de flujo de trabajo”, en la página 17).

name Representa el nombre de un atributo de objeto de datos de flujo de trabajo de Cúram (tal y como se describe en Capítulo 4, “Objetos de datos de flujo de trabajo”, en la página 17).

11.5.2 Validaciones

- Los atributos de objeto de datos de flujo de trabajo del `target-attribute` (atributo de destino) padre y del `source-attribute` (atributo de origen) del subflujo utilizados en la correlación de salida de subflujo deberán ser válidos en el contexto de la definición de proceso contenedora.
- El tipo de datos del atributo de objeto de datos de flujo de trabajo especificado por la etiqueta padre `target-attribute` deberá coincidir con o ser asignable desde el atributo especificado por la etiqueta del subflujo `source-attribute`.
- Si la correlación de salida de subflujo especificada utiliza un objeto de datos de flujo de trabajo, los atributos correlacionados de objeto de datos de flujo de trabajo tanto del `target-attribute` padre como del `source-attribute` del proceso de subflujo deberán ser de tipo lista.

Capítulo 12. Inicio de bucle y fin de bucle

12.1 Requisitos previos

- Los detalles básicos comunes a todos los tipos de actividades soportadas en un flujo de trabajo Cúram se describen en Capítulo 6, "Actividad base", en la página 31 y son aplicables a las actividades de inicio/fin de bucle descritas aquí.

12.2 Descripción general

En muchos procesos de negocio, es necesario *repetir* hasta que se cumple alguna condición. En Cúram, esto se implementa mediante las actividades inicio de bucle (loop-begin-activity) y fin de bucle (loop-end-activity). Todas las actividades que se encuentran entre un inicio de bucle y su fin de bucle asociado se repiten hasta que el bucle finaliza.

En una definición de proceso, las actividades de inicio y fin de bucle van emparejadas, y los metadatos permiten que cada de inicio de bucle sepa cuál es su fin de bucle asociado y viceversa. Para añadir una secuencia de actividades a un bucle, se crea una transición a partir de la actividad de inicio de bucle hasta la primera actividad a repetir. Las actividades posteriores en la secuencia se enlazan utilizando transiciones tal y como se haría normalmente fuera de un bucle; sin embargo, la última actividad de la secuencia tendrá una transición a la actividad de fin de bucle. Suele tenderse a añadir también una transición desde la actividad de fin de bucle al inicio para crear el ciclo; sin embargo, esto es incorrecto y da lugar a una definición de proceso no válida.

Un bucle también debe especificar los criterios que seguirá el bucle para determinar si se debe terminar o no. Para soportar esto, un bucle en un flujo de Cúram tiene una condición de salida de bucle.

Un bucle podrá contener otro bucle siempre y cuando estén totalmente anidados y no se solapen entre sí. Esto garantiza que los bucles y, por lo tanto, la definición de proceso, siga siendo una estructura de bloque válida tal y como requiere el motor de flujo de trabajo de Cúram (consulte Capítulo 18, "Estructura del flujo de trabajo", en la página 121).

12.2.1 Tipo de bucle

Además de la condición de salida de bucle, un bucle también especifica si la condición debe probarse antes de que se ejecute el bucle (un bucle while) o al final de la ejecución del bucle (un bucle do-while). Un bucle while nunca podrá ejecutar las actividades en el bucle y saltar a la actividad que va a continuación del bucle si la condición de salida se cumple al inicio del bucle, mientras que un bucle do-while ejecutará las actividades del bucle al menos una vez.

12.3 Metadatos

12.3.1 Actividad de inicio de bucle

```
<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

El elemento `loop-type` (tipo de bucle) especifica cómo se va a ejecutar el bucle conforme a lo detallado en 12.2.1, "Tipo de bucle", en la página 91. Los dos únicos valores válidos para el atributo `name` (nombre) son `while` y `do-while`.

condition

La etiqueta `condition` especifica la condición que se va a evaluar en función de los valores del objeto de datos de flujo de trabajo (consulte Capítulo 4, "Objetos de datos de flujo de trabajo", en la página 17). Cuando los objetos de datos de flujo de trabajo de lista están presentes en el flujo de trabajo, al crearse una expresión de condición de bucle utilizando un objeto de datos de flujo de trabajo de lista, se hacen disponibles dos atributos que no forman parte de los metadatos de objeto de datos de flujo de trabajo Dichos atributos son los siguientes:

- *size()*: evalúa a un número (de tipo ENTERO) que indica el número de elementos de la lista.
- *isEmpty()*: evalúa a un indicador BOOLEANO que indica si la lista contiene elementos o no.

Los valores reales de los metadatos se utilizan en otros lugares de los metadatos de la definición de proceso y por tanto se describen en el correspondiente capítulo, Capítulo 16, "Condiciones", en la página 113.

block-endpoint-ref

El elemento `block-endpoint-ref` permite en este contexto que `loop-begin-activity` (actividad de fin de bucle) reconozca sus `loop-end-activity` (actividades de inicio de bucle) asociadas. Esta información es útil para el motor de flujo de trabajo cuando ejecuta el bucle. Por ejemplo, cuando una condición de salida de un bucle while evalúa a `true` antes de que ejecute el bucle, `block-endpoint-ref` indica al motor de flujo de trabajo que a qué actividad tiene que saltar para continuar con la ejecución del proceso.

12.3.2 Actividad de fin de bucle

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

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

  </loop-end-activity>
```

block-endpoint-ref

`block-endpoint-ref` permite en este contexto que `loop-end-activity` (actividad de fin de bucle) reconozca sus `loop-begin-activity` (actividades de inicio de bucle) asociadas. Esta información es útil para el motor de flujo de trabajo cuando ejecuta el bucle. Por ejemplo, si después de la ejecución de un bucle la condición de salida evalúa a false, `block-endpoint-ref` indica al motor de flujo de trabajo a qué actividad hay que saltar para empezar otra iteración del bucle.

12.4 Información de entorno de ejecución

Es de esperar que cualquier actividad dentro de un bucle se ejecute más de una vez durante la ejecución de una instancia de proceso. Para evitar que las sucesivas iteraciones corrompan los datos de la instancia de proceso de la actividad, cada instancia de actividad está asociada a una iteración específica, de modo que el motor de flujo de trabajo la identifica de forma exclusiva independientemente del número de veces que se ejecute el bucle.

12.5 Descripción de WDO de contexto

El objeto de datos de flujo de trabajo Context_Loop está disponible en las siguientes ocasiones:

- Cuando se crea una condición de bucle asociada a una actividad de inicio de bucle.
- Cuando se crean las condiciones de transición de salida a partir de una actividad de inicio de bucle, o de cualquier actividad contenida en un bucle (consulte Capítulo 16, “Condiciones”, en la página 113).
- Cuando se crean las correlaciones de entrada para cualquier actividad automática o actividad de subflujo dentro de un bucle.
- Cuando se crean las correlaciones de entrada para cualquier función de estrategia de asignación o función manejadora de fecha límite presente en una actividad dentro de un bucle.
- Cuando se especifica un parámetro de texto de asunto para una actividad manual o de decisión contenida en un bucle o para una notificación adjuntada a una actividad contenida en un bucle.
- Cuando se especifican parámetros de texto de acción para una actividad manual o de decisión contenida en un bucle o para una notificación adjuntada action una actividad contenida en un bucle.
- Cuando se especifica el identificador de una asociación de objeto de negocio para una actividad manual contenida en un bucle.
- Cuando se especifica un parámetro de texto de pregunta para una pregunta de selección múltiple o de formato libre para una actividad de decisión contenida en un bucle.
- Cuando se especifica un parámetro de texto de cuerpo para una notificación adjunta a una actividad contenida en un bucle.

Los atributos disponibles del objeto de datos de flujo de trabajo de Context_Loop son:

Context_Loop.loopCount
El número de veces que un bucle se ha iterado.

Capítulo 13. Paralela

13.1 Requisitos previos

- Los detalles básicos comunes a todos los tipos de actividades soportadas en un flujo de trabajo Cúram se describen en Capítulo 6, “Actividad base”, en la página 31 y son aplicables a la actividad de paralela descrita aquí.
- Como las actividades paralelas engloban actividades existentes en una definición de proceso de flujo de trabajo, los metadatos descritos en Capítulo 9, “Manual”, en la página 61 y Capítulo 10, “Decisión”, en la página 77 también son relevantes para la actividad paralela descrita aquí.

13.2 Descripción general

El un proceso de negocio, puede ser necesario enviar a la vez varias tareas a distintos agentes humanos para agilizar el proceso global.Cuando se conocen una serie de rutas paralelas en tiempo de desarrollo, esto puede lograrse fácilmente mediante una división. Sin embargo, en algunos casos el número de rutas solo se conoce en tiempo de ejecución. Tales situaciones pueden modelarse mediante actividades paralelas.

Una actividad paralela actúa como un envoltorio alrededor de actividades existentes. El efecto de utilizar una de estas nuevas actividades en tiempo de ejecución es que se ejecutan en paralelo varias instancias de la actividad envuelta. Hasta la fecha, los únicos tipos soportados de actividad envuelta son las actividades Manual (Capítulo 9, “Manual”, en la página 61) y Decisión (Capítulo 10, “Decisión”, en la página 77). Por tanto, la ejecución de una actividad paralela equivale actualmente a la creación y asignación de varias tareas en paralelo.

13.3 Metadatos

Una actividad paralela debe especificar el tipo de actividad que envuelve. Un objeto de datos de flujo de trabajo de lista también debe estar asociado a la actividad paralela. El número de elementos en este objeto de datos de flujo de trabajo de lista determinará el número de instancias de dicha actividad envuelta que creará el motor de flujo de trabajo en tiempo de ejecución.

13.3.1 Metadatos genéricos de una actividad paralela

```
<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>


    o bien........

    <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

Refleja el tipo de actividad envuelta por la actividad paralela. Actualmente se soportan dos tipos de actividades envueltas, las actividades Capítulo 9, “Manual”, en la página 61 y Capítulo 10, “Decisión”, en la página 77. Los tipos de actividad que pueden envolverse en una actividad paralela pueden consultarse en la tabla `ParallelActivityType` .

list-wdo-name

Cada actividad paralela debe tener un objeto de datos de flujo de trabajo de lista asociado. El número de instancias de la actividad envuelta creadas en tiempo de ejecución está determinado por el número de elementos en este objeto de datos de flujo de trabajo de lista.

13.3.2 Metadatos de una actividad paralela manual

El ejemplo siguiente ilustra los metadatos asociados a la actividad envuelta de tipo `Manual`. Estos metadatos son *exactamente* los mismos que los vistos para una actividad manual y que se describen en Capítulo 9, “Manual”, en la página 61, por lo tanto no se describirán aquí de nuevo. Las validaciones correspondientes a las correlaciones de actividad manual paralela también se describen en Capítulo 9, “Manual”, en la página 61. El objeto de datos de flujo de trabajo `Context_Parallel` y un elemento indexado procedente del WDO de lista de actividad paralela se pueden utilizar en todas las correlaciones disponibles de una actividad manual paralela. A continuación pueden verse ejemplos de este uso:

```
<parallel-activity id="1" category="AC1">
      <list-wdo-name>EmployerDetailsListWDO</list-wdo-name>
      <manual-activity>
        ...
        <task>
          <message>
            <message-text>
              <localized-text>
                <locale language="es">Comprobar detalles
                de empleador para %1s. Este es el empleador número: %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 Metadatos de una actividad paralela de decisión

El ejemplo siguiente ilustra los metadatos asociados a la actividad envuelta de tipo `Decision`. Estos metadatos son *exactamente* los mismos que los vistos para una actividad de decisión y que se describen en Capítulo 10, “Decisión”, en la página 77, por lo tanto no se describirán aquí de nuevo. Las validaciones correspondientes a las correlaciones de actividad de decisión paralela también se describen en Capítulo 10, “Decisión”, en la página 77. El objeto de datos de flujo de trabajo `Context_Parallel` y un elemento indexado procedente del WDO de lista de actividad paralela se pueden utilizar en todas las correlaciones disponibles de una actividad de decisión paralela. A continuación pueden verse ejemplos de este uso:

```
<parallel-activity id="1" category="AC1">
      <list-wdo-name>ChildDetailsListWDO</list-wdo-name>
      <decision-activity>
        ...
        <message>
          <message-text>
            <localized-text>
              <locale language="es">En esta tarea los detalles
              del hijo %1s debe validarse. Este es el hijo
              número: %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="es">Validar los detalles de hijo
                para %1s asociados con este caso %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="es">¿Son los detalles de este
                hijo cuyo nombre es %1s y apellido es
                %2s correctos?</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="es">Sí</locale>
                </localized-text>
              </answer-text>
            </answer>
            <answer name="answerNo">
              <answer-text>
                <localized-text>
```

13.3.4 Validaciones

- Deberá especificarse un objeto de datos de flujo de trabajo para una actividad paralela. Dicho objeto deberá ser objeto de datos de flujo de trabajo de lista y deberá ser válido en el contexto de la definición de proceso de flujo de trabajo contenedora.
- Todas las demás validaciones relacionadas con las actividades paralelas se tratan en los capítulos que describen las actividades que una actividad paralela puede englobar (es decir, Capítulo 9, “Manual”, en la página 61 y Capítulo 10, “Decisión”, en la página 77).

13.3.5 Información de entorno de ejecución

El motor de flujo de trabajo carga los datos de instancia para el objeto de datos de flujo de trabajo de lista asociado a la actividad paralela. Por cada elemento del objeto de datos de flujo de trabajo de lista, se crea y ejecuta una instancia de la actividad englobada. Los detalles de lo que ocurre cuando se ejecutan dichas instancias de la actividad envuelta pueden encontrarse en los correspondientes capítulos que describen las actividades que puede englobar una actividad paralela (Capítulo 9, “Manual”, en la página 61 y Capítulo 10, “Decisión”, en la página 77).

En tiempo de ejecución, el motor de flujo de trabajo trata una actividad paralela como si fuera varias actividades contenidas en un bloque *paralelo (AND) (AND) de división/unión*. Se crea una instancia de actividad por elemento en el WDO de lista de actividad paralela (p.ej. si dicha lista contiene tres elementos, se crearán tres instancias de actividad).Esto garantiza que todas las instancias de actividad asociadas a la actividad paralela tengan que completarse antes de que la actividad paralela real se considere completa y pueda avanzar el flujo de trabajo.

A fin de resolver las correlaciones asociadas a una actividad paralela, se asocia cada instancia de la actividad englobada a un elemento del WDO de lista de actividad paralela. El elemento se indexa utilizando el objeto de datos de flujo de trabajo Context_Parallel (p.ej. ChildDetailsListWDO[Context_Parallel.occurrenceCount]).

13.3.6 Descripción de WDO de contexto

Cada instancia de actividad paralela está asociada a un elemento del WDO de lista de actividad paralela. Se accede a este elemento mediante el objeto de datos de flujo de trabajo Context_Parallel para indexar el WDO de lista de actividad paralela (p. ej. ChildDetailsListWDO[Context_Parallel.occurrenceCount]). Los elementos indexados podrán utilizarse entonces para correlacionar los datos de la forma habitual. En los ejemplos de metadatos mostrados anteriormente pueden verse correlaciones así (consulte 13.3.2, “Metadatos de una actividad paralela manual”, en la página 96 y 13.3.3, “Metadatos de una actividad paralela de decisión”, en la página 97. El único atributo disponible en este objeto de datos de flujo de trabajo es:

Context_Parallel.occurrenceCount

Cada instancia de actividad paralela está asociada a un elemento del WDO de lista de actividad paralela. El atributo `occurrenceCount` es el índice de ese elemento en el WDO de lista de la actividad paralela. Es de tipo INTEGER y es un índice basado en cero.

Capítulo 14. Notificaciones de actividad

14.1 Descripción general

El motor de flujo de trabajo es capaz de notificar a los usuarios interesados sobre el progreso de una instancia de proceso de flujo de trabajo. Básicamente, el motor de flujo de trabajo puede generar una notificación cuando una actividad se ejecuta si la notificación se ha especificado en los metadatos de la definición de proceso asociada. Una notificación se especifica para una actividad como metadatos adicionales que pueden adjuntarse a cualquier tipo de actividad.

Cuando el motor de flujo de trabajo ejecuta una actividad, comprueba si se ha configurado una notificación para dicha actividad. Si existe una, el motor de flujo de trabajo crea una notificación en la que se detalla que se ha realizado un determinado paso en el proceso de flujo de trabajo. La entrega de estas notificaciones al usuario se determina mediante el mecanismo de entrega de notificaciones configurado en la aplicación Cúram. Las notificaciones se pueden entregar mediante correos electrónicos, como alertas enviadas al la bandeja de entrada de un usuario o mediante ambos, correos y alertas.

14.2 Detalles de notificación

Una notificación no es más que información que se envía a un agente humano cuando se ejecuta un paso del proceso. Las notificaciones se manifiestan como alertas en la bandeja de entrada del usuario o como correos electrónicos. Los agentes a los que hay que enviar la notificación están determinados por la estrategia de asignación (consulte 14.3, “Estrategia de asignación de notificación”, en la página 105) especificada para la notificación. Los detalles mostrados al usuario en la alerta o en el correo electrónico se especifican como parte de los metadatos de notificación.

14.2.1 Metadatos

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

  ...

    <notification delivery-mechanism="DM1">
      <subject>
       <message>
          <message-text>
           <localized-text>
              <locale language="es">
              El número de caso %1n del demandante %2s se
              ha cerrado.
              </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="es">
              Este caso concierne a %1n y al demandante %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="es">
                Ver la tarea asociada al caso %1n .
                </locale>
              </localized-text>
            </message-text>
            <message-parameters>
              <wdo-attribute wdo-name="TaskCreateDetails"
                name="caseID" />
            </message-parameters>
          </message>
          <link-parameter name="childID">
            <wdo-attribute wdo-name="ChildDependents"
              name="childID" />
          </link-parameter>
          <multiple-occurring-action>
      <list-wdo-name>ChildDependents</list-wdo-name>
          </multiple-occurring-action>
        </action>
        <action page-id="viewCaseHome" principal-action="false">
          <message>
            <message-text>
              <localized-text>
                <locale language="es">
                Ver los detalles de caso de %1n.
                </locale>
```

delivery-mechanism
Describe el mecanismo utilizado para entregar la notificación. Los mecanismos de entrega disponibles se especifican en tabla de códigos `DeliveryMechanism` de la aplicación. Tanto la aplicación de Cúram como los clientes pueden ampliar esta tabla y añadir mecanismos de entrega adicionales si fuera necesario. El mecanismo de entrega especificado no desempeña ningún rol funcional en el motor de flujo de trabajo, ya que se limita a llamar al mecanismo de entrega configurado en la aplicación para entregar la notificación recién creada.

subject
Representa un mensaje de texto parametrizado en el que se detalla el asunto de la notificación para todos los entornos locales. Dicho asunto se mostrará en la bandeja de entrada del usuario para la alerta de notificación. Puede obtener detalles sobre los mensajes parametrizados consultando Capítulo 9, “Manual”, en la página 61.

body Representa un mensaje de texto parametrizado que representa el cuerpo del texto asociado a esta notificación para todos los entornos locales. Cuando el usuario pulsa en el asunto de la notificación en la bandeja de entrada, este texto del cuerpo se mostrará como el texto completo de la notificación.

allocation-strategy
Representa la estrategia de asignación utilizada para determinar los agentes a los que se va a enviar esta notificación (consulte 14.3, “Estrategia de asignación de notificación”, en la página 105).

actions
De la misma manera que una Capítulo 9, “Manual”, en la página 61 puede tener acciones asociadas a su tarea, una notificación puede tener acciones asociadas que el usuario notificado puede llevar a cabo. Estos metadatos representan los detalles de dichas acciones de notificación y los detalles de los metadatos de las acciones detalladas en 9.3, “Detalles de tarea”, en la página 61.

multiple-occurring-action
Indica que esta acción de notificación representa una acción que ocurre varias veces. Esto significa que si se especifican estos metadatos para una acción de notificación, el motor de flujo de trabajo creará un registro de acción por cada elemento del objeto de datos de flujo de trabajo de lista especificado como acción con varias ocurrencias cuando se ejecute dicha actividad.

Debe tenerse en cuenta que cuando la acción que se produce varias veces se especifica para una acción de notificación, a continuación debe utilizarse un atributo del objeto de datos de flujo de trabajo asociado como parámetro de enlace para la acción de notificación.

list-wdo-name
Nombre del objeto de datos de flujo de trabajo de lista para su uso con la acción de ocurrencia múltiple.

14.2.2 Validaciones

- Deberá definirse un asunto para la notificación.
- Cada atributo de objeto de datos de flujo de trabajo correlacionado con un asunto de notificación deberá existir en la definición de proceso contenedora y ser un objeto de datos de flujo de trabajo válido.
- Si un elemento indexado de un objeto de datos de flujo trabajo de lista (es decir, CaseList[Context_Loop.loopCount]) se utiliza como parámetro de texto del asunto de la notificación, el objeto de datos de flujo de trabajo deberá ser un objeto de datos de flujo de trabajo de lista y la actividad que contiene la correlación deberá estar contenida en un bucle.
- Si el objeto de datos de flujo de trabajo `Context_Parallel` se utiliza como parámetro del texto de asunto de la notificación, la actividad que contenga la notificación deberá ser una actividad `paralela`.
- Si un elemento indexado procedente del de objeto de datos de flujo de trabajo de lista Parallel se utiliza como parámetro del texto de asunto de la notificación, la actividad que contenga la correlación

deberá ser una actividad paralela (es decir, ParallelListWDO[Context_Parallel.occurrenceCount]). El objeto de datos de flujo de trabajo indexado por el el objeto objeto de datos de flujo de trabajo Context_Parallel deberá ser el objeto de datos de flujo de trabajo de lista de la actividad paralela.

- Deberá definirse un cuerpo de notificación.
- Cada atributo de objeto de datos de flujo de trabajo correlacionado con un cuerpo de notificación deberá existir en la definición de proceso contenedora y ser un objeto de datos de flujo de trabajo válido.
- Si un elemento indexado de un objeto de datos de flujo trabajo de lista (es decir, CaseList[Context_Loop.loopCount]) se utiliza como parámetro de texto del cuerpo de la notificación, el objeto de datos de flujo de trabajo deberá ser un objeto de datos de flujo de trabajo de lista y la actividad que contiene la correlación deberá estar contenida en un bucle.
- Si el objeto de datos de flujo de trabajo `Context_Parallel` se utiliza como parámetro del texto del cuerpo de la notificación, la actividad que contenga la notificación deberá ser una actividad `paralela`.
- Si un elemento indexado procedente del de objeto de datos de flujo de trabajo de lista Parallel se utiliza como parámetro del texto del cuerpo de la notificación, la actividad que contenga la correlación deberá ser una actividad paralela (es decir, ParallelListWDO[Context_Parallel.occurrenceCount]). El objeto de datos de flujo de trabajo indexado por el el objeto objeto de datos de flujo de trabajo Context_Parallel deberá ser el objeto de datos de flujo de trabajo de lista de la actividad paralela.
- Deberá definirse una estrategia de asignación para una notificación de actividad.
- Si se especifica una función como estrategia de asignación de notificaciones, debe ser un método de negocio de Cúram válido que devuelva un objeto `AllocationTargetList`.
- Un conjunto de reglas especificado como estrategia de asignación de notificaciones deberá ser un conjunto de reglas de asignaciones válido.
- Deberá especificarse un mecanismo de entrega para una notificación de actividad.
- Los atributos de objeto de datos de flujo de trabajo correlacionados con los parámetros de texto de acción de notificación y de enlace de acción de notificación para una acción de notificación deberán existir en la definición de proceso contenedora.
- Si un elemento indexado de un objeto de datos de flujo trabajo de lista (es decir, PersonDetailsList[Context_Loop.loopCount]) se utiliza como correlación de parámetro de texto de acción de notificación o de enlace de acción de notificación, el objeto de datos de flujo de trabajo deberá ser un objeto de datos de flujo de trabajo de lista y la actividad que contiene la correlación deberá estar contenida en un bucle.
- Si el objeto de datos de flujo de trabajo `Context_Parallel` se utiliza como correlación de parámetro de texto de acción de notificación o de enlace de acción de notificación, la actividad que contenga la notificación deberá ser una actividad `paralela`.
- Si un elemento indexado procedente del de objeto de datos de flujo de trabajo de lista Parallel se utiliza como correlación de parámetro de texto de acción de notificación o de enlace de acción de notificación, la actividad que contenga la correlación deberá ser una actividad paralela (es decir, ParallelListWDO[Context_Parallel.occurrenceCount]). El objeto de datos de flujo de trabajo indexado por el el objeto objeto de datos de flujo de trabajo Context_Parallel deberá ser el objeto de datos de flujo de trabajo de lista de la actividad paralela.
- El número de marcadores de posición utilizado en el texto del asunto de la notificación, el texto de la acción de notificación y el texto del cuerpo de la notificación deberá ser igual al número de atributos de objeto de datos de flujo de correlacionados (para todos los entornos locales).
- El objeto de datos de flujo de trabajo especificado para su uso en la acción de ocurrencia múltiple deberá ser un objeto de datos de flujo de trabajo válido en el contexto de la definición de proceso de flujo de trabajo contenedora. También deberá ser de tipo List.
- Al menos un atributo del objeto de datos de flujo de trabajo de lista de acción de ocurrencia múltiple deberá utilizarse en los parámetros de enlace especificados para una acción de ocurrencia múltiple.

14.2.3 Código

Por cada acción definida, la página de acción deberá hacer referencia a una página Cúram válida en la aplicación cuyos parámetros de página sean rellenados completamente por los parámetros de enlace de acción contenidos en los metadatos de notificación.

A la aplicación se le proporciona un API `LocalizableStringResolver` que permite resolver las cadenas de mensaje parametrizadas. Los métodos de esta API resolverán y devolverán el mensaje especificado para los entornos locales necesarios. Junto a esto, se resolverán los objetos de datos de flujo de trabajo a utilizar en los marcadores de posición del mensaje y se incluirán como parte de la cadena devuelta.

Como parte del API `LocalizableStringResolver`, se proporciona una interfaz `NotificationStringResolver` para resolver los mensajes parametrizados asociados a las notificaciones. El asunto de la notificación, el cuerpo y el texto de la acción pueden resolverse para su uso en la aplicación utilizando los métodos contenidos en esta API. La aplicación debe utilizar estos métodos para procesar la notificación una vez que el motor de flujo de trabajo haya invocado el método de entrega de notificación asociado en la aplicación.

14.2.4 Información de entorno de ejecución

Una vez que el motor de flujo de trabajo ha completado la ejecución de una actividad, comprueba si se ha definido una notificación asociada para dicha actividad. Si hay una definida, el motor determina los usuarios que hay que notificar a partir de la estrategia de asignación empleada y llama al método de entrega de notificaciones de la aplicación con los detalles de notificación.

14.3 Estrategia de asignación de notificación

14.3.1 Requisitos previos

La estrategia de asignación de notificaciones determina el usuario o usuarios que se van a notificar una vez que haya tenido lugar la actividad asociada. La definición de la estrategia de asignación de notificaciones a utilizar es exactamente igual que la que se utiliza para las tareas de actividad manual (consulte 9.4, “Estrategia de asignación”, en la página 67).

14.3.2 Código

La aplicación debe implementar la interfaz de de devolución de llamada `NotificationDelivery` para determinar cómo se manejan las notificaciones en la aplicación.

El motor de flujo de trabajo invocará el método `deliverNotification` de la clase de implementación `curam.util.workflow.impl.NotificationDelivery` para procesar la notificación. El motor pasará a este método de aplicación la lista de destinos de asignación determinada por la estrategia de asignación y los detalles de la notificación necesaria.

La propiedad de aplicación curam.custom.notifications.notificationdelivery define qué implementación de la interfaz `NotificationDelivery` utilizará el motor de flujo de trabajo para procesar la notificación.

El método `deliverNotification` de esta clase de implementación está sobrecargado. Esto se debe a que los diferentes tipos de estrategia de asignación devuelven los destinos de asignación en formatos distintos. No obstante, esto es un detalle de implementación del que no tienen que preocuparse los desarrolladores de clases personalizadas de entrega de notificaciones, sobre todo porque el procesamiento de negocio de todas las versiones del método debería ser el mismo.

```
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);

            ...
      }
```

Para paliar este problema, `curam.core.sl.impl.DefaultNotificationDeliveryAdapter` proporciona un mecanismo más cómodo de implementar un programa de resolución de trabajo. Esta clase implementa los distintos métodos y convierte sus parámetros de entrada en listas de destino de asignación, permitiendo a los desarrolladores de lógica personalizada de entrega de notificaciones extender esta clase e implementar un método que se invoca independientemente del origen de los destinos de asignación.

```
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);

            ...
      }
```

Además de esta clase de adaptador, la aplicación se distribuye con una implementación de entrega de notificaciones que ya va lista para utilizarse. Dicha clase se denomina `curam.core.sl.impl.DefaultNotificationDelivery` y también sirve de ejemplo de cómo extender el adaptador.

Las estrategias de entrega de notificaciones se listan en la tabla de códigos DELIVERYMECHANISM. La adición de una nueva estrategia consiste simplemente en extender esta tabla de códigos con una nueva estrategia (por ejemplo, SMS) e implementar una estrategia de entrega que reconozca este código y lleve a cabo la lógica adecuada. Sin embargo, y puesto que la clase de entrega de notificaciones se establece mediante una única propiedad de aplicación, la sustitución de la clase `curam.core.sl.impl.DefaultNotificationDelivery` inhabilitaría los mecanismos de entrega que ya van preinstalados. Si el objetivo es ampliar, y no sustituir, los mecanismos de entrega preinstalados, las clases personalizadas deberían extender `curam.core.sl.impl.DefaultNotificationDelivery` de modo que se conserve la funcionalidad original. La clase `curam.core.sl.impl.DefaultNotificationDelivery` se ha implementado teniendo esto en cuenta.

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

 public class DefaultNotificationDelivery
     extends DefaultNotificationDeliveryAdapter {

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

   protected boolean selectDeliveryMechanism(
     NotificationDetails notificationDetails,
      AllocationTargetList allocationTargetList) {

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

   ...

 }
```

La clase `curam.core.sl.impl.DefaultNotificationDelivery` implementa el método `deliverNotification` del adaptador abstracto, pero delega de forma inmediata la identificación del mecanismo a utilizar a un método protegido. Las subclases pueden sobrescribir el método protegido `selectDeliveryMechanism` para identificar los mecanismos de entrega personalizados y llevar a cabo las operaciones adecuadas tal y como se muestra en el ejemplo siguiente:

```
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);
   }
 }
```

Tenga en cuenta que el método `selectDeliveryMechanism` de la clase personalizada delega en primer lugar en la superclase antes de ejecutar su propia lógica. Al extenderse la funcionalidad de esta forma, se permite que las clases personalizadas invoquen el mecanismo de entrega preinstalado sin tener que conocer los códigos específicos que reconoce la clase padre. Este enfoque también facilita las actualizaciones, ya que si una versión futura de Cúram soporta más mecanismos de entrega preinstalados, no tendrá que cambiarse una clase personalizada implementada como se muestra aquí para valerse de la nueva funcionalidad.

El distintivo booleano que devuelve la función de entrega anterior se utiliza para indicar al motor de flujo de trabajo si la notificación se ha entregado a al menos un usuario del sistema. Si no ha sido así, el motor entregará un registro de auditoría de flujo de trabajo en el que se detalle este hecho.

Capítulo 15. Transiciones

15.1 Descripción general

Las transiciones proporcionan los enlaces entre actividades. Son la construcción de flujo principal y dictan el orden en el que se ejecutarán las actividades. Las transiciones son unidireccionales y una actividad puede tener varias transiciones de salida y de entrada que forman puntos de rama y de sincronización respectivamente. Puesto que cada definición de proceso debe tener una actividad de inicio y una de final (consulte Capítulo 6, “Actividad base”, en la página 31), una definición de proceso puede considerarse informalmente como un grafo dirigido en el que las actividades son los vértices, las transiciones son los arcos y todas las rutas que comienzan en la actividad de inicio conducen a la actividad final.

15.2 Metadatos

```
<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

Una definición de proceso de flujo de trabajo debe contener al menos una transición. Este elemento contiene los detalles de todas las transiciones entre las actividades en la definición de proceso de flujo de trabajo especificada.

transition

Contiene los detalles de una transición entre dos actividades en la definición de proceso de flujo especificada. Los campos obligatorios que constituyen una transición se describen a continuación:

id Es un identificador de 64 bits proporcionado por el servidor de claves de Cúram cuando se crean transiciones en la herramienta de definición de procesos (PDT). El identificador de transición debe ser exclusivo dentro de la definición de un proceso, pero no es necesaria la exclusividad global dentro de todas las definiciones de proceso del sistema.

from-activity-idref
Identificador de 64 bits de la actividad de origen de la transición.

to-activity-idref
Identificador de 64 bits de la actividad de destino de la transición.

condition
De forma opcional, las transiciones pueden tener una condición que decida si la transición se va a seguir o no. Una condición es una lista de expresiones que realizan operaciones lógicas sobre atributos de objetos de datos de flujo de trabajo. Las condiciones se describen con más detalle en Capítulo 16, “Condiciones”, en la página 113

15.3 Validaciones

- La actividad de origen definida para la transición deberá ser una actividad válida dentro de la definición de proceso de flujo de trabajo contenedora.
- La actividad de destino definida para la transición deberá ser una actividad válida dentro de la definición de proceso de flujo de trabajo contenedora.
- Las actividades de origen y destino definidas para una transición no podrán ser la misma actividad.
- La actividad de inicio de proceso en una definición de proceso de flujo de trabajo no deberá contener transiciones entrantes.
- La actividad de fin de proceso en una definición de proceso de flujo de trabajo no deberá contener transiciones salientes.
- Todas las actividades definidas en la definición de proceso de flujo de trabajo, excepto la actividad de fin de proceso, deberán contener al menos una transición de entrada.
- Todas las actividades definidas en la definición de proceso de flujo de trabajo, excepto la actividad de inicio de proceso, deberán contener al menos una transición de salida.

15.4 Información de entorno de ejecución

Las actividades que realizan trabajo relacionado con la aplicación (a diferencia del trabajo exclusivo del motor de flujo de trabajo, como actividades de finalización de proceso y de ruta) requieren un límite transaccional claro entre los códigos del motor y de la aplicación. También resultan útiles las invocaciones asíncronas entre el motor de flujo de trabajo y la aplicación (p.ej. un usuario no debería tener que esperar mientras el flujo de trabajo transiciona a la siguiente actividad antes de que se le devuelva el control en la interfaz de usuario).

Con este fin, hay tres funciones diferenciadas en una actividad de flujo: `start()`, `execute()` y `complete()`. Tras la finalización de una actividad en la instancia de proceso de flujo de trabajo, el motor de flujo de trabajo llama a la función para continuar con el proceso. Esta función evalúa las transiciones salientes de la actividad para determinar cuál(es) se va(n) a seguir.

Por cada actividad a seguir, se llama la correspondiente función `start()`. Luego se configuran los datos de instancia de actividad adecuados para esa actividad. Si la actividad se va a ejecutar directamente sin necesidad de mensajería JMS (el API de Java Message Service (JMS) forma parte de Java EE), es decir, siempre se ejecuta una actividad de ruta directamente al no haber trabajo relacionado con la aplicación, entonces en este punto se invoca el método `execute()`. De lo contrario, se envía un mensaje JMS al ejecutar la actividad especificada (es decir, una actividad automática). El manejador de mensajes de flujo de trabajo resuelve el proceso y la actividad especificados en el mensaje y llama a la función `execute()` de la actividad.

Después de llamar al código de aplicación para llevar a cabo el trabajo especificado por la actividad, se envía otro mensaje para completar la actividad. De nuevo, el manejador de mensajes de flujo de trabajo resuelve el proceso y la actividad especificados en el mensaje y llama a la función `complete()` de la

actividad. Tras marcar la actividad como completada, se vuelve a llamar la función para continuar con el proceso para resolver el conjunto de transiciones a seguir desde la actividad completada y el proceso empieza de nuevo.

Capítulo 16. Condiciones

16.1 Descripción general

Las construcciones de control de flujo descritas en Capítulo 15, “Transiciones”, en la página 109 y Capítulo 12, “Inicio de bucle y fin de bucle”, en la página 91 requieren o soportan la evaluación de condiciones para determinar cómo debe seguir el flujo de trabajo. Las actividades de inicio de bucle deben tener algún metadato que especifique las condiciones de salida del bucle, mientras que las transiciones pueden tener opcionalmente una condición para decidir si se sigue una determinada transición.

En este capítulo se describe la construcción de metadatos de definición de proceso que representa una condición. Una condición es una lista de expresiones que realizan operaciones lógicas sobre atributos de objetos de datos de flujo de trabajo. La propia condición es un compuesto cuyo valor es la conjunción o disyunción de sus expresiones constituyentes. Las construcciones padre (bucles y transiciones) son responsables de llevar a cabo las acciones adecuadas como resultado de la evaluación de las condiciones.

16.2 Metadatos

```
<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

Este metadato es obligatorio para una actividad de inicio de bucle (ya que un bucle debe tener una condición de salida especificada), pero opcional para una transición (una transición puede no tener una condición especificada). Contiene los detalles de todas las expresiones definidas para la condición.

expression

Contiene los detalles de una expresión contenida en una condición. Puede haber una o varias expresiones especificadas para una condición asociada. Pueden definirse dos tipos de expresión en una condición. Son las expresiones de función (utilizando una de las dos funciones predefinidas, `not()` y `isNothing()`) y las expresiones de elemento de datos (donde la expresión de condición creada aplica el operador elegido a dos atributos de objeto de datos de flujo de trabajo, o bien a un atributo de objeto de datos de flujo de trabajo y a una constante). Una expresión de transición consta de los atributos siguientes:

id
Representa un identificador de `64 bits` proporcionado por el servidor de claves de Cúram cuando se crean expresiones de transición en la PDT. El identificador de expresión debe ser exclusivo dentro de la definición de un proceso, pero no es necesaria la exclusividad global dentro de todas las definiciones de proceso del sistema.

data-item-rhs
Representa el nombre del elemento de datos que se utiliza en el lado derecho de la expresión de condición. En el caso de una expresión de condición del elemento de datos, puede representar un atributo de objeto de datos de flujo de trabajo (consulte Capítulo 4, "Objetos de datos de flujo de trabajo", en la página 17) o un valor constante al que se va a aplicar el operador seleccionado. Para las expresiones de condición de función, esto representa un atributo de objeto de datos de flujo de trabajo contra el que se utilizará cualquiera de las dos funciones predefinidas para evaluar la condición.

data-item-lhs
Esta etiqueta de metadatos es opcional, porque no es necesaria para una expresión de condición de función. En el caso de una expresión de condición de elemento de datos, representa el nombre del elemento de datos a utilizar en el lado izquierdo de la condición (es decir, un atributo de objeto de datos de flujo de trabajo).

operation
Esta etiqueta de metadatos es opcional, porque no es necesaria para una expresión de condición de función. En el caso de una expresión de condición de elemento de de datos, representa un identificador para la operación lógica que se aplica a dos atributos de objeto de datos de flujo de trabajo, o bien a un atributo de objeto de datos de flujo de trabajo y a un valor constante. Lo siguiente es la lista de operadores válidos que se pueden utilizar en una expresión de condición de elemento de datos:

Tabla 4. Operadores de expresiones de condición

Operador	Explicación
==	igual a
!=	distinto de
<=	menor o igual que
>=	mayor o igual que
<	menor que
>	mayor que

conjunction
Representa un identificador para una conjunción lógica que se puede utilizar en una expresión de condición de función o de elemento de datos. Hay dos valores posibles para este atributo, `and` ('y', el valor predeterminado) y `or` ('o'). Cuando una condición consta de varias expresiones, se utiliza la conjunción lógica en la evaluación de la condición completa.

function
Es opcional, porque sólo se utiliza cuando se especifica una expresión de condición de función. Como se ha indicado anteriormente, existen dos funciones predefinidas, `Not()` y

`isNothing()`. La función `Not()` actúa como un operador de inversión lógica. En casos normales, se aplica a un valor booleano. La función `isNothing()` puede aplicarse a cualquier tipo de atributo de objeto de datos de flujo de trabajo que no sea un valor booleano. Puede utilizarse para probar los escenarios donde los datos necesarios no existen o no se han proporcionado. La función devuelve un valor booleano `True` si el atributo de objeto de datos de flujo de trabajo que se examina no contiene ningún dato.

opening-brackets
Es opcional (el valor predeterminado es 0), ya que puede que no se especifique para ninguno de los tipos de expresión de condición. Representa el número de paréntesis abiertos a insertar al principio de la expresión.

closing-brackets
Es opcional (el valor predeterminado es 0), ya que puede que no se especifique para ninguno de los tipos de expresión de condición. Representa el número de paréntesis cerrados a insertar al final de la expresión.

Los números de paréntesis abiertos y cerrados especificados en una expresión individual no tienen por qué coincidir (a menos que sólo haya una expresión en la condición). Los números de paréntesis abiertos y cerrados en la condición global (incluidas todas las expresiones) deben ser iguales. Por tanto, se deberá tener cuidado al especificar el número y la posición de los paréntesis abiertos y cerrados dentro de una expresión individual y en la condición en conjunto, ya que estos paréntesis ayudan a determinar cómo se evalúan la condición y las expresiones individuales dentro de dicha condición. También se deberá prestar atención al especificar la conjunción de una expresión, porque si se hace incorrectamente podrán generarse resultados inesperados.

16.3 Validaciones

- El atributo de objeto de datos de flujo de trabajo especificado como elemento de datos del lado derecho de la expresión de condición debe ser un atributo de objeto de datos de flujo de trabajo válido en el contexto de la definición de proceso de flujo de trabajo contenedora.
- El atributo de objeto de datos de flujo de trabajo especificado como elemento de datos del lado izquierdo de la expresión de condición debe ser un atributo de objeto de datos de flujo de trabajo válido en el contexto de la definición de proceso de flujo de trabajo contenedora.
- El operador especificado en una expresión de condición de elemento de datos debe ser un operador válido y soportado.
- La función especificada en una expresión de condición de función debe ser una función válida y soportada.
- La conjunción especificada en una expresión de condición debe ser una conjunción válida y soportada.
- El número de paréntesis abiertos y el número de paréntesis cerrados deben ser iguales en el contexto de la condición global.
- Si la función `Not()` se especifica en una expresión de condición de función, el tipo de atributo de objeto de datos de flujo trabajo especificado como elemento de datos del lado derecho de la expresión debe ser de tipo BOOLEAN.
- Si la función `isNothing()` se especifica en una expresión de condición de función, el tipo de atributo de objeto de datos de flujo trabajo especificado como elemento de datos del lado derecho de la expresión no debe ser de tipo BOOLEAN.
- Si el elemento de datos del lado derecho de una expresión de condición de elemento de datos es un atributo de objeto de datos de flujo de trabajo, el tipo de dicho atributo deberá ser compatible con el correspondiente atributo de objeto de datos de flujo de trabajo de elemento de datos del lado izquierdo. De igual forma, si el elemento de datos del lado derecho se ha especificado como valor constante, deberá ser compatible con el tipo del correspondiente atributo del objeto de datos de flujo de trabajo de elemento de datos del lado izquierdo.

- Si el lado izquierdo o el derecho de una expresión de condición de transición contiene un elemento indexado de un objeto de datos de flujo de trabajo de lista (es decir, ChildDependents[Context_Loop.loopCount].age), el objeto de datos de flujo de trabajo asociado deberá ser un objeto de datos de flujo de trabajo de lista y las actividades implicadas en la transición deberán estar contenidas en un bucle.
- En el caso de una expresión de condición de bucle, si el lado derecho o el izquierdo de la expresión especifica el atributo *size()* para un objeto de datos de flujo de trabajo, dicho objeto de datos de flujo de trabajo deberá ser un objeto de datos de flujo de trabajo de lista.
- En el caso de una expresión de condición de bucle, si el lado derecho o el izquierdo de la expresión especifica el atributo *size()* para un objeto de datos de flujo de trabajo, el elemento del otro lado de la expresión deberá poder asignarse al tipo INTEGER.
- En el caso de una expresión de condición de bucle, si el lado derecho o el izquierdo de la expresión especifica el atributo *isEmpty()* para un objeto de datos de flujo de trabajo, dicho objeto de datos de flujo de trabajo deberá ser un objeto de datos de flujo de trabajo de lista.
- En el caso de una expresión de condición de bucle, si el lado derecho o el izquierdo de la expresión especifica el atributo *isEmpty()* para un objeto de datos de flujo de trabajo, el elemento del otro lado de la expresión deberá poder asignarse al tipo BOOLEAN.

Capítulo 17. División/Unión

17.1 Introducción

Las transiciones vinculan actividades en una definición de proceso. En la configuración más básica de actividades y transiciones, cada actividad sólo tiene una transición entrante y una saliente. Sin embargo, con frecuencia es útil seguir más de una ruta al salir de una actividad, lo que da lugar a una división (es decir, varias transiciones que parten de una actividad). Para soportar una estructura de bloques válida en una definición de proceso (consulte Capítulo 18, "Estructura del flujo de trabajo", en la página 121), a cada división debe corresponderle una unión (es decir, varias transiciones que convergen en una actividad). En general, una división permite que se ejecuten varias hebras de trabajo a la vez, mientras que una unión es el punto de sincronización recíproca de dichas hebras.

Hay dos motivos por los que una actividad puede tener una división (y, por extensión, que alguna otra actividad más abajo en la línea pueda tener una unión). El primero es permitir que un trabajo que no tenga dependencias pueda realizarse en paralelo, mientras que el segundo es permitir elegir entre diversas rutas del flujo de trabajo.

A nivel de metadatos, cada actividad tiene un tipo de división y otro de unión. Cuando una actividad solo tiene una transición saliente o entrante, se asignará un tipo `none` (ninguna) a la división o a la unión respectivamente. Los otros dos tipos de división y unión, `selección` (también conocida como XOR) y `paralela` (también conocida como AND), se explican por sí mismos y son el tema principal de este capítulo.

17.2 División XOR de selección

17.2.1 Metadatos

```
<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 Está presente en cada actividad y contiene los detalles de la división de la actividad. Incluye una lista de las transiciones de la actividad especificada que resolverá el motor de flujo de trabajo cuando la actividad asociada se complete para examinar si se pueden seguir o no.

El orden de las transiciones en esta lista es importante para un tipo de división de `XOR`, porque la primera transición elegible en la lista ordenada de transiciones es la que seguirá el motor de flujo de trabajo. En el ejemplo metadatos anterior, si se cumplen las condiciones de transición para los

identificadores de transición 2, 3 y 4, será la transición con el identificador 2 la que se siga, ya que es la primera transición elegible en la lista de transiciones ordenadas.

type Esto representa el tipo de la división. Como se ha descrito anteriormente, existen tres tipos posibles de división. Un tipo de división `none` indica que solo hay una transición de salida desde la actividad especificada. Un tipo de división `xor` indica una selección, y esto significa que se seguirá la primera transición elegible de la lista de transiciones ordenadas. Un tipo de división `and` indica una ruta paralela de ejecución que garantiza que se sigan todas las transiciones elegibles en la lista ordenada de transiciones.

transition-id
Contiene una referencia a la transición especificada. Cuando el tipo de división sea `xor` o `and`, habrá varias entradas de esta etiqueta de metadatos.

idref Contiene una referencia a una transición en la definición de proceso de flujo de trabajo.

17.3 División AND paralela

17.3.1 Metadatos

```
<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>
```

Los metadatos del tipo de división `and` son similares a los del tipo de división `xor` (consulte 17.2, “División XOR de selección”, en la página 119). La diferencia es que el tipo de división se especifica como `and`. Esto garantiza que, cuando el motor de flujo de trabajo determina la lista de transiciones a seguir desde una actividad determinada, el orden de las transiciones en esta lista no sea relevante, porque en una división `and` se siguen todas las transiciones elegibles. La lista ordenada de transiciones se mantiene en la instancia de este tipo de división para facilitar el cambio del tipo de división de `and` a `xor`, en cuyo caso el orden de las transiciones volvería a ser relevante.

Capítulo 18. Estructura del flujo de trabajo

18.1 Descripción general

La estructura de un proceso de flujo de viene determinada por las actividades en el proceso y las transiciones entre ellas. Por tanto, un flujo forma un grafo en el que las actividades son nodos y las transiciones son arcos (el grafo formado por un flujo de trabajo puede verse utilizando la característica *Visualizar proceso de flujo de trabajo* en la herramienta de definición de procesos).

Para que el motor de flujo de trabajo para interpretar y ejecutar correctamente un proceso, el grafo formado por dicho proceso deberá ajustarse a determinados criterios. Este capítulo presenta dichos criterios bajo dos títulos principales: Estructura de grafo y Estructura de bloques.

18.2 Estructura de grafo

Puesto que el conjunto de actividades y transiciones de un proceso forman un grafo, puede aplicarse la teoría de grafos para detectar diversos problemas bien conocidos antes de que llegue a ejecutarse un proceso.

Teoría de grafos: La teoría de grafos es una rama de las matemáticas. Afortunadamente, las partes de la teoría de grafos que son relevantes para un flujo de trabajo son muy simples. Por lo tanto, este capítulo no requiere ningún conocimiento previo de teoría de grafos (ni mucho menos una licenciatura en matemáticas). En Internet hay abundante información sobre la teoría de grafos en la que puede encontrarse un análisis más detallado de muchos de los asuntos tratados en este capítulo.

Por ejemplo: considere un proceso en el que una actividad tiene una transición a otra actividad, que a su vez tiene una transición que vuelve a la primera actividad. Esto forma un ciclo en el grafo del proceso.

Si no hubiera condiciones en las transiciones, el proceso acabaría con seguridad en un bucle infinito. Dichos bucles se conocen como bucles informales (o bucles 'ad hoc') y su presencia hace que sean imposibles determinadas validaciones estructurales de utilidad. Por esta razón (entre otras), el flujo de trabajo de Cúram proporciona construcciones formales para delimitar las secciones iterativas de un proceso (las actividades de inicio y finalización de bucle). Esto permite detectar la presencia de bucles ad hoc en los procesos e impide que tales procesos se publiquen.

Analogías de código: Muchos desarrolladores estará familiarizados con la sentencia GOTO de un lenguaje de programación y las llaves que se utilizan normalmente para delimitar el inicio ({) y el final (}) de un bucle formal.

GOTO es análogo a los bucles ad hoc en un flujo de trabajo. Las llaves son análogas a las actividades formales de inicio y finalización de bucle (loop-begin-activity y loop-end-activity) en un flujo de trabajo.

18.3 Estructura de bloques

Existen varios elementos de flujo de trabajo que pueden afectar a la selección de la ruta (o rutas) de flujo a través de un flujo de trabajo en tiempo de ejecución. Estos incluyen:

- 17.2, “División XOR de selección”, en la página 119
- 17.3, “División AND paralela”, en la página 120
- Capítulo 12, “Inicio de bucle y fin de bucle”, en la página 91

Estos elementos siempre van emparejados. Esto se debe a que delimitan áreas en las que el proceso debe mostrar un determinado comportamiento (relacionado con el flujo de control). Estas áreas suelen referirse como 'bloques' porque tienen un punto de inicio concreto que debe tener su correspondiente punto final.

Considere un proceso con una estructura en la que todas las rutas que salen de una división de selección (que garantiza que sólo se siga una ruta de salida) confluyen en una unión paralela (que esperará a que todas las rutas entrantes terminen antes de ejecutar la siguiente actividad). En tal caso, es seguro que el proceso se bloqueará en la unión paralela. Esto es un ejemplo de un problema en una estructura de bloques que las validaciones pueden detectar antes incluso de que se ejecute un proceso.

18.3.1 Una analogía para bloques

Una analogía habitual de cómo funcionan los "bloques" en un flujo de trabajo es el modo en que los paréntesis (como estos) funcionan en una oración. Los paréntesis tienen un comienzo explícito '(' que siempre va acompañado de un final específico ')'. Delimitan un área de la oración que tiene un significado concreto.

El modo en que los paréntesis operan en una expresión matemática es una analogía más cercana. Además de los correspondientes paréntesis abiertos y cerrados, una expresión matemática puede valerse de varios tipos de delimitadores. Las expresiones encerradas entre paréntesis pueden anidarse unas dentro de otras, pero no pueden solaparse. Esto es muy similar a la forma en que funcionan los bloques en un flujo de trabajo.

18.3.2 Tipos de bloque soportados por un flujo de trabajo

En los apartados siguientes se describen los diferentes tipos de bloques en el flujo de trabajo de Cúram, cómo comienzan/terminan y cuál es su finalidad.

18.3.2.1 Bloque 'selección' (XOR)

Un *bloque de selección* comienza en una división de selección (XOR) y termina en una unión de selección (XOR) (los 'paréntesis'). Indica que, de todas las vías de acceso posibles dentro del bloque, no puede seguirse más de *una*.

La división tiene varias transiciones de salida que indican las posibles rutas que una instancia de proceso puede seguir. Puesto que es un bloque de selección, las rutas son son mutuamente excluyentes: una determinada instancia de proceso solo se seguirá una.

La división de selección debe ir emparejada con correspondiente unión de selección. Dicha unión marca el punto en el que el proceso deja de ser distinto en cada ruta, de modo que las rutas confluyen de nuevo (es decir, el proceso que queda es común).

18.3.2.2 Bloque 'paralelo' (AND)

Un *bloque paralelo* comienza en una división paralela (AND) y termina en una unión paralela (AND) (los 'paréntesis'). Indica que, de todas las vías de acceso posibles dentro del bloque, pueden seguirse *muchas o todas*.

La división tiene varias transiciones de salida que indican las posibles rutas que una instancia de proceso puede seguir. Dado que este es un bloque paralelo, podrá seguirse cualquier número de rutas en paralelo (suponiendo que se cumplan sus condiciones de transición).

La división paralela debe ir emparejada con la correspondiente unión paralela. Esto indica el punto en el que las rutas paralelas deben sincronizarse antes de que pueda continuar el flujo de trabajo.

18.3.2.3 Bloque de 'bucle'

Un *bloque de bucle* empieza en una actividad de inicio de bucle y termina en una actividad de fin de bucle (los 'paréntesis'). Indica que la sección del flujo de trabajo delimitada por las actividades de inicio y fin de bucle se repite mientras la condición del bucle se cumpla.

La actividad de inicio de bucle marca el punto al que debe volver la ejecución si se cumple la condición del bucle (es decir, el lugar al que se vuelve si el motor determina que el bucle debe iterar). La actividad de fin de bucle marca el punto al que debe saltar la ejecución si *no* se cumple la condición del bucle.

18.4 Reglas estructurales

Existen determinadas reglas estructurales que los diseñadores de flujo de trabajo deben tener en cuenta al construir definiciones de proceso. Cuando se valida un proceso de flujo de trabajo de Cúram, la validaciones comprueban si la estructura del proceso se ajusta a estas reglas. Como en todas las validaciones, el objetivo es garantizar que el motor de flujo de trabajo pueda ejecutar el proceso.

18.4.1 Reglas de la estructura de grafo

Un proceso Cúram debe formar un grafo que tenga las propiedades siguientes: dirigido, conectado y acíclico. Puede que suene complicado, pero solo son términos técnicos que designan propiedades muy sencillas de los grafos.

- Un grafo "dirigido" es uno en el que cada arista solo tiene un sentido (suele conocerse como dígrafo). En términos de flujo de trabajo, esto significa que una transición de la actividad A a la actividad B no puede utilizarse para volver de B a A. Esto es así en un flujo de trabajo Cúram. Se menciona aquí solo porque la propiedad 'acíclico' (véase más abajo) se define de forma distinta para grafos y dígrafos.
- Un grafo "conectado" es aquel en el que pueden alcanzarse todos los nodos. En términos de flujo de trabajo, esto significa que cada actividad del proceso debe ser accesible en al menos una ruta desde la actividad inicial a la actividad final.

  Esto impide que los flujos de trabajo tengan una estructura en la que una o más actividades nunca puedan ejecutarse.
- Por último, un dígrafo "acíclico" es aquel en el que no hay ciclos *dirigidos*. En términos de flujo de trabajo, esto significa que no puede haber ningún bucle ad hoc (es decir, un bucle construido mediante transiciones en lugar de actividades de inicio y fin de actividad).

  Puede que los bucles ad hoc parezcan cómodos, pero (al igual que las sentencias GOTO de los lenguajes de programación) pueden hacer que un proceso sea muy difícil de leer y comprender. Las construcciones explícitas de bucle dan lugar a definiciones de proceso más claras y comprensibles.

  Además, permite al motor saber dónde pueden tener lugar los bucles, de modo que puede llevar cuenta de cuántas veces ha iterado un bucle en tiempo de ejecución.

18.4.2 Reglas de la estructura de bloques

Como se ha mencionado anteriormente, el modo en que los paréntesis funcionan en una expresión matemática es una analogía cercana a cómo funcionan los "bloques" en un flujo de trabajo. Recuerde, hay varios tipos de bloques: selección, paralelo y bucle. En el flujo de trabajo de Cúram:

- Todas las construcciones de inicio de bloque (división de selección, división paralela o actividad de inicio de bucle) deben terminarse con la correspondiente estructura de finalización de bloque (unión de selección, unión paralela o actividad de finalización de bucle).

  En el caso de divisiones y uniones, todas las rutas de salida de una división deben confluir en la unión correspondiente.

  Fundamento: La exigencia de que divisiones y uniones (por ejemplo) vayan emparejadas mejora la legibilidad. En una sección que contenga varias rutas, deja claro si puede seguirse una (o varias) rutas. Esto a su vez deja claro si es necesaria una sincronización o no en el punto donde confluyen las rutas.

  Si no se exigiese que fueran emparejadas, sería posible (fácil) modelar procesos que con seguridad se bloquearían, o procesos en los que podría alcanzarse el final del proceso antes de que algunas actividades hubieran terminado de ejecutar.
- Los bloques pueden anidarse unos dentro de otros (por ejemplo, una división de selección dentro de un bucle), pero no pueden intercalarse (p. ej. ninguna de las transiciones de una división de selección puede ir a una actividad fuera del bucle).

Esto ayuda a evitar situaciones difíciles de procesar para el motor, y resulta muy intuitivo para los desarrolladores de flujo de trabajo.

Considere un bucle que contenga una unión con dos transiciones entrantes: una procedente de una actividad dentro del bucle y la otra procedente de una actividad fuera del bucle.

En esta situación es muy difícil decidir cómo debería funcionar la sincronización de la unión. Puede que una transición de entrada solo se active una vez, mientras que la otra podría activarse varias veces. Cualquier regla que maneje tal situación sería arbitraria y, por tanto, poco intuitiva.

Los flujos de trabajo definidos mediante bloques de selección, paralelos y de bucle tienen una estructura clara y sencilla cuyo significado puede entenderse a primera vista.

18.5 Validaciones

Un flujo de trabajo Cúram válido debe formar un grafo dirigido, conectado y acíclico estructurado en bloques. En su mayor parte, estas propiedades (dirigido, conectado, acíclico) son discretas, de modo que la herramienta de definición de procesos (PDT) puede comprobarlas de forma independiente antes de publicar un proceso. Las validaciones estructurales realizadas en una definición de proceso se realizan en un orden característico y se describen a continuación.

18.5.1 Comprobaciones sintácticas simples

El primer conjunto de validaciones estructurales que se llevan a cabo son las comprobaciones *sintácticas*. Estas garantizan que se configuren correctamente las uniones y divisiones de actividad (consulte Capítulo 17, “División/Unión”, en la página 119) en la definición de proceso. Dichas validaciones incluyen:

- Todas las actividades, salvo las actividades de *inicio* y *fin* deberán tener al menos una transición de entrada y una de salida.
- Cualquier actividad con más de una transición de entrada *deberá* tener especificado un tipo de unión (es decir, un tipo de unión que no sea igual a *NONE*).
- Cualquier actividad con más de una transición de salida *deberá* tener un tipo de división (es decir, un tipo de división que no sea igual a *NONE*).
- Cualquier actividad con *exactamente* una transición de entrada deberá tener un tipo de unión de *NONE*.
- Cualquier actividad con *exactamente* una transición de salida *deberá* tener un tipo de división de *NONE*.
- El tipo de división de una actividad `paralela` deberá ser *NONE*.
- El tipo de unión de una actividad `paralela` deberá ser *NONE*.
- Una actividad `paralela` deberá tener *exactamente* una transición de entrada.
- Una actividad `paralela` deberá tener *exactamente* una transición de salida.
- El tipo de división de la actividad en el lado lejano de la transición de entrada a una actividad `paralela` *deberá* ser *NONE*.
- El tipo de unión de la actividad en el lado lejano de la transición de salida de una actividad `paralela` *deberá* ser *NONE*.

18.5.2 Comprobaciones de grafo

El segundo conjunto de validaciones estructurales que se llevan a cabo son las comprobaciones de *grafo*. Estas garantizan que el grafo del flujo sea un grafo dirigido, conectado y acíclico. Dichas validaciones incluyen:

- El flujo de trabajo deberá formar un grafo 'conectado'. Esto significa que cada actividad *deberá* aparecer en al menos una ruta que vaya de la actividad de *inicio* a la actividad de *finalización*.
- El flujo de trabajo deberá formar un dígrafo acíclico. Esto significa que *no* puede haber ninguna ruta en el flujo de trabajo que pase dos veces por la misma actividad. Esta validación se aplica a ciclos creados solo por las transiciones - los ciclos creados mediante actividades de inicio y fin de bucle (loop-begin-activity y loop-end-activity respectivamente) son perfectamente válidos.

- Cada subgrafo de instancia dentro del flujo de trabajo deberá terminar correctamente. Esto significa que, comenzando en la actividad de *inicio*, toda ruta posible en el flujo de trabajo deberá terminar en la actividad de *finalización*.

18.5.3 Comprobaciones de bloque

El tercer conjunto de validaciones estructurales que se llevan a cabo son las comprobaciones de *bloque*. Dichas comprobaciones garantizan que la estructura de bloques del grafo del flujo sea correcta.

Las construcciones de inicio de bloque: actividad de inicio de proceso, actividad de inicio de bucle, División paralela (AND) y División de selección (XOR). Las construcciones de fin de bloque correspondientes son: actividad de fin de proceso, actividad de fin de bucle, Unión paralela (AND) y Unión de selección (XOR).

Basándose en estas, se ejecutan las siguientes validaciones de estructura de bloque:

- Por cada inicio de bloque deberá haber un cierre de bloque correspondiente (p.ej. si hay una actividad de inicio de bucle en el flujo de trabajo, deberá haber la correspondiente actividad de fin de bucle).
- Los tipos de inicio/fin de bloque deben estar emparejados (p.ej. si hay una división AND paralela en el grafo del flujo de trabajo, deberá emparejarse con la correspondiente unión (AND) paralela).
- Los bloques podrán anidarse pero no solaparse.

Capítulo 19. Servicios web de flujo de trabajo

19.1 Descripción general

Los flujos de trabajo de Cúram pueden interoperar con otros sistemas de flujo de trabajo soportando aspectos específicos del estándar BPEL *Business Process Execution Language* del grupo Oasis. Los procesos BPEL pueden promulgar procesos de flujo de trabajo de Cúram y recibir una notificación cuando el proceso completa.

El motor de flujo de trabajo de Cúram no no está diseñado para ser un motor de orquestación BPEL completo. En cambio, los flujos de trabajo de Cúram deberían poder participar en procesos orquestados por BPEL. Esto se realiza proporcionando funcionalidad para exponer los procesos de flujo de trabajo de Cúram como servicios web que se pueden invocar desde los enlaces de socio de proceso BPEL.

19.2 Exposición de un servicio web de flujo de trabajo

Los servicios web de flujo de trabajo se crean sobre el soporte de servicios web existente en Cúram. En concreto, el motor de flujo de trabajo requiere un objeto de proceso de negocio (BPO) modelado como un servicio web orientado a documento (consulte el capítulo *Servicios web entrantes de Cúram* de la *Guía de referencia de modelado Cúram* para obtener más detalles).

El BPO de servicio web no es más que un frontend del API de promulgación de flujos de trabajo (`curam.util.workflow.impl.EnactmentService`). Sien do esto así, solo hace falta uno de dichos BPO por aplicación. Ya se proporciona un BPO adecuado en la aplicación Cúram: `Logical View::MetaModel::Curam::Facades:: Workflow::WebService::WorkflowProcessEnactment`.

Para utilizar los servicios web de flujo de trabajo, el BPO denominado `Logical View::MetaModel::Curam::Facades:: Workflow::WebService::WorkflowProcessEnactment` debe asignarse a un componente de servidor del servicio web estereotipo.

Los servicios web de Cúram se pueden personalizar de otras formas como, por ejemplo, protegiéndolos mediante WS-Security tal y como se describe en el capítulo *Servicios web seguros* de la *Guía de referencia de modelado de Cúram*. Todas las personalizaciones de servicios web de flujo de trabajo deberán hacerse en este BPO.

Nota: Dado que todos los servicios web de flujo de trabajo están manejados por el mismo BPO, cualquier personalización afectará a todas las definiciones de proceso expuestas servicio web.

19.2.1 Promulgación de procesos

La exposición de una definición de proceso de flujo de trabajo de Cúram como un servicio web solo requiere marcarlo como tal en la herramienta de definición de procesos (PDT) o directamente en los metadatos, como se describe en Capítulo 3, “Metadatos de una definición de proceso”, en la página 13. Una vez marcadas las definiciones de proceso como servicio web, deberán reconstruirse el servidor, el EAR del servidor y el EAR de servicios web.

Al igual que otros servicios web de Cúram, solo podrá accederse al WSDL del servicio una vez desplegado el EAR de los servicios web. El nombre de servicio web de flujo de trabajo es el mismo que el nombre del proceso. Por lo tanto, se puede acceder al WSDL con un URL similar al siguiente: http://servidorprueba:9082/CuramWS/services/<NombreProceso>?wsdl

El contenido del WSDL viene determinado en parte por la entrada al proceso (los atributos WDO marcados como obligatorios para la promulgación) y la salida del proceso (los atributos del WDO marcado como salida del proceso) (consulte 4.2, "Metadatos", en la página 18). El tipo de puerto WSDL es el nombre del proceso y la operación para promulgar un proceso es siempre `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>
```

Figura 2. Tipo de puerto de promulgación de proceso

19.2.2 Devolución de llamada de terminación de proceso

Un sistema externo (probablemente un proceso BPEL, aunque no necesariamente) que promulgue un flujo de trabajo de Cúram a través de servicios web requerirá con frecuencia una notificación de que el proceso se ha completado y posiblemente algunos datos de salida de la definición de proceso. Esto requiere, por cada definición de proceso, especificar un servicio web que se invoque cuando el proceso finalice.

El servicio web de devolución de llamada se especifica en los metadatos de definición de proceso utilizando la PDT o directamente en los metadatos como se describe en Capítulo 3, "Metadatos de una definición de proceso", en la página 13.

Nota: Antes de utilizarlo en una definición de proceso de flujo, el servicio web de devolución de llamada deberá estar registrado como conector de servicio web de salida de Cúram, tal y como se describe en el capítulo *Conectores de servicio web de salida de Cúram* de la *Guía de referencia de modelado de Cúram*.

El servicio web de devolución de llamada debe ser implementado por un sistema externo, pero debe ajustarse a una definición de tipo de puerto especificada por el servicio web de flujo de trabajo de Cúram. Pueden obtenerse más detalles en 19.3, "Invocación desde procesos BPEL".

19.3 Invocación desde procesos BPEL

La creación de procesos BPEL que promulgan procesos de flujo de trabajo de Cúram está fuera del ámbito de este documento. Sin embargo, el WSDL de cada servicio web de proceso de flujo de trabajo contiene información que pueden utilizar los procesos BPEL.

Tipo de puerto de devolución de llamada

Existe un tipo de puerto en WSDL para un servicio web de flujo de trabajo de Cúram que no está implementado por el propio servicio. El nombre de este tipo de puerto es el nombre del proceso con la palabra "Complete" añadido a él (`<NombreProceso>Complete`).

La finalidad de este tipo de puerto no implementado es definir la interfaz de servicio web que un servicio web de flujo de trabajo de Cúram espera que implemente el proceso BPEL que lo promulga. Este tipo de puerto debe ser implementado por el servicio web de devolución de llamada configurado en la definición de proceso (consulte 19.2.2, "Devolución de llamada de terminación de proceso").

```
<!--Implementado por el proceso BPEL-->
  <wsdl:portType name="SomeCuramWorkflowComplete">
    <wsdl:operation name="processCompleted">
      <wsdl:input message="intf:processCompletedRequest"
                  name="processCompletedRequest"/>
    </wsdl:operation>
  </wsdl:portType>
```

Figura 3. Tipo de puerto de devolución de llamada

Tipo de enlace de socio

Técnicamente, lo único que hace falta para permitir a un proceso de flujo de trabajo de Cúram participar en un proceso orquestado BPEL es exponer el proceso como un servicio web. No obstante, es posible añadir algunos metadatos que ayuden al desarrollador de procesos BPEL definiendo los tipos de puerto implicados en el enlace de socio y los roles que desempeñan.

La especificación BPEL permite definir tipos de enlace de socio en el WSDL del servicio que se va a invocar en el enlace de socio utilizando el mecanismo de extensión WSDL. El WSDL generado para un servicio web de flujo de trabajo de Cúram define el tipo de enlace de socio en el que espera participar en y especifica los tipos de puerto que desempeñan cada rol.

```
<!--Tipo de enlace de socio-->
  <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>
```

Figura 4. Extensiones WSDL para BPEL

Capítulo 20. Ubicaciones de archivo

20.1 Descripción general

Si bien hay programas de utilidad tales como la herramienta de definición de procesos (PDT) y otras interfaces de usuario administrativas, con frecuencia es necesario exportar las salidas de dichas herramientas y someterlas a un control de versiones. Por supuesto, será necesario volver a colocar estos archivos externalizados en el sistema del entorno de ejecución cuando se compile o se instale Cúram. El patrón en Cúram es colocar dichos archivos en una carpeta de fuentes predefinida a partir de la cual se cargan en la base de datos (quizá después de algún procesamiento previo). En este capítulo se describe la ubicación de los archivos fuente relacionados con el flujo de trabajo.

20.2 Archivos de definición de proceso de flujo de trabajo

Las definiciones de proceso de flujo de trabajo (publicadas y no publicadas) pueden importarse en la tabla de base de datos relevante utilizando el destino estándar **build database**.

Estas definiciones de proceso de flujo de trabajo deben almacenarse en archivos XML en un subdirectorio `workflow` bajo el directorio de componentes relevante del servidor Cúram (p.ej. `... \EJBServer\components\core\workflow` para el componente `core`, `... \EJBServer\components\Appeal\workflow` para el componente `Appeal`, etc.).

Cada componente de la aplicación Cúram puede tener un directorio workflow que contenga los archivos XML de definición de proceso relevantes para él. Los archivos de definición de proceso almacenados en estos directorios workflow se importarán automáticamente cuando se ejecute el destino **build database**. Si los archivos de definición de proceso no son válidos o si el nombre y la versión de las definiciones no coinciden con los utilizados en los nombres de archivo, la importación fallará.

Los archivos XML de definición de proceso de flujo de trabajo en el sistema de archivos deberán ajustarse a convenio de nomenclatura estricto. Es el siguiente: `Nombre proceso_vVersión proceso.xml`, donde:

- `Nombre proceso` es el nombre del proceso de flujo de trabajo.
- `Versión proceso` es la versión del proceso de flujo de trabajo.

Una misma versión de una definición de proceso puede existir en varios componentes de la aplicación Cúram. La versión importada siempre se obtendrá del componente que tenga el orden de prioridad de componentes más elevado. La prioridad de ordenación de un componente se configura mediante la variable COMPONENT_ORDER_PRECEDENCE.

Cuando se importe una definición de proceso, se le asignará un nuevo identificador de definición de proceso exclusivo en la base de datos en la que se importe. A diferentes versiones de la misma definición de proceso se les asignará el mismo identificador exclusivo y sólo se podrá importar una versión sin publicar de una definición de proceso. Para manejar las definiciones de proceso de flujo de trabajo no válidas cargadas durante la construcción de la base de datos (destino build database), se llevan a cabo estrictas validaciones en el motor de flujo de trabajo. Tales validaciones garantizan que una definición de proceso de flujo de trabajo no pueda cargarse en la memoria caché de definiciones de proceso y ejecutarse a menos que antes haya pasado todas las validaciones de proceso. Estas validaciones se describen en capítulos anteriores de este documento.

20.2.1 Personalización de un archivo de definición de proceso de flujo de trabajo

20.2.1.1 Creación de un archivo de definición de proceso de flujo de trabajo

Todos los archivos de definición de proceso de flujo de trabajo nuevos deben crearse en el subdirectorio workflow del directorio `... \EJBServer\components\custom` . Para crear un archivo de definición de proceso, puede utilizarse la PDT para crear la definición necesaria y especificar todos los detalles. La herramienta puede exportar luego la definición a un archivo y colocarla en la ubicación especificada anteriormente.

20.2.1.2 Modificación de un archivo existente de definición de proceso de flujo de trabajo

Con la PDT, visualice la última versión de la definición de proceso que requiere una modificación. Cree una versión nueva de dicha definición de proceso utilizando la herramienta. Realice los cambios, valide y publique el flujo de trabajo.

Exporte la definición de proceso de flujo de trabajo recién publicada utilizando la PDT y colóquela en el subdirectorio workflow del directorio `... \EJBServer\components\custom` .

20.3 Archivos de definición de sucesos

Los sucesos proporcionan un mecanismo para que los componentes débilmente acoplados de la aplicación Cúram comuniquen información relativa a los cambios de estado en el sistema. Cuando uno módulo de la aplicación genera un suceso, uno o más módulos recibirán notificación de que ha sucedido ese suceso siempre que estén registrados como oyentes de ese suceso. Para utilizar esta funcionalidad, se tienen que definir algunos sucesos, algún código de aplicación debe generar estos sucesos y se tienen que definir y registrar algunos manejadores de sucesos como oyentes de tales sucesos.

Los sucesos se definen en Cúram en archivos XML que especifican las clases de suceso y los tipos de suceso. Estos archivos se crean con una extensión `.evx` y se colocan en la carpeta `events` de un componente de Curam (p. ej. `... EJBServer\components\core\events`) desde donde los scripts de construcción los recogen y procesan.

Hay dos tipos de salida generada por el comando **evgen**; archivos `.java` (para las constantes de código que hacen que el uso de sucesos sea menos proclive a errores) y archivos `.dmx` (scripts de base de datos de Cúram para cargar definiciones de suceso en la base de datos). Los artefactos Java generados a partir de archivos de sucesos fusionados se colocan en el directorio `/build/svr/events/gen/[paquete]`, donde `[paquete]` es el atributo de paquete especificado en el archivo de definición de suceso. Los scripts de base de datos producidos a partir de archivos de sucesos fusionados se colocan en el directorio `/build/svr/events/gen/dmx`.

El capítulo 10 de la *Guía del desarrollador del servidor Cúram* proporciona una descripción completa de los sucesos y de cómo pueden utilizarse en la aplicación Cúram.

Capítulo 21. Configuración

21.1 Descripción general

En su mayor parte, las opciones de configuración no son globales a todas las definiciones de proceso de flujo de trabajo. Más bien son específicas de cada definición y, por lo tanto, están contenidas en la propia definición del proceso. Dicho esto, hay un pequeño número de propiedades de aplicación que afectan a Cúram Workflow Management System en su conjunto. En este capítulo se describen dichas propiedades.

21.2 Propiedades de aplicación

Las siguientes propiedades de aplicación se puede establecer en el archivo Application.prx:

Nombre de propiedad	Descripción
curam.custom. workflow.workresolver	*Finalidad:* El nombre completo de la clase de aplicación que implementa la interfaz de devolución de llamada de `WorkResolver`. Consulte 9.4, "Estrategia de asignación", en la página 67 para obtener más información. *Tipo:* String *Valor predeterminado:* curam.core.sl.impl.DefaultWorkResolver
curam.workflow. automaticallyaddtasktousertasks	*Finalidad:* una vez resueltos los destinos de asignación de una tarea, si dicha tarea se asigna a un único usuario y el valor de esta propiedad se establece a yes/true, el sistema añadirá automáticamente esta tarea a la lista Mis tareas de un usuario en su bandeja de entrada para que pueda trabajar con ella. *Tipo:* String *Valor predeterminado:* NO
curam.custom.notifications. notificationdelivery	*Finalidad:* El nombre completo de la clase de aplicación que implementa la interfaz de devolución de llamada de `NotificationDelivery`. Consulte 14.3, "Estrategia de asignación de notificación", en la página 105 para obtener más información. *Tipo:* String *Valor predeterminado:* curam.core.sl.impl.NotificationDeliveryStrategy
curam.workflow.disable.audit. wdovalueshistory.before.activity	*Finalidad:* El motor de flujo de trabajo llena la tabla de auditoría de datos WDO de la instancia de proceso, 'WDOValuesHistory', en tres puntos diferentes durante la ejecución de una instancia de proceso de flujo de trabajo (antes de la ejecución de una actividad, después de la ejecución de una actividad y antes de la evaluación de las transiciones que salen de una actividad). Cuando se especifica a true, esta propiedad garantiza que no se escriba ningún dato en la tabla de auditoría de datos WDO antes de que una actividad se ejecute. *Tipo:* BOOLEAN *Valor predeterminado:* FALSE
curam.workflow.disable.audit. wdovalueshistory.after.activity	*Finalidad:* El motor de flujo de trabajo llena la tabla de auditoría de datos WDO de la instancia de proceso, 'WDOValuesHistory', en tres puntos diferentes durante la ejecución de una instancia de proceso de flujo de trabajo (antes de la ejecución de una actividad, después de la ejecución de una actividad y antes de la evaluación de las transiciones que salen de una actividad). Cuando se especifica a true, esta propiedad garantiza que no se escriba ningún dato en la tabla de auditoría de datos WDO después de haberse ejecutado una actividad. *Tipo:* BOOLEAN *Valor predeterminado:* FALSE

Nombre de propiedad	Descripción
curam.workflow.disable.audit.wdovalueshistory .transition.evaluation	*Finalidad:* El motor de flujo de trabajo llena la tabla de auditoría de datos WDO de la instancia de proceso, 'WDOValuesHistory', en tres puntos diferentes durante la ejecución de una instancia de proceso de flujo de trabajo (antes de la ejecución de una actividad, después de la ejecución de una actividad y antes de la evaluación de las transiciones que salen de una actividad). Cuando se especifica a true, esta propiedad garantiza que no se escriba ningún dato en la tabla de auditoría de datos WDO antes de que se evalúen las transiciones de una actividad. *Tipo:* BOOLEAN *Valor predeterminado:* FALSE
curam.custom.workflow.processcachesize	*Finalidad:* el motor de flujo de trabajo almacena en memoria caché versiones de definiciones de proceso (para minimizar la sobrecarga cuando se buscan metadatos). Esta propiedad controla el número máximo de versiones de proceso almacenadas en la memoria caché. Cuando se alcanza este número, el motor empezará a expulsar versiones de proceso de la memoria caché utilizando una política de expulsión de la menos utilizada recientemente (least-recently-used). Las modificaciones en tiempo de ejecución del valor de esta propiedad entrarán en vigor la próxima vez que el motor de flujo de trabajo intente insertar una versión de proceso en la memoria caché. *Tipo:* Integer *Valor predeterminado:* 250
curam.batchlauncher.dbtojms. notification.batchlaunchermode	Consulte la *Guía de proceso por lotes de Cúram, sección 5.3* para obtener información adicional.
curam.batchlauncher.dbtojms. notification.encoding	Consulte la *Guía de proceso por lotes de Cúram, sección 5.3* para obtener información adicional.
curam.batchlauncher.dbtojms. notification.host	Consulte la *Guía de proceso por lotes de Cúram, sección 5.3* para obtener información adicional.
curam.batchlauncher.dbtojms. messagespertransaction	Consulte la *Guía de proceso por lotes de Cúram, sección 5.3* para obtener información adicional.
curam.batchlauncher.dbtojms. notification.port	Consulte la *Guía de proceso por lotes de Cúram, sección 5.3* para obtener información adicional.

Capítulo 22. JMSLite

22.1 Introducción

JMSLite es un servidor ligero de Java Message Service (JMS) desarrollado por Cúram que ejecuta paralelamente al entorno de prueba basado en RMI. Por lo tanto puede ejecutar dentro de un entorno de desarrollo integrado (IDE).

Esto permite probar las definiciones de proceso dentro de un entorno de desarrollo integrado (IDE), es decir, sin necesidad de desplegar la aplicación en un servidor de EJB. Cuando se utiliza en combinación con la herramienta de definición de proceso, JMSLite permite a los desarrolladores definir, desplegar y promulgar los flujos de trabajo, todo dentro del entorno de desarrollo integrado (IDE).

22.2 Qué hace JMSLite

JMSLite es un servidor JMS que solo implementa las secciones de la especificación de JMS necesarias para soportar las pruebas de flujos de trabajo de Cúram basadas en el entorno de desarrollo integrado (IDE): es decir, mensajería transaccional y punto a punto. Esto significa que JMSLite soporta transacciones ACID que impliquen a la base de datos de la aplicación *y* a las destinaciones de cola de flujo de trabajo definidas en la infraestructura. No soporta colas personalizadas (definidas por aplicación) ni el dominio publicación/suscripción (p.ej. temas).

Por lo tanto, JMSLite permite al servicio de promulgación de flujo de trabajo y al motor de flujo de trabajo enviar mensajes de JMS de forma asíncrona. Esto significa que las llamadas de la aplicación a las API de infraestructura relacionadas con el flujo de trabajo (tales como los servicios de promulgación y de sucesos) no son bloqueantes. El API pasa mensajes al motor de flujo de trabajo, que pone en marcha las instancias de proceso de forma asíncrona (p.ej. ejecuta actividades automáticas, crea y asigna tareas, etc.).

22.3 Por qué JMSLite

El propósito de JMSLite es hacer que el motor de flujo de trabajo se comporte en un entorno de desarrollo integrado (IDE) de la forma más parecida posible a como se comporta cuando se despliega en un servidor de aplicaciones. Esto incrementa la probabilidad de detectar problemas pronto (mientras se prueba en el entorno de desarrollo integrado (IDE) en vez de detectarlas después (cuando se hacen las pruebas en un servidor de aplicaciones), reduciendo así los riesgos y el coste.

Por ejemplo, considere la siguiente situación: suponga que WMS (ejecutando en un entorno de desarrollo integrado (IDE)) promulga flujos de trabajo de forma *síncrona*.

Recordatorio: En la producción, los flujos se promulgan de forma *asíncrona* porque se supone que tienen un tiempo de vida muy largo (del orden de horas, días o semanas) comparado con las operaciones normales de un usuario (del orden de segundos o milisegundos).
Suponga también que un desarrollador escribe un método que promulga un flujo de trabajo de aprobación de casos automática y después (justo después de llamar al servicio de promulgación) intenta hacer algo con el resultado (p.ej. comprobar si el caso se ha aprobado automáticamente). Puesto que el entorno de prueba funciona de forma diferente (de manera síncrona) a como lo hace en el entorno de producción, el código funcionará correctamente en la prueba, pero fallará en producción (esto es un ejemplo de un error de 'acoplamiento temporal').

Sin embargo, puesto que JMSLite de forma asíncrona, este problema aparecería en el entorno de desarrollo integrado (IDE) de la misma manera que lo haría en un servidor de aplicaciones, lo que permitiría al desarrollador detectarlo antes.

22.4 Utilización de JMSLite

El servidor JMSLite sondea las colas y desempaqueta todos los mensajes que encuentra en ellas. Estos mensajes dan lugar a llamadas del servidor JMSLite al servidor RMI que se necesita para probar en un entorno de desarrollo integrado (IDE) los métodos Cúram (conocido normalmente como `StartServer`). El servidor JMSLite se inicia como una hebra cuando se invoca el proceso (`StartServer`). Puesto que el servidor JMSLite envía mensajes al motor de flujo de trabajo que ejecuta en el servidor RMI, es necesario iniciar StartServer en modo de depuración al depurar métodos de flujo de trabajo.

22.5 Depuración de flujos de trabajo

Normalmente, una aplicación invoca los métodos de la infraestructura de Cúram. Sin embargo, en un flujo de trabajo se hace a menudo al revés, es decir, el motor de flujo de trabajo (infraestructura) llama a un método de aplicación (p.ej. un método de asignación de trabajo). En estos casos, un desarrollador de aplicaciones no puede dar un paso desde la llamada al método `curam.util.workflow.impl.EnactmentService.startProcess()` al método de su aplicación (asignación de trabajo). En este caso, el desarrollador tendrá que establecer puntos de interrupción en el método que desea depurar y, a continuación, ejecutar el método que promulga el flujo de trabajo. El motor de flujo de trabajo invocará entonces (de forma asíncrona) el método de la aplicación, lo que provocará que se llegue al punto de interrupción. El depurador detendrá entonces la ejecución en el punto de interrupción especificado, permitiendo así la depuración normal.

Los métodos de aplicación que entran en la categoría anterior son los siguientes:

- Métodos de actividad automática
- Funciones de asignación de trabajo
- El método de entrega de notificaciones de aplicación
- El método de resolución de trabajo de aplicación

Capítulo 23. Bandeja de entrada y gestión de tareas

23.1 Descripción general

Las tareas se utilizan para asignar trabajo a usuarios del sistema y realizar un seguimiento del mismo, y se generan cuando el motor de flujo de trabajo ejecuta actividades Capítulo 9, “Manual”, en la página 61, Capítulo 10, “Decisión”, en la página 77 o Capítulo 13, “Paralela”, en la página 95. Los usuarios de la aplicación Cúram utilizan la bandeja de entrada y las funciones de gestión de tareas asociadas para gestionar dichas tareas. En los apartados siguientes se describen las opciones de configuración y personalización que están disponibles para las áreas de bandeja de entrada y gestión de tareas del WMS de Cúram.

23.2 Configuración de la bandeja de entrada

23.2.1 Valores de configuración de los tamaños de lista de la bandeja de entrada

Hay una serie de vistas de lista de tareas disponibles en la bandeja de entrada. Estas incluyen las siguientes:

- *Mis tareas abiertas*: una lista de tareas en las que está trabajando actualmente el usuario.
- *Mis tareas aplazadas*: una lista de tareas en las que el usuario está trabajando, pero que se han aplazado a una fecha posterior.
- *Tareas disponibles*: una lista de tareas disponibles en las que puede trabajar el usuario.
- *Resultados de búsqueda de consulta de tareas*: una lista de tareas que son el resultado de ejecutar una consulta de tareas.
- *Tareas de cola de trabajos*: una lista de tareas asignadas a una cola de trabajos.

También hay una lista en la bandeja de entrada que muestra las notificaciones que se han entregado a un usuario.

- *Mis notificaciones*: una lista de notificaciones que se han entregado al usuario.

Las vistas de lista de bandeja de entrada pueden configurarse para limitar el número de registros devueltos al usuario. Pueden establecerse las siguientes propiedades de aplicación en el archivo Application.prx:

Tabla 5. Valores de configuración de los tamaños de lista de la bandeja de entrada

Nombre de propiedad	Descripción
curam.inbox.max.task.list.size	*Finalidad:* el valor de la propiedad controla el número de tareas que se muestran en las diversas vistas de tareas de la bandeja de entrada. Entre las páginas de listas de tareas de la bandeja de entrada afectadas por el valor de esta propiedad se incluyen: Mis tareas abiertas, Mis tareas aplazadas, Tareas disponibles, Búsqueda de consulta de tareas y Tareas de cola de trabajos. Si el número de tareas que se va a visualizar supera el valor especificado, se mostrará un mensaje en el que se informa al usuario que no se están mostrando todos los registros que coinciden con los criterios de búsqueda de la página. Este mensaje muestra tanto el número de tareas visualizadas como también el número total de tareas que coinciden con los criterios de búsqueda. *Tipo:* Integer *Valor predeterminado:* 100
curam.notification.max.list.size	*Finalidad:* el valor de la propiedad controla el número de notificaciones mostradas en la vista de lista de la bandeja de entrada Mis notificaciones. Si el número de notificaciones que se va a visualizar supera el valor especificado, se mostrará un mensaje en el que se informa al usuario que no se están mostrando todos los registros que coinciden con los criterios de búsqueda de la página. Este mensaje muestra tanto el número de notificaciones visualizadas como también el número total de notificaciones que coinciden con los criterios de búsqueda. *Tipo:* Integer *Valor predeterminado:* 100

23.2.2 Valores de configuración de la función Obtener tarea siguiente

Hay una serie de funciones de acceso directo disponibles en la bandeja de entrada para recuperar la tarea siguiente en la que trabajar. Dichas funciones son las siguientes:

- Obtener tarea siguiente - recupera la tarea siguiente de las tareas disponibles al usuario.
- Obtener tarea siguiente tarea de unidad organizativa preferida - recupera la siguiente tarea asignada a la unidad organizativa preferida del usuario.
- Obtener tarea siguiente de cola preferida - recupera la siguiente tarea asignada a la cola de trabajos preferida del usuario.
- Obtener tarea siguiente de cola - recupera la siguiente tarea asignada a una cola de trabajos seleccionada por el usuario.

El algoritmo utilizado por estas funciones de acceso directo para recuperar la tarea siguiente puede configurarse mediante las siguientes propiedades de aplicación en el archivo Application.prx:

Tabla 6. Valores de configuración de la función Obtener tarea siguiente

Nombre de propiedad	Descripción
curam.workflow. reservenexttaskwithpriorityfilter	*Finalidad:* el valor de la propiedad controla si el algoritmo de Obtener siguiente tarea utiliza la prioridad de una tarea para determinar la siguiente tarea a recuperar. Si se establece a *YES*, el valor por omisión, se utilizará la prioridad de la tarea a tal fin (las prioridades tal y como se especifican en la propiedad curam.workflow.taskpriorityorder). De lo contrario, la tarea que se va a recuperar se basará en las tareas que se han asignado al usuario durante más tiempo. *Tipo:* String *Valor predeterminado:* Yes
curam.workflow.taskpriorityorder	*Finalidad:* hay tres prioridades de tarea especificadas en el sistema de gestión de flujos de trabajo, a saber, High (Alta), Medium (Media) y Low (Baja) que se corresponden con los códigos de tabla de códigos TP1, TP2 y TP3 de la tabla de códigos `TaskPriority`. En algunos casos, los clientes pueden tener el requisito de añadir una nueva prioridad de tarea (p.ej. Crítica, con un valor de código de tabla de códigos TP4). La recuperación de tareas empleando la prioridad de tarea que contenga este valor garantizará así que las tareas críticas aparezcan detrás de aquellas que tengan una prioridad baja (cuando lo que se pretende es que las que tengan esta prioridad se recuperen antes). Esta propiedad permite especificar las prioridades de tarea en el orden que sea necesario para satisfacer los requisitos del cliente. *Tipo:* String *Valor predeterminado:* TP1,TP2,TP3

23.2.3 Valores de redirección de tareas y de bloqueo de asignación de las mismas

La redirección de tareas permite al usuario redirigir tareas a otro usuario, a un objeto organizativo (unidad organizativa, puesto o trabajo) o a cola de trabajos durante un período de tiempo especificado. El bloqueo de asignación de tareas permite al usuario asegurarse de que no se asignen tareas durante un período de tiempo especificado. Esta funcionalidad está disponible al usuario en el área Preferencias de tareas de la bandeja de entrada. Sin embargo, puede que no todos los usuarios del sistema necesiten acceder ellos mismos a la configuración de períodos de bloqueo de asignación de tareas o redirección de tareas. Para facilitar este requisito, pueden inhabilitarse estas áreas de funcionalidad de la bandeja de entrada para determinados usuarios mediante identificadores de seguridad. La tabla siguiente detalla los identificadores de seguridad que un usuario debe tener para disponer de estas funciones.

Tabla 7. Identificadores de seguridad y acciones asociadas

Nombre del identificador de seguridad	Acción permitida
UserTaskRedirection. listTaskRedirectionHistoryForUser	Permite a un usuario ver todos los períodos de redirección de la tareas especificados para él.
UserTaskRedirection. redirectTasksForUser	Permite a un usuario crear un periodo de redirección de tareas para sí mismo.
UserTaskRedirection. clearTaskRedirectionForUser	Permite a un usuario borrar uno de sus períodos de redirección de tareas.
UserTaskAllocationBlocking.list. TaskAllocationBlockingHistoryForUser	Permite a un usuario ver todos los períodos de bloqueo de asignación de la tareas especificados para él.
UserTaskAllocationBlocking. blockTaskAllocationForUser	Permite a un usuario crear un periodo de bloqueo de asignación de tareas para sí mismo.
UserTaskAllocationBlocking. clearTaskAllocationBlockForUser	Permite a un usuario borrar uno de sus períodos de bloqueo de asignación de tareas.

23.3 Personalización de la bandeja de entrada

El comportamiento predeterminado de las funcionalidades Acciones de bandeja de entrada, Acciones de tarea y Búsqueda de tareas puede modificarse utilizando Guice para invocar código personalizado que sustituya al comportamiento predeterminado.

Nota: Guice es una infraestructura desarrollada por *Google* que se sale del ámbito de este documento. Puede obtener información adicional sobre Guice consultando la guía del usuario de Guice.

Cúram Workflow Management System contiene los siguientes puntos de personalización y sus correspondientes implementaciones predeterminadas:

Tabla 8. Puntos de personalización

Punto de personalización	Clase de interfaz	Clase de implementación predeterminada
Acciones de bandeja de entrada	`curam.core.hook.task.impl.InboxActions`	`curam.core.hook.task.impl.InboxActionsImpl`
Acciones de tarea	`curam.core.hook.task.impl.TaskActions`	`curam.core.hook.task.impl.TaskActionsImpl`
Búsqueda de tareas y búsqueda de tareas disponibles	`curam.core.hook.task.impl.SearchTask`	`curam.core.hook.task.impl.SearchTaskImpl`
Consulta de tareas	`curam.core.hook.task.impl.TaskQuery`	`curam.core.hook.task.impl.TaskQueryImpl`
Generación de SQL de búsqueda de tareas	`curam.core.hook.task.impl.SearchTaskSQL`	`curam.core.hook.task.impl.SearchTaskSQLImpl`

Las siguientes *Acciones de bandeja de entrada* pueden personalizarse:
- Obtener siguiente tarea
- Obtener siguiente tarea de la unidad organizativa preferida
- Obtener siguiente tarea de la cola preferida
- Obtener siguiente tarea de la cola de trabajos
- Suscribir usuario a la cola de trabajos
- Anular subscripción de usuario de la cola de trabajos

Las siguientes *Acciones de tarea* pueden personalizarse:
- Añadir comentario
- Cerrar
- Crear
- Aplazar
- Reiniciar
- Reenviar
- Modificar tiempo trabajado
- Modificar prioridad
- Modificar fecha límite
- Reasignar
- Añadir a Mis tareas

Los siguientes métodos de *Búsqueda de tareas* y *Búsqueda de tareas disponibles* pueden personalizarse:
- `countAvailableTasks`

- countTasks
- searchAvailableTasks
- searchTask
- validateSearchTask

Los siguientes métodos de *Consulta de tareas* pueden personalizarse:
- createTaskQuery
- modifyTaskQuery
- runTaskQuery
- validateTaskQuery

Los siguientes métodos de generación de SQL de búsqueda de tareas pueden personalizarse. Estos métodos se utilizan para generar el SQL de todas las funciones anteriores de búsqueda de tareas.
- getBusinessObjectTypeSQL
- getCategorySQL
- getCountSQLStatement
- getCreationDateSQL
- getDeadlineSQL
- getFromClause
- getOrderBySQL
- getOrgObjectSQL
- getPrioritySQL
- getReservedBySQL
- getRestartDateSQL
- getSelectClause
- getSQLStatement
- getStatusSQL
- getTaskIDSQL
- getWhereClause

23.3.1 Cómo personalizar la bandeja de entrada

Lo siguiente es una descripción de cómo personalizar la acción de bandeja de entrada curam.core.hook.task.impl.InboxActionsImpl.getNextTask. Pue de seguirse el mismo proceso para personalizar cualquiera de los otros puntos de personalización.

Debe crearse una clase de punto de enganche personalizada. Dicha clase *debe* extender la clase de implementación predeterminada. El diagrama siguiente muestra las relaciones entre las clases:

Figura 5. Diagrama de clases de la personalización

Nota: La clase personalizada *nunca* deberá implementar directamente la clase de interfaz, pues esto podría conducir a excepciones en tiempo de compilación durante una una actualización si se añaden nuevos métodos a la interfaz. En tal caso, la clase personalizada no implementaría los métodos nuevos y, por lo tanto, se rompería el contrato entre la clase de interfaz y la clase de implementación, lo que provocaría excepciones en tiempo de compilación.

23.3.1.1 Personalización de la implementación predeterminada

La signatura de la función getNextTask en la interfaz curam.core.hook.task.impl.InboxActions es la siguiente:

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

@ImplementedBy(InboxActionsImpl.class)
public interface InboxActions {

  public long getNextTask(String userName);

   .
   .
   .
   .
}
```

La implementación predeterminada de la función se especifica en la clase curam.core.hook.task.impl.InboxActionsImpl

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

public class InboxActionsImpl implements InboxActions {

  public long getNextTask(String userName) {
    // El código de la implementación predeterminada va aquí...
  }

  .
  .
  .
  .
}
```

Para personalizar getNextTask, el método deberá implementarse en la nueva clase personalizada creada anteriormente que extiende la clase de implementación predeterminada curam.core.hook.task.impl.InboxActionsImpl .

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

public class CustomInboxActionsImpl extends InboxActionsImpl {

  public long getNextTask(final String userName) {
    // El código personalizado de la implementación va aquí...
  }

}
```

Para asegurarse de que la aplicación ejecute la nueva clase personalizada en lugar de la implementación predeterminada, deberá escribirse una nueva clase custom.hook.task.impl.Module.java que extienda com.google.inject.AbstractModule con el método configure implementado como se muestra en el ejemplo siguiente:

```
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);
    }
}
```

Por último, el nombre de clase `custom.hook.task.impl.Module` deberá insertarse en la columna *ModuleClassName* de la tabla de base de datos *ModuleClassName*. Puede insertarse agregando una fila adicional al archivo `ModuleClassName.DMX` o directamente en la tabla de base de datos si fuera necesario.

Con este enfoque, cuando la aplicación se vuelva a desplegar, el sistema invocará la versión personalizada de la función `getNextTask` en lugar de la implementación predeterminada.

Avisos

Esta información se ha desarrollado para productos y servicios ofrecidos en los Estados Unidos. Es posible que IBM no ofrezca los productos, servicios o características que se describen en este documento en otros países. Solicite información al representante local de IBM acerca de los productos y servicios disponibles actualmente en su zona. Cualquier referencia a un producto, programa o servicio de IBM no pretende afirmar ni implica que sólo pueda utilizarse ese producto, programa o servicio de IBM. En su lugar, se puede utilizar cualquier producto, programa o servicio funcionalmente equivalente que no vulnere ningún derecho de propiedad intelectual de IBM. No obstante, es responsabilidad del usuario evaluar y verificar el funcionamiento de cualquier producto, programa o servicio que no sea de IBM. IBM podría tener patentes o solicitudes de patentes pendientes relacionadas con el tema principal que se describe en este documento. La posesión de este documento no confiere ninguna licencia sobre dichas patentes. Puede enviar consultas sobre licencias, por escrito, a:

IBM Director of Licensing

IBM Corporation

North Castle Drive

Armonk, NY 10504-1785

EE.UU.

Para consultas sobre licencias relacionadas con información de doble byte (DBCS), póngase en contacto con el departamento de propiedad intelectual de IBM de su país o envíe sus consultas, por escrito, a:

Intellectual Property Licensing

Legal and Intellectual Property Law.

IBM Japan Ltd.

19-21, Nihonbashi-Hakozakicho, Chuo-ku

Tokio 103-8510, Japón

El párrafo siguiente no se aplica al Reino Unido ni a ningún otro país donde las disposiciones en él expuestas sean incompatibles con la legislación local: INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN "TAL CUAL" SIN GARANTÍA DE NINGUNA CLASE, YA SEA EXPLÍCITA O IMPLÍCITA, INCLUIDAS, PERO SIN LIMITARSE A ELLAS, LAS GARANTÍAS IMPLÍCITAS DE NO VULNERABILIDAD, COMERCIALIZACIÓN O IDONEIDAD PARA UN PROPÓSITO DETERMINADO. Algunos países no permiten la renuncia a garantías explícitas o implícitas en determinadas transacciones, por lo que puede que esta declaración no sea aplicable en su caso.

La información de este documento puede incluir imprecisiones técnicas o errores tipográficos. Periódicamente se efectúan cambios en la información aquí contenida; estos cambios se incorporarán en nuevas ediciones de la publicación. IBM puede reservarse el derecho de realizar mejoras y/o cambios en los productos y/o programas descritos en esta publicación en cualquier momento sin previo aviso.

Cualquier referencia incluida en esta información a sitios web que no sean de IBM sólo se proporciona para su comodidad y en ningún modo constituye una aprobación de dichos sitios web. El material de esos sitios web no forma parte del material de este producto de IBM y la utilización de esos sitios web se realizará bajo su total responsabilidad.

IBM puede utilizar o distribuir cualquier información que se le facilite de la manera que considere adecuada, sin contraer por ello ninguna obligación con el remitente. Los titulares de licencias de este programa que deseen obtener información sobre el mismo con el fin de permitir: (i) el intercambio de información entre programas creados independientemente y otros programas (incluido éste) y el uso mutuo de información que se haya intercambiado, deben ponerse en contacto con:

IBM Corporation

Dept F6, Bldg 1

294 Route 100

Somers NY 10589-3216

EE.UU.

Dicha información puede estar disponible, sujeta a los términos y condiciones apropiados, incluido en algunos casos el pago de una cuota.

IBM proporciona el programa bajo licencia que se describe en este documento y todo el material bajo licencia disponible para el mismo bajo los términos del Acuerdo de cliente de IBM, el Acuerdo internacional de licencias de programas de IBM o cualquier acuerdo equivalente entre las partes.

Los datos de rendimiento incluidos aquí se determinaron en un entorno controlado. Por lo tanto, los resultados obtenidos en otros entornos operativos pueden variar considerablemente. Algunas mediciones podrían haberse realizado en sistemas en desarrollo y, por lo tanto, no existe ningún tipo de garantía de que dichas mediciones sean las mismas en los sistemas con disponibilidad general. Además, es posible que algunas mediciones se hayan calculado mediante extrapolación. Los resultados reales pueden variar. Los usuarios de este documento deben verificar los datos aplicables a sus entornos específicos.

La información relacionada con productos que no son de IBM se ha obtenido de los proveedores de dichos productos, de sus anuncios publicados o de otras fuentes de disponibilidad pública.

IBM no ha probado estos productos y no puede confirmar la precisión de rendimiento, compatibilidad ni otras afirmaciones relacionadas con productos que no son de IBM. Las preguntas relativas a las prestaciones de productos que no son de IBM deben dirigirse a los proveedores de dichos productos.

Las afirmaciones relativas a las intenciones futuras de IBM están sujetas a cambio o retirada sin previo aviso, y sólo representan objetivos

Todos los precios de IBM que se muestran son precios de distribuidor recomendados por IBM, corresponden al momento actual y están sujetos a cambios sin aviso previo. Los precios de los distribuidores pueden variar.

Esta información se ofrece con fines de planificación únicamente. La información incluida en este documento puede cambiar antes de que los productos descritos estén disponibles.

Esta información contiene ejemplos de datos e informes utilizados en operaciones comerciales diarias. Para ilustrarlos de la manera más completa posible, los ejemplos incluyen los nombres de personas, empresas, marcas y productos. Todos estos nombres son ficticios y cualquier parecido con nombres y direcciones utilizados por empresas comerciales reales son mera coincidencia.

LICENCIA DE COPYRIGHT:

Esta información contiene programas de aplicación de ejemplo en lenguaje fuente, que ilustran técnicas de programación en diversas plataformas operativas. Puede copiar, modificar y distribuir los programas de ejemplo de cualquier forma, sin tener que pagar a IBM, con intención de desarrollar, utilizar, comercializar o distribuir programas de aplicación que estén en conformidad con la interfaz de programación de aplicaciones (API) de la plataforma operativa para la que están escritos los programas de ejemplo. Estos ejemplos no se han probado exhaustivamente bajo todas las condiciones. Por lo tanto, IBM no puede garantizar ni implicar la fiabilidad, capacidad de servicio o función de estos programas. Los programas de ejemplo se proporcionan "TAL CUAL", sin garantía de ningún tipo. IBM no es responsable de ningún daño resultante de la utilización de los programas de ejemplo por parte del usuario.

Todas las copias o fragmentos de las copias de estos programas de ejemplo o cualquier trabajo que de ellos se derive, deberán incluir un aviso de copyright como el que se indica a continuación:

© (el nombre de la empresa) (año). Algunas partes de este código proceden de los programas de ejemplo de IBM Corp.

© Copyright IBM Corp. _escriba el año o los años_. Reservados todos los derechos.

Si visualiza esta información en una copia software, es posible que no aparezcan las fotografías ni las ilustraciones en color.

Marcas registradas

IBM, el logotipo de IBM e ibm.com son marcas registradas de International Business Machines Corp., registradas en muchas jurisdicciones en todo el mundo. Otros nombres de productos y servicios pueden ser marcas registradas de IBM u otras empresas. Encontrará una lista actual de marcas registradas de IBM en la web en "Copyright and trademark information" en http://www.ibm.com/legal/us/en/copytrade.shtml.

Apache es una marca registrada de Apache Software Foundation.

Java y todas las marcas registradas y logotipos basados en Java son marcas registradas de Oracle y/o sus filiales.

Otros nombres pueden ser marcas registradas de sus respectivos propietarios. Otros nombres de empresas, productos o servicios pueden ser marcas registradas o de servicio de terceros.

Impreso en España