

---

## *Q-flow 3.0: Web services*

Código del manual: Qf303008ESP  
Versión: 1.1  
Se aplica a: Q-flow 3.04  
Última revisión: 7/5/2009

Q-flow 3.0

# Web services

---

© Urudata Software  
Canelones 1370 • Piso 2 CP11200  
Montevideo, Uruguay  
Teléfono: (598 2) 900 76 68 • Fax: 900 78 56

# Tabla de Contenido

<b>Introducción .....</b>	<b>4</b>
<b>Organización de este manual .....</b>	<b>4</b>
<b>Convenciones usadas en este manual .....</b>	<b>4</b>
<b>Descripción de los web services .....</b>	<b>4</b>
<b>WebStart .....</b>	<b>4</b>
Métodos de WebStart .....	4
<b>WebResponse .....</b>	<b>5</b>
Métodos de WebResponse .....	5
<b>WebLists .....</b>	<b>5</b>
Métodos de WebLists .....	6
<b>WebLinks .....</b>	<b>6</b>
Método de WebLinks .....	7
<b>WebOperations.....</b>	<b>7</b>
Métodos de WebOperations.....	7
<b>WebOrganization.....</b>	<b>7</b>
Métodos de WebOrganization .....	7

## Introducción

El propósito de este manual es describir los 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

Q-flow incluye los siguientes web services:

- WebStart
- WebResponse
- WebLists
- WebLinks
- WebOperations
- WebOrganization

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\_Returno

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#.

## Descripción de los web services

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

### WebStart

El web service WebStart tiene métodos que inician workflows.

### Métodos de WebStart

- **StartFlowNow(Guid templateId, string title, string description):Guid** – inicia un workflow basado en el template indicado por el parámetro templateId. El parámetro title indica el título

que se le dará al workflow, y el parámetro description indica la descripción. Devuelve el identificador global (Guid) del workflow 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 workflow, 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 workflow que será iniciado. El objeto NewFlowMessage, además de tener propiedades que representan el título y la descripción del workflow, tiene propiedades que representan los datos de aplicación, roles y archivos adjuntos del workflow. Estas propiedades pueden ser modificadas para inicializar los datos, roles y archivos adjuntos del workflow 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 workflow 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 workflow con el método StartFlow.

## WebResponse

El web service WebResponse tiene métodos para responder tareas.

### Métodos de WebResponse

- **GetTask(Guid FlowId, Guid StepId, Guid Told):TaskMessage** – obtiene un objeto que representa la tarea determinada por el identificador de un workflow (FlowId), un paso de ese workflow (StepId) y un destinatario de ese paso (Told). 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 Told, string ResponseKey):void** – responde una tarea con la respuesta indicada por el parámetro ResponseKey. La tarea que se responde está determinada por el identificador de un workflow (FlowId), un paso de ese workflow (StepId) y un destinatario de ese paso (Told).
- **RespondTaskNowWithProgress(Guid FlowId, Guid StepId, Guid Told, 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. La tarea que se responde está determinada por el identificador de un workflow (FlowId), un paso de ese workflow (StepId) y un destinatario de ese paso (Told).

## WebLists

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

## Métodos de WebLists

- **GetCurrentUserTasks():Task[]**– obtiene la lista de tareas del usuario en cuyo contexto se invoca el método.
- **GetFlowTasks(Guid flowid):Task[]** – obtiene la lista de tareas del workflow cuyo identificador es indicado por el parámetro flowid.
- **GetTemplates():Template[]** – obtiene una lista de 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.
- **GetMyTemplates():Template[]** - obtiene una lista de 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.
- **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 workflow.
  - **TaskId:** identificador de la tarea
  - **TaskTold:** 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).
  - **TemplateCorrelativId:** 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.
  - **FlowCorrelativId:** identificador correlativo del workflow.
  - **FlowName:** nombre del workflow.
  - **FlowFlag:** bandera del workflow.
  - **FlowStartDate:** fecha de inicio del workflow.
  - **TaskName:** nombre de la tarea.
  - **TaskDescription:** descripción de la tarea.
  - **TaskSubject:** asunto de la tarea.
  - **TaskTimeStarted:** fecha de inicio de la tarea.
  - Una columna por cada uno de los datos especificados en el parámetro dataToInclude, con los valores de esos datos.
- **GetCurrentUserTasksByTemplateNameWithDataNames(string[] templatesFilter, string[] dataToInclude) : DataTable**: este método devuelve lo mismo que el método GetCurrentUserTasksByTemplatesWithData. La diferencia está en los parámetros. En lugar de especificar los identificadores de los templates cuyas tareas se desea obtener, se especifican los nombres de esos templates. Además, los datos de aplicación cuyos valores se desea obtener son especificados también por nombre en el parámetro dataToInclude.

## WebLinks

El web service WebLinks dispone solamente de un método.

## Método de WebLinks

- **HasUserPermissionsToSeeLink(Guid linkId):bool:** devuelve “true” si el usuario actual (aquél cuyas credenciales se utilizan para invocar el web service) tiene permiso para ver el vínculo cuyo identificador es el valor del parámetro linkId.

## WebOperations

El web service WebOperations trabaja en conjunto 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 métodos del web service WebOperations.

### Métodos de WebOperations

- **FinalizeWaitingStepsByName(Guid flowId, string templateStepName): void:** dado un paso de sincronización, especificado por el identificador del workflow 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 workflow 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.

## WebOrganization

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

### Métodos de WebOrganization

- **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.
- **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.
- **DeleteUser(Guid userId, Guid userNodeId):void** – elimina la cuenta de usuario cuyo identificador coincide con el valor del parámetro userId. El parámetro userNodeId es el identificador del nodo en el que se encuentra el usuario.
- **DeleteGroup(Guid groupId, Guid groupNodeId):void** – elimina el grupo cuyo identificador coincide con el valor del parámetro groupId. El parámetro groupNodeId es el identificador del nodo en el que se encuentra el grupo.
- **GetCalendars():CalendarMessage[]** – devuelve un array de objetos que representan los calendarios existentes en el sistema y contienen los datos de éstos.
- **GetGroup(Guid groupId, Guid groupNodeId):GroupMessage** – devuelve un objeto que representa el grupo cuyo identificador coincide con el valor del parámetro groupId. El parámetro groupNodeId indica el identificador del nodo al que pertenece el grupo.

- **GetGroupsWithUser(Guid userId):SecurityMemberMessage[]** – devuelve un array con objetos que representan aquellos grupos que tienen como miembro el usuario cuyo identificador coincide con el valor del parámetro userId.
- **GetSecurityMemberByName(string name): SecurityMemberMessage** - dado el nombre de un miembro del modelo organizacional (nodo, grupo o usuario), devuelve un objeto con sus datos.
- **GetNotifiers: NotifierMessage[]** – devuelve un array que contiene objetos que representan los servicios de notificación existentes.
- **GetSecurityProviders():SecurityProviderMessage[]** – devuelve un array de objetos que representan los proveedores de seguridad y contienen los datos de éstos.
- **GetSecurityRoles():SecurityRoleMessage[]**: - devuelve un array 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 array con objetos que representan aquellos roles de seguridad que tienen como miembro el usuario cuyo identificador coincide con el valor del parámetro userId.
- **GetUser(Guid userId, Guid userNodeId):UserMessage** – dado el identificador de un usuario y el identificador del nodo al que pertenece, 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.
- **GetUserByExtendedProperties(ExtendedPropertyMessage[] properties, RestrictionType restrictionType): UserMessage[]** - 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, boolean excludeSubNodeMembers): UserMessage[]**: 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 este, 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.
- **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 este, 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.
- **MoveUser(Guid user, Guid toNodeid): void** - mueve el usuario indicado por el parámetro user al nodo indicado por el parámetro toNodeid.