Web services API

19/01/2021

Q-flow 5.0

urudata Software

Tabla de Contenido

INTRODUCCIÓN

El propósito de este manual es describir la API de web services de Q-flow para apoyar a desarrolladores de aplicaciones que los utilizan.

Este manual contiene solamente información acerca de la interfaz y del comportamiento de los web services. Para obtener información acerca de cómo instalarlos y configurarlos, consulte el manual de instalación de Q-flow.

ORGANIZACIÓN DE ESTE MANUAL

Primero encontrará una explicación de cómo invocar cualquier servicio, incluyendo cómo hacer uso de la interfaz Swagger. Luego, Q-flow incluye los siguientes web services:

- WebBot
- WebChart
- WebDeploy
- WebFlow
- WebLinks
- WebLists
- WebOperations
- WebOrganization
- WebPackage
- WebQueue
- WebResponse
- WebStart

El manual dedica una sección a cada uno de esos web services.

CONVENCIONES USADAS EN ESTE MANUAL

Este manual representa la firma de un método de un web service de la siguiente forma:

Nombre_del_método(lista_de_parámetros):Tipo_de_Retorno

Los parámetros están separados por comas, y cada parámetro está representado por su tipo seguido de su nombre, como en código C#. Ejemplo: string name. Si un método no devuelve nada, el tipo de retorno es “void”, como en C#.

API DE SERVICIOS

Para invocar los servicios expuestos se debe usar la URL <nombreServidor>/<nombreDirectorioVirtual>/api/<Servicio/<Método>. El nombre del servidor y del directorio son los configurados durante la instalación; el servicio y el método son los ya explicados.

Para autenticarse, puede hacer uso del método que se encuentra en el grupo Auth:

Token(string grant_type, string userLogon, string password, Guid tenantId)

Este método devuelve un token de autenticación al pasarle las credenciales de un usuario: logon, contraseña y el identificador de la instancia en la que se encuentra. El parámetro “grant_type” debe tener el valor “password”. El token obtenido se puede agregar a la autorización del cabezal con clave “bearer” para usar los servicios estando autenticado. Si no se utiliza, se iniciará sesión cada vez que se use un servicio.

Existe una interfaz con Swagger para facilitar el uso experimental de estos web services. Para poder hacer uso de ella, se debe tener en el archivo web.config de la API de servicios la clave “EnableSwaggerUI” en “true”, dentro de la sección “AppSettings”. Para acceder a la interfaz, la URL es <nombreServidor>/<nombreDirectorioVirtual>/swagger/ui/index. Para poder usar la autenticación mencionada, al invocar el método Token, dicho token se encuentra en el valor de la propiedad “access_token” del resultado. Debe copiar dicho token y luego hacer clic en el botón “Autorizar” que se encuentra en la parte superior de la página.

Esto abrirá un cuadro de diálogo en el que se le solicitará el token. En el campo de texto, escriba “bearer”, un espacio y luego pegue el token. Finalmente haga clic en “Autorizar”. Puede entonces proceder a probar los servicios.

DESCRIPCIÓN DE LOS WEB SERVICES

A continuación, se describe cada uno de los web services de Q-flow.

WebBot

El web service WebBot es el que permite que un bot pueda obtener información acerca de qué tareas tiene pendientes, e informar a Q-flow acerca de qué tareas completó y canceló. Una explicación de qué es un bot y para qué sirve puede ser encontrada en el manual del diseñador de procesos del negocio.

Un bot, en general, utilizará los métodos de WebBot de la siguiente forma:

1. Invocará el método GetActiveJobs para obtener la lista de trabajos pendientes.
2. Recorrerá los trabajos devueltos por GetActiveJobs para procesar cada uno de ellos. Para obtener los datos necesarios para procesar un trabajo, invocará el método GetJob, que devuelve un objeto que contiene todos los datos del trabajo.
3. El bot procesa el trabajo. Mientras lo hace, modifica el objeto obtenido mediante el método GetJob. También puede modificar datos de aplicación, roles y adjuntos. Si durante el procesamiento de un trabajo el bot debe obtener el contenido de un archivo adjunto, invoca el método GetAttachmentContent.
4. Una vez procesado el trabajo, si pudo completarlo, el bot invocará el método CompleteJob, pasándole el objeto que modificó para que Q-flow guarde los cambios. Si debe abortar el trabajo (por ejemplo, porque hubo un error), invocará el método AbortJob.

Métodos de WebBot

- **AbortJob(BotJob job, string message): void** – aborta un trabajo del bot. El parámetro “job” representa el trabajo que se desea abortar, y se debe haber obtenido previamente mediante la invocación del método GetActiveJobs. El parámetro “message” permite especificar un mensaje para, por ejemplo, indicar por qué el trabajo fue cancelado.

- **CompleteJob(BotJobMessage job): void** – marca un trabajo del bot como completado. Un bot debe invocar este método al terminar de procesar un trabajo para indicarle a Q-flow que lo ha completado. El parámetro "job" se tiene que haber obtenido previamente mediante la invocación del método GetJob. Al invocar este método, se guardan los cambios efectuados por el bot. En el caso de los adjuntos, si se quitó alguno de la lista de adjuntos, Q-flow lo eliminará. Si se agregó un adjunto (y se le cargó la propiedad "Content" y el nombre), Q-flow lo agregará. Para modificar un adjunto, basta con modificar la propiedad "Content" del objeto AttachmentMessage que lo representa.
- **GetActiveJobs(Guid botId): BotJob[]** – devuelve un vector que indica qué trabajos del bot con identificador "botId" están pendientes.
- **GetActiveJobsByFlow(Guid botId, Guid flowId): BotJob[]** – devuelve un vector que indica qué trabajos del bot con identificador "botId" están pendientes para el proceso con identificador "flowId".
- **GetAttachmentContent(BotJob job, Guid attachId): byte[]** – devuelve el contenido del archivo adjunto indicado por "attachId". El bot debe tener permisos para acceder al archivo.
- **GetJob(BotJob job): BotJobMessage** – obtiene un objeto con los datos necesarios (datos del proceso, parámetros) para procesar el trabajo indicado por el parámetro "job". Entre otras cosas, se trae la lista de adjuntos del proceso (si el paso de bot correspondiente tiene el permiso de listar adjuntos), pero sin el contenido de los adjuntos. Para obtener el contenido de un archivo adjunto, utilice el método GetAttachmentContent.
- **UpdateJob(BotJobMessage job):void** - actualiza un trabajo del bot, sin marcarlo como completado.

WebChart

El web service WebChart permite recuperar información sobre las gráficas del sitio web, así como obtener los datos que abastecen las gráficas "Carga de tareas de mis supervisados" y "Tareas del usuario por estado".

Métodos de WebChart

- **GetMyManagedTaskLoad(Guid packageId):UserTaskLoadResponse[]** – devuelve un vector de objetos de la clase UserTaskLoadResponse. Cada uno de estos objetos contiene información acerca de la carga de tareas de un usuario supervisado por el

usuario cuya cuenta se utilizó para invocar el web service. Sólo se tienen en cuenta las tareas que pertenezcan a procesos del paquete indicado por “packageId” o de paquetes que desciendan de éste.

- **GetUserTaskLoad(Guid packageId, Guid userId): UserTaskLoadResponse** – devuelve un objeto que indica la carga de tareas del usuario con identificador “userId”. Sólo se tienen en cuenta las tareas que pertenezcan a procesos del paquete indicado por “packageId” o de paquetes que desciendan de éste.
- **GetPackageCharts(Guid packageId): ChartMessage[]** – devuelve un arreglo de objetos de la clase ChartMessage, que contiene información sobre una gráfica tal como el nombre, las dimensiones y las columnas. Estos objetos corresponden a las gráficas ubicadas en el paquete identificado por “packageId”.
- **GetMyPackageCharts(Guid packageId): ChartMessage[]** – devuelve un arreglo de objetos de la clase ChartMessage, correspondientes a las gráficas ubicadas en el paquete identificado por “packageId” y que pertenecen al usuario actual.
- **GetChart(Guid chartId): ChartMessage** – devuelve un objeto ChartMessage correspondiente a la gráfica identificada por el parámetro “chartId”.
- **GetMyChart(Guid chartId): ChartMessage** – devuelve un objeto ChartMessage correspondiente a la gráfica identificada por el parámetro “chartId”, siempre que la gráfica le pertenezca al usuario actual.
- **GetChartData(ChartDataRequest request): ChartDataResponse** – devuelve un objeto con los datos que abastecen la gráfica con los parámetros representados en el objeto “request”.
- **GetMyChartData(ChartDataRequest request): ChartDataResponse** – devuelve un objeto con los datos que abastecen la gráfica con los parámetros representados en el objeto “request”, siempre que la gráfica pertenezca al usuario.

WebDeploy

El web service WebDeploy provee funcionalidades de importación y exportación de paquetes (con los elementos contenidos en ellos) y el modelo organizacional, entre otras cosas.

Métodos de WebDeploy

- **ExportChartsById(Guid[] chartIds):ChartMessage[]:** exporta las gráficas identificadas por los identificadores provistos en el parámetro “chartIds”.
- **ExportChartsByPackage(Guid packageId):ChartMessage[]** – exporta las gráficas definidas en el paquete especificado por el parámetro “packageId”.

- **ExportDashboardsById(Guid[] dashboardIds):DashboardMessage[]** – exporta los tableros de control cuyos identificadores son especificados por el parámetro.
- **ExportDashboardsByPackage(Guid packageId):DashboardMessage[]** – exporta los tableros de control del paquete cuyo identificador es indicado en el parámetro.
- **ExportKpisById(Guid[] kpiIds):KpiMessage[]** – exporta los indicadores identificados por los identificadores provistos en el parámetro “kpiIds”.
- **ExportKpisByPackage(Guid packageId):KpiMessage[]** – exporta los indicadores definidos en el paquete especificado por el parámetro “packageId”.
- **ExportLinksByPackage(Guid packageId):LinkMessage[]** – exporta los vínculos y categorías definidos en el paquete especificado por el parámetro “packageId”.
- **ExportLinksByParent(Guid parentLinkId):LinkMessage[]** – exporta los descendientes del vínculo o categoría indicado por el parámetro “parentLinkId”.
- **ExportOrganizationNode(Guid nodeId):OrganizationModelMessage** – exporta el nodo organizacional solicitado en el parámetro “nodeId”. El objeto devuelto incluye la información del nodo solicitado, incluyendo su contenido, así como toda su descendencia. Además incluye elementos de configuración: proveedores de seguridad, roles de seguridad y calendarios.
- **ExportPackage(Guid packageId, bool recursive):PackageMessage** - devuelve un objeto con todos los datos del paquete identificado por “packageId”. Si el parámetro “recursive” tiene valor “true”, además de los datos del paquete identificado por “packageId” el objeto devuelto contiene todo el árbol cuya raíz es ese paquete (el paquete y todos sus descendientes).
- **ExportViewsById(Guid[] viewIds):ViewMessage[]** – devuelve un vector de objetos de la clase ViewMessage con todos los datos de las vistas cuyos identificadores están contenidos en el parámetro “viewIds”.
- **ExportViewsByPackage(Guid packageId):ViewMessage[]** – devuelve un vector de objetos de la clase ViewMessage con todos los datos de las vistas del paquete identificado por “packageId”.
- **ImportCharts(ChartMessage[] charts):void** – importa las gráficas contenidas en el parámetro “charts”.
- **ImportDashboards(DashboardMessage[] dashboards)** – importa los tableros de control contenidos en el parámetro.
- **ImportKpis(KpiMessage[] kpis):void** – importa los indicadores contenidos en el parámetro “kpis”.
- **ImportLinks(LinkMessage[] links, Guid parentLinkId):void** – importa los vínculos contenidos en el parámetro “links”, y los pone dentrol del vínculo o categoría cuyo identificador se indica en el parámetro “parentLinkId”.

- **ImportOrganizationNode(OrganizationModelMessage organizationModel, Guid parentNodeId):void** – importa los datos del modelo organizacional pasados en el parámetro “organizationModel”. Los nodos se importan bajo el nodo identificado por el parámetro “parentNodeId”.
- **ImportPackage(PackageMessage package, Guid parentPackageId, PackageImportOptions options):ImportPackageResponse** - importa un paquete, representado por el parámetro “package”. El parámetro “parentPackageId” es el identificador del paquete dentro del cual se desea importar el paquete. El parámetro “options” permite especificar opciones de importación equivalente a las disponibles en el diseñador de procesos, estas son: “UpdateTemplateParameters” y “FixMissingReferences”. El objeto devuelto tiene una lista de advertencias que indica posibles problemas y decisiones tomadas por Q-flow durante la importación.
- **ImportViews(ViewMessage[] views):void** – importa las vistas contenidas dentro del objeto recibido por parámetro.

WebFlow

El web service WebFlow permite obtener y modificar datos de un proceso fuera del contexto de alguna tarea.

Métodos de WebFlow

- **CheckInAttachment(Guid flowId, Guid attachId):void** – protege el archivo adjunto indicado. Esto es análogo a hacer clic en el botón “Proteger” en el formulario de edición del proceso en el sitio web viejo. Esta funcionalidad está deprecada.
- **CheckOutAttachment(Guid flowId, Guid attachId):void** – desprotege el archivo adjunto indicado. Esto es análogo a hacer clic en el botón “Desproteger” en el formulario de edición del proceso en el sitio web viejo. Esta funcionalidad está deprecada.
- **DeleteAttachment(Guid flowId, Guid attachId):void** – elimina el archivo adjunto indicado.
- **DownloadAttachment(Guid flowId, Guid attachId):byte[]** – Devuelve el contenido del archivo adjunto identificado por “attachId”. Esto es análogo a hacer clic en un adjunto protegido en el formulario de edición del proceso. Vea la descripción de DownloadCheckedOutAttachment para ver la diferencia entre ambos métodos.
- **DownloadCheckedOutAttachment(Guid flowId, Guid attachId):byte[]** – Devuelve el contenido del archivo adjunto identificado por attachId. Esto es análogo a hacer clic en

un adjunto desprotegido en el formulario de edición del proceso. La diferencia entre este método y el método DownloadAttachment importa cuando un usuario desprotegió un archivo adjunto, lo actualizó, y todavía no lo volvió a proteger (notar que la protección y desprotección de adjuntos está deprecada). DownloadCheckedOutAttachment devuelve el borrador que todavía no se protegió, que todavía no es público y, si se deshace la desprotección, nunca lo será. DownloadAttachment devuelve la versión actual pública del archivo. Ejemplo: Juan adjunta el archivo Informe.docx a un proceso. El título del informe es "Informe". Pablo desprotege el informe, cambia el título por "Informe de ventas" y sube sus cambios, pero sin proteger el archivo. En este caso, DownloadAttachment devuelve la versión cuyo título es "Informe", mientras que DownloadCheckedOutAttachment devuelve la versión cuyo título es "Informe de ventas", es decir, el borrador que Pablo todavía no protegió. Una vez que Pablo haya protegido esa versión, ésta será pública y ambos métodos devolverán la misma.

- **EditFlow(FlowMessage request):void** - modifica los datos de un proceso. Para utilizar este método, primero se debe obtener un objeto FlowMessage mediante una llamada al método GetEditFlowInfo. Después se modifica el objeto y se lo envía como parámetro del método EditFlow.
- **GetEditFlowInfo(Guid flowId):FlowMessage** - obtiene los datos de un proceso para poder modificarlos y guardarlos mediante el método EditFlow.
- **GetFlowInfo(Guid flowId):FlowMessage -** obtiene los datos del proceso.
- **UndoCheckOutAttachment(Guid flowId, Guid attachId):void** – deshace la desprotección del archivo adjunto indicado. Esta funcionalidad está deprecada.
- **UploadCheckedOutAttachment(Guid flowId, Guid attachId, byte[] content):void** - envía al servidor un archivo adjunto para actualizar un archivo adjunto del cual antes se hizo checkout, para actualizar su contenido.
- **UploadNewAttachment(Guid flowId, string name, byte[] content):Guid** - agrega un nuevo archivo adjunto al proceso indicado. El parámetro name indica el nombre del archivo, y el parámetro content es un array de bytes con el contenido del nuevo adjunto. Devuelve el identificador que se le asignó al nuevo adjunto.

WebLink

El web service WebLink dispone solamente de un método.

Método de WebLink

- **HasUserPermissionsToSeeLink(string url):bool** – devuelve “true” si el usuario actual (aquel cuyas credenciales se utilizan para invocar el web service) tiene permiso para ver el vínculo cuya URL se indica en el parámetro “url”.

WebList

El web service WebList tiene métodos para obtener listas de tareas.

Métodos de WebList

- **GetCurrentUserTasks():Task[]** – obtiene un vector que contiene las tareas del usuario en cuyo contexto se invoca el método.
- **GetCurrentUserTasksByTemplatesWithData(Guid[] templatesFilter, Guid[] dataToInclude) :DataTable** – devuelve un objeto DataTable con información de las tareas que pertenecen a los templates cuyos identificadores se especifican en el parámetro templatesFilter. Las columnas del objeto DataTable devuelto son:
    - **FlowId:** identificador del proceso.
    - **TaskId:** identificador de la tarea
    - **TaskToId:** identificador de la tarea propia de un usuario (si una tarea está dirigida a n usuarios, está compuesta por n tareas propias de cada uno de esos usuarios).
    - **TemplateCorrelativeId:** identificador correlativo del template.
    - **TemplateId:** identificador del template.
    - **TemplateName:** nombre del template.
    - **TemplateVersionId:** identificador de la versión del template.
    - **TemplateVersionName:** nombre de la versión del template.
    - **FlowCorrelativeId:** identificador correlativo del proceso.
    - **FlowName:** nombre del proceso.
    - **FlowFlag:** bandera del proceso.
    - **FlowStartDate:** fecha de inicio del proceso.
    - **TaskName:** nombre de la tarea.
    - **TaskDescription:** descripción de la tarea.
    - **TaskSubject:** asunto de la tarea.
    - **TaskTimeStarted:** fecha de inicio de la tarea.
- **GetCurrentUserTasksByTemplatesNamesWithDataNames(string[] templatesFilter, string[] dataToInclude):DataTable** – similar al anterior, pero en lugar de utilizar los identificadores de los templates y datos cuyos datos se deben traer, se utilizan sus nombres.

- **GetFlowTasks(Guid flowId):Task[]** – obtiene un vector que contiene las tareas del proceso indicado por el parámetro flowId.
- **GetFlowsUsingQuickSearch(string query):Flow[]** – funciona como la búsqueda rápida del sitio web. Busca procesos a partir de un texto: si el texto es un número, busca un proceso que tenga ese número como identificador correlativo. De lo contrario, busca procesos cuyos nombres sean similares al texto.
- **GetMyTemplates():Template[]** - obtiene un vector con los objetos que representan todos los templates del usuario actual, y contienen información básica de éstos. El resultado es similar a lo que se muestra en la vista "Mis Templates" del sitio web. El usuario actual es aquél cuyas credenciales son utilizadas para invocar el web service.
- **GetPackageViewData(PackageViewDataRequest request):PackageViewDataResponse** – devuelve los datos de una vista. Es como pedirle a Q-flow que muestre una vista, pero desde fuera de Q-flow. En el sitio web, no se muestran todos los datos de una vista en la misma página. Se muestra una cantidad limitada de filas por página. El usuario puede elegir la página que desea ver. El parámetro de este método es un objeto de tipo PackageViewDataRequest que contiene propiedades que permiten especificar el número de página y el tamaño. La propiedad PageSize indica el tamaño de página, y la propiedad PageNumber, el número de página que se desea ver, de modo que invocar el método GetPackageViewData es análogo a hacer clic sobre un número de página de una vista, dado cierto tamaño de página. Para traer todos los elementos de la vista, se puede pedir la página 1 (PageNumber = 1) e indicar que el tamaño de la página sea Int32.MaxValue. Además de esas propiedades, el parámetro contiene las siguientes:
    - **PackageId:Guid:** identificador del paquete al cual pertenece la vista cuyos datos se están obteniendo.
    - **ParameterValues:ParameterValue[]:** en esta propiedad se pueden cargar objetos de tipo ParameterValue para especificar los valores de los parámetros, si la vista cuyos datos se pretenden obtener es una vista con parámetros.
    - **ViewId:Guid:** es el identificador de la vista cuyos datos se quieren obtener.
- **GetTemplates():Template[]** – obtiene un vector con los objetos que representan todos los templates del sistema y contienen información básica de éstos. El resultado es similar a lo que se muestra en la vista "Templates" del sitio web.

WebOperations

El web service WebOperations tiene métodos para delegar una tarea, para finalizar un proceso y para trabajar con el paso de sincronización (ver manual del diseñador de procesos del negocio). Un paso de sincronización puede ser configurado para que espere la ocurrencia de una acción externa. La forma de indicarle a Q-flow que la acción externa tuvo lugar es invocar alguno de los dos métodos del web service WebOperations que tienen ese propósito.

Métodos de WebOperations

- **DelegateTask(DelegationRequest request):void** – delega una tarea. El parámetro "request" es un objeto de tipo DelegationRequest que permite especificar todos los datos necesarios para delegar la tarea.
- **FinalizeFlow(Guid flowId)** – finaliza el proceso indicado por el parámetro "flowId".
- **FinalizeWaitingStepsByName(Guid flowId, string templateStepName):void** – dado un paso de sincronización, especificado por el identificador del proceso al que pertenece y el nombre del paso, indica que ese paso debe terminar la espera por una acción externa.
- **FinalizeWaitingStepsById(Guid flowId, Guid templateStepId):void** – dado un paso de sincronización, especificado por el identificador del proceso al que pertenece y el identificador del paso del template en el que se basa, indica que ese paso debe terminar la espera por una acción externa.
- **GetFlowWaitingSteps(Guid flowId):FlowStepMessage[]** – dado el identificador de un proceso ("flowId"), devuelve una lista de objetos que representan los pasos de sincronización de ese proceso que estén en estado de espera.

WebOrganization

El web service WebOrganization permite interactuar con el modelo organizacional de Q-flow.

Métodos de WebOrganization

- **CreateGroup(string name, string description, Guid groupNodeId):Guid** – crea un grupo. El parámetro "name" indica el nombre del grupo. El parámetro "description" indica la descripción del grupo y "groupNodeId" es el identificador del nodo en el que se debe crear el nuevo grupo.

- **CreateOrganizationNode(string name, string description, Guid parentNodeId):Guid** – crea un nodo organizacional. El parámetro “name” indica el nombre del nodo. El parámetro “description” indica su descripción y el parámetro “parentNodeId” es el identificador del nodo en el que se debe crear el nuevo nodo.
- **CreateUser(string name, string description, Guid userNodeId):Guid** – crea una cuenta de usuario. El parámetro “name” indica el nombre de usuario. El parámetro description indica la descripción del usuario y userNodeId es el identificador del nodo en el que se debe crear la nueva cuenta.
- **DeleteGroup(Guid groupId):void** – elimina el grupo identificado por “groupId”.
- **DeleteOrganizationNode(Guid nodeId):void** – elimina el nodo identificado por “nodeId”.
- **DeleteUser(Guid userId):void** – elimina la cuenta de usuario identificada por “userId”.
- **GetCalendars():CalendarMessage[]** – devuelve un vector de objetos que representan los calendarios existentes en el sistema y contienen los datos de éstos.
- **GetGroup(Guid groupId):GroupMessage** – devuelve un objeto que representa el grupo identificado por “groupId”.
- **GetGroupsWithUser(Guid userId):SecurityMemberMessage[]** – devuelve un vector con objetos que representan aquellos grupos que tienen como miembro el usuario identificado por “userId”.
- **GetNotifiers(): NotifierMessage[]** – devuelve un vector que contiene objetos que representan los servicios de notificación existentes.
- **GetOrganizationNode(Guid nodeId):OrganizationNodeMessage** – devuelve un objeto que representa el nodo cuyo identificado por “nodeId”.
- **GetOrganizationNodeChildren(Guid nodeId, bool recursive):OrganizationNodeTreeMessage** – dado el identificador de un nodo organizacional (nodeId), devuelve sus hijos. Si el parámetro “recursive” tiene valor “true”, devuelve todo el subárbol cuya raíz es el nodo con identificador “nodeId”.
- **GetSecurityMembersByExtendProperties(ExtendedPropertyMessage [] properties, bool includeDisabled):SecurityMemberMessage[]** - dado un conjunto de propiedades extendidas con valores especificados, devuelve un vector con los elementos del modelo organizacional que tienen todas esas propiedades extendidas. Si el parámetro “includeDisabled” es “true”, va a incluir los elementos del modelo organizacional deshabilitados. De lo contrario, si “includeDisabled” es “false” no va a incluir estos elementos.
- **GetSecurityMemberByName(string name):SecurityMemberMessage** – devuelve el elemento del modelo organizacional que tenga el nombre indicado. Notar que el nombre no es único. Puede haber dos miembros con el mismo nombre. En ese caso, se devolverá el primero que se encuentre.

- **GetSecurityProviders():SecurityProviderMessage[]** – devuelve un vector de objetos que representan los proveedores de seguridad y contienen los datos de éstos.
- **GetSecurityRoles():SecurityRoleMessage[]:** - devuelve un vector de objetos que representan los roles de seguridad existentes en el sistema y que contienen los datos de éstos.
- **GetSecurityRolesWithUser(Guid userId):SecurityRolesMessage[]** – devuelve un vector con los roles de seguridad que tienen como miembro el usuario con identificador userId.
- **GetUser(Guid userId):UserMessage** – dado el identificador de un usuario, devuelve un objeto con los datos de ese usuario.
- **GetUserByLogon(string logon):UserMessage** – dado el logon de un usuario, devuelve un objeto con sus datos.
- **GetUsersByExtendedProperties(ExtendedPropertyMessage[] properties, RestrictionType restrictionType):SimpleUserMessage[]** - dado un conjunto de propiedades extendidas con valores especificados, devuelve el conjunto de usuarios que tienen todas esas propiedades con los valores indicados (si RestrictionType es "And") o el conjunto de usuarios que tienen por lo menos una de esas propiedades con el valor indicado (si RestrictionType es "Or"). Por ejemplo, si RestrictionType es "And" y las propiedades son "Sexo" con el valor "M" y "País" con el valor "Uruguay", devuelve todos los usuarios que tengan la propiedad "Sexo" con el valor "M" y la propiedad "País" con el valor "Uruguay". Si la RestrictionType es "Or", devuelve los usuarios que tienen la propiedad "Sexo" en "M" o la propiedad "País" en "Uruguay".
- **GetUsersInGroup(Guid groupId, bool excludeSubNodeMembers):SimpleUserMessage[]** – dado el identificador de un grupo, devuelve todos sus usuarios. Si "excludeSubNodeMembers" es "false", devuelve también los usuarios de todos los grupos descendientes de ese grupo. De lo contrario, devuelve solamente los usuarios del grupo cuyo identificador fue indicado.
- **ModifyGroup(GroupMessage group):void** – dado un objeto GroupMessage que representa un grupo, modifica los datos de ese grupo, reemplazando sus valores por los que son proporcionados en el parámetro. Es conveniente invocar el método GetGroup antes de invocar éste, para obtener un objeto GroupMessage que contenga los datos actualizados del grupo. Después, se modifica el objeto obtenido y se lo utiliza como parámetro para invocar este método.
- **ModifyOrganizationNode(OrganizationNodeMessage node):void** – dado un objeto OrganizationNodeMessage que representa un nodo organizacional, modifica los datos de ese nodo, reemplazando sus valores por los que son proporcionados en el parámetro. Es conveniente invocar el método GetOrganizationMessage antes de invocar éste, para obtener un objeto que contenga los datos actualizados del nodo.

Después, se modifica el objeto obtenido y se lo utiliza como parámetro para invocar este método.

- **ModifyUser(UserMessage user):void** – dado un objeto UserMessage que representa un usuario, modifica los datos de ese usuario, reemplazando sus valores por los que son proporcionados en el parámetro. Es conveniente invocar el método GetUser antes de invocar éste, para obtener un objeto UserMessage que contenga los datos actualizados del usuario. Después, se modifica el objeto obtenido y se lo utiliza como parámetro para invocar este método.
- **MoveGroup(Guid groupId, Guid toNodeId): void** – mueve el grupo indicado por el parámetro "groupId" al nodo indicado por el parámetro "toNodeId".
- **MoveOrganizationNode(Guid newParentId, Guid nodeId):void** – mueve el nodo organizacional con identificador "nodeId" hacia el nodo con identificador "newParentId".
- **MoveUser(Guid userId, Guid toNodeId):void** - mueve el usuario indicado por el parámetro userId al nodo indicado por el parámetro toNodeId.

WebPackage

El web service WebPackage permite interactuar con la estructura de paquetes de Q-flow.

Métodos de WebPackage

- **GetPackage(Guid packageId):SimplePackageMessage** – dado el identificador de un paquete devuelve un objeto con los datos básicos de éste.

WebQueue

El web service WebQueue permite trabajar con colas de trabajo.

Métodos de WebQueue

- **CheckInTask(Guid flowId, Guid taskId, Guid taskToId):void** – dados los identificadores de un proceso (flowId), de una tarea (taskId) y de la parte de esa tarea que es propia de un usuario (taskToId), hace que el usuario con cuyas credenciales se está invocando

el web service deje la tarea. La tarea tiene que haber sido tomada por el usuario antes (por ejemplo, con el método CheckOutTask).

- **CheckOutTask(Guid flowId, Guid taskId, Guid taskToId):void** – dados los identificadores de un proceso (flowId), de una tarea (taskId) y de la parte de esa tarea que es propia de un usuario (taskToId), hace que el usuario con cuyas credenciales se está invocando el web service tome la tarea. La tarea tiene que estar dirigida a una cola de trabajo para la cual el usuario tenga permisos de actuar.
- **GetMyWorkQueueTasks(Guid workQueueId):Task[]** – dado el identificador de una cola de trabajo, permite obtener las tareas de esa cola tomadas por el usuario con cuyas credenciales se invoca el web service.
- **GetUnassignedWorkQueueTasks(Guid workQueueId):Task[]** – dado el identificador de una cola de trabajo, permite obtener las tareas de esa cola que no están tomadas por ningún usuario.

WebResponse

El web service WebResponse tiene métodos para responder tareas. Para especificar una tarea se utilizan:

- El identificador del proceso al que pertenece (**flowId**)
- El identificador del paso que corresponde a la tarea en el proceso (**stepId**).
- El identificador de la parte de la tarea específica a un destinatario (**toId**).

Métodos de WebResponse

- **CheckInAttachment(Guid flowId, Guid stepId, Guid toId, Guid attachId):void** - Protege el archivo adjunto identificado por “attachId”. Esto es análogo a hacer clic en el botón “Proteger” para ese adjunto en el sitio web viejo. Esta funcionalidad está deprecada.
- **CheckOutAttachment(Guid flowId, Guid stepId, Guid toId, Guid attachId):void** - Desprotege el archivo adjunto identificado por “attachid”. Esto es análogo a hacer clic en el botón “Desproteger” para ese adjunto en el sitio web viejo. Esta funcionalidad está deprecada.
- **DeleteAttachment(Guid flowId, Guid stepId, Guid toId, Guid attachId):void** - Elimina el archivo adjunto identificado por “attachId”. Esto es análogo a hacer clic en el botón “Eliminar” para ese adjunto.
- **DownloadAttachment(Guid flowId, Guid stepId, Guid toId, Guid attachId):byte[]** – Devuelve el contenido del archivo adjunto identificado por “attachId”. Esto es análogo a hacer clic en un adjunto protegido en el formulario de tarea. Para ver la diferencia

entre este método y el método DownloadCheckedOutAttachment, vea la descripción de éste.

- **DownloadCheckedOutAttachment(Guid flowId, Guid stepId, Guid toId, Guid attachId):byte[]** - devuelve el contenido del archivo adjunto identificado por "attachId". Esto es análogo a hacer clic en un adjunto desprotegido en el formulario de tarea. La diferencia entre este método y el método DownloadAttachment importa cuando un usuario desprotegió un archivo adjunto, lo actualizó, y todavía no lo volvió a proteger (notar que la protección y desprotección de adjuntos está deprecada). DownloadCheckedOutAttachment devuelve el borrador que todavía no se protegió, que todavía no es público y, si se deshace la desprotección, nunca lo será. DownloadAttachment devuelve la versión actual pública del archivo. Ejemplo: Juan adjunta el archivo Informe.docx a un proceso. El título del informe es "Informe". Pablo desprotege el informe, cambia el título por "Informe de ventas" y sube sus cambios, pero sin proteger el archivo. En este caso, DownloadAttachment devuelve la versión cuyo título es "Informe", mientras que DownloadCheckedOutAttachment devuelve la versión cuyo título es "Informe de ventas", es decir, el borrador que Pablo todavía no protegió. Una vez que Pablo haya protegido esa versión, ésta será pública y ambos métodos devolverán la misma.
- **GetTask(Guid flowId, Guid stepId, Guid toId):TaskMessage** – obtiene un objeto que representa la tarea determinada por el identificador de un proceso ("FlowId"), un paso de ese proceso ("StepId") y la parte de la tarea específica a un destinatario ("ToId"). Una vez obtenido ese objeto, es posible usarlo para responder la tarea que representa por medio de alguno de los otros métodos de ese web service.
- **RespondTask(TaskMessage task):void** – responde la tarea representada por parámetro. Ese objeto debe haber sido obtenido por medio del método GetTask y ser modificado para indicar la respuesta que se da a la tarea y, opcionalmente, un porcentaje de progreso.
- **RespondTaskNow(Guid flowId, Guid stepId, Guid toId, string responseKey):void** – responde una tarea con la respuesta indicada por el parámetro ResponseKey.
- **RespondTaskNowWithProgress(Guid flowId, Guid stepId, Guid toId, string responseKey, byte progress):void** – responde una tarea con la respuesta indicada por el parámetro "ResponseKey", y actualizando el progreso de la tarea con el valor del parámetro "Progress".
- **UndoCheckOutAttachment(Guid flowId, Guid stepId, Guid toId, Guid attachId):void** - Deshace la desprotección del archivo adjunto identificado por "attachId". Esto es análogo a hacer clic en el botón "Deshacer desprotección" para ese adjunto en el sitio web viejo. Esta funcionalidad está deprecada.

- **UploadCheckedOutAttachment(Guid flowId, Guid stepId, Guid toId, Guid attachId, byte[] content):void** - Actualiza el contenido del archivo adjunto identificado por “attachid". El adjunto debe estar desprotegido para actualizar su contenido. Esto es análogo a seleccionar un archivo con el mismo nombre que un adjunto y hacer clic en “Subir” (cuando el adjunto está desprotegido).
- **UploadNewAttachment(Guid flowId, Guid stepId, Guid toId, string name, byte[] content):Guid** - Crea un nuevo archivo adjunto con el nombre y contenido especificados por los parámetros “name” y “content” respectivamente. Devuelve el identificador del adjunto creado. Esto es análogo a seleccionar un archivo con nombre diferente de los adjuntos existentes y hacer clic en “Subir”.

WebStart

El web service WebStart tiene métodos que inician procesos.

Métodos de WebStart

- **StartFlowNow(Guid templateId, string title, string description):Guid** – inicia un proceso basado en el template indicado por el parámetro “templateId”. El parámetro “title” indica el título que se le dará al proceso, y el parámetro “description” indica la descripción. Devuelve el identificador global (Guid) del proceso iniciado.
- **StartFlowNowCorrelativeID(long templateCorrelativeID, string title, string description):Guid** – hace lo mismo que el método StartFlowNow, pero en lugar de recibir un parámetro del tipo Guid para identificar el template en el que se debe basar el nuevo proceso, recibe un parámetro de tipo long, que indica el identificador correlativo del template a ser utilizado.
- **StartFlow(NewFlowMessage flowMessage):Guid** – hace lo mismo que los métodos StartFlowNow y StartFlowNowCorrelative, pero recibe un parámetro del tipo NewFlowMessage para indicar los datos del proceso que será iniciado. El objeto NewFlowMessage, además de tener propiedades que representan el título y la descripción del proceso, tiene propiedades que representan los datos de aplicación, roles y archivos adjuntos del proceso. Estas propiedades pueden ser modificadas para inicializar los datos, roles y archivos adjuntos del proceso que se va a iniciar. Para obtener el objeto NewFlowMessage que se va a utilizar para invocar este método, primero debe invocar el método GetFlowInfoCorrelativeId o el método GetNewFlowInfo.

- **GetNewFlowInfoCorrelativeId(long templateCorrelativeID):NewFlowMessage** – dado un template indicado por su identificador correlativo, devuelve un objeto de tipo NewFlowMessage que puede ser utilizado para iniciar un proceso con el método StartFlow.
- **GetNewFlowInfo(Guid templateId):NewFlowMessage** – dado un template indicado por su identificador global, devuelve un objeto de tipo NewFlowMessage que puede ser utilizado para iniciar in proceso con el método StartFlow.