
Q-flow 3.0: Referencia de la interfaz de scripting

Código del manual: Qf304010ESP
Versión: 1.0
Se aplica a: Q-flow 3.04
Última revisión: 17/3/2009

Q-flow 3.0

Referencia de la interfaz de scripting

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

Tabla de Contenido

Introducción	4
Referencia de la interfaz de scripting de Q-flow	4
Scripts de pasos de código, de manejadores de eventos y de integraciones	4
CodeScriptBase	4
Scripts de pasos de evaluación por código.....	5
CodeEvaluationScriptBase.....	5
Clases manejadas por los scripts	6
FlowContext.....	6
ThreadContext.....	7
Attachment	8
Data.....	8
DataInstance	9
Parameter.....	10
Role.....	10
RoleMember	11
IScriptHost.....	11

Introducción

Los pasos de código, manejadores de eventos y otros elementos de Q-flow permiten desarrollar scripts. Q-flow incluye un conjunto de clases que permiten manipular información de Q-flow dentro de esos scripts. Dichas clases están en el namespace `Qframework.Steps.Configuration.Scripting`. Este manual describe esas clases.

Referencia de la interfaz de scripting de Q-flow

Los scripts deben ser escritos en C# o Visual Basic .Net, y pueden acceder a las clases de los siguientes namespaces del Framework .NET:

- `System.IO`
- `System.Data` (se puede crear conexiones a bases de datos SQL u Oracle)

Scripts de pasos de código, de manejadores de eventos y de integraciones

Un script de un paso de código, de un manejador de eventos o de una integración debe declarar una clase derivada de la clase `CodeScriptBase`. Cuando un paso de código es agregado a un template, Q-flow automáticamente crea en el script el código que declara la clase y la relación de herencia, además del método `Execute()`. Lo mismo pasa cuando un manejador de eventos es creado. En el caso de los scripts de integraciones, Q-flow crea todo el código del script, aunque el usuario tiene la posibilidad de modificarlo si así lo desea.

El método `Execute()` es un método abstracto de la clase `CodeScriptBase`, y por lo tanto debe ser implementado en el script. Cuando Q-flow ejecuta un paso de código en un workflow, crea un objeto de la clase declarada en el script e invoca el método `Execute()` de ese objeto. Lo mismo sucede cuando ejecuta una integración y cuando ejecuta el manejador de un evento.

CodeScriptBase

La clase `CodeScriptBase` es la clase base de todos los scripts de los pasos de código. Tiene un conjunto de métodos y propiedades protegidos y, por lo tanto, disponibles en sus clases derivadas. Estos métodos y propiedades devuelven objetos de otras clases que también pertenecen al namespace `Qflow.Steps.Configuration.Scripting`, y que son descritas más adelante.

Propiedades

- **Binder: `QflowBinder`:** devuelve un objeto de tipo `QflowBinder`, que sirve para hacer conversiones en métodos que recorren colecciones.

- **Host: IScriptHost:** devuelve un objeto de tipo IScriptHost, que contiene funciones que permiten obtener información general. Cumple las mismas funciones que el objeto Info de Q-flow 2.x.
- **Context: ThreadContext:** representa el hilo en el que se encuentra el paso de código, y permite acceder y manipular información referente a ese hilo.
- **Flow: FlowContext:** representa el workflow en el que se encuentra el paso de código, y permite acceder y manipular información referente a ese workflow.
- **Parameters: List<Parameter>:** devuelve una lista de parámetros. Estos parámetros sólo son útiles en los scripts de las integraciones. Son utilizados en los scripts correspondientes a integraciones.
- **Data: List<Data>:** permite obtener la lista de datos de aplicación del workflow.
- **Roles: List<Roles>:** permite obtener la lista de roles del workflow.
- **Attachments: List<Attachments>:** permite obtener la lista de archivos adjuntos del workflow. Esta lista sirve para obtener los archivos adjuntos de un workflow, pero no para agregarle archivos adjuntos.

Métodos

- **Initialize(IScriptHost host, ThreadContext context):** inicializa la clase con el objeto Host y el objeto ThreadContext indicados. Este método es invocado por Q-flow antes de ejecutar el script.
- **GetData(Guid guid): Data:** permite obtener el dato de aplicación correspondiente al Guid indicado.
- **GetData(string name): Data:** permite obtener un dato de aplicación cuyo nombre es el indicado en el parámetro. Si hay más de un dato de aplicación con ese nombre, devuelve el primero que encuentra.
- **GetRole(Guid guid): Role:** permite obtener el rol correspondiente al Guid indicado.
- **GetRole(string name): Role:** permite obtener un rol cuyo nombre es el indicado en el parámetro. Si hay más de un rol con ese nombre, devuelve el primero que encuentra.
- **GetParameterData(string paramName): Data:** este método sólo es útil en los scripts de las integraciones. Es utilizado internamente por Q-flow y permite obtener el dato de aplicación asociado a un determinado parámetro.

Scripts de pasos de evaluación por código

El script de un paso de evaluación por código debe declarar una clase derivada de la clase CodeEvaluationScriptBase. Cuando un paso de código es agregado a un template, Q-flow automáticamente crea en el script el código que declara la clase y la relación de herencia, además del método Evaluate().

El método Evaluate() es un método abstracto de la clase CodeEvaluationScriptBase, y por lo tanto debe ser implementado en el script del paso de código. Cuando Q-flow ejecute el paso de código en un workflow, creará un objeto de la clase declarada en el script e invocará el método Evaluate() de ese objeto. El resultado de esa invocación, que es “verdadero” o “falso”, determina el resultado de la evaluación del paso.

CodeEvaluationScriptBase

La clase CodeEvaluationScriptBase es la clase base de todos los scripts de los pasos de evaluación por código. Tiene un conjunto de métodos y propiedades protegidos y, por lo tanto, disponibles en sus

clases derivadas. Estos métodos y propiedades devuelven objetos de otras clases que también pertenecen al namespace Qflow.Steps.Configuration.Scripting, y que son descritas más adelante.

Propiedades

Las propiedades de la clase CodeEvaluationScriptBase son las mismas que las de la clase CodeScriptBase, con la excepción de que no posee las propiedades Parameters:List<Parameters> y Host:IScriptHost.

Métodos

Los métodos de la clase CodeEvaluationScriptBase son los mismos que los de la clase CodeScriptBase, con la excepción de que no posee el método GetParameterData(string paramName):Data.

Clases manejadas por los scripts

Las propiedades de las clases de los scripts devuelven objetos de otras clases. Estos objetos representan workflow, hilos, datos y otros elementos. Esta sección describe esas clases.

FlowContext

La clase FlowContext permite manipular información del workflow que está ejecutando el script. La propiedad Flow disponible en los scripts devuelve un objeto de este tipo.

Constructores

La clase FlowContext tiene dos constructores que Q-flow utiliza internamente y no están pensados para ser utilizados en los scripts. Q-flow crea estos objetos automáticamente al crear los objetos de las clases de los scripts. Utilizar estos constructores no debería ser necesario durante el desarrollo de scripts, y no es recomendable.

- **FlowContext():** crea un objeto FlowContext vacío.
- **FlowContext(Guid flowId, long flowCorrelativeId, Guid templateId, Guid templateVersionId):** crea un objeto FlowContext, asignándole el identificador de workflow, el identificador correlativo de workflow, el identificador del template y el identificador de la versión indicados. No carga información del workflow cuyo identificador se indica.

Propiedades

- **FlowCorrelativeID: Long:** devuelve identificador correlativo del workflow. El identificador correlativo es un número que es asignado al workflow en el momento de su creación. Es secuencial, es decir, el número de un workflow es igual al número del workflow anterior más uno.
- **FlowID: Guid:** devuelve el identificador del workflow.
- **TemplateID: Guid:** devuelve el identificador del template en el que se basa el workflow.
- **VersionID: Guid:** devuelve el identificador de la versión en la que se basa el workflow.
- **Name: string:** permite especificar u obtener el nombre del workflow.

- **Description: string:** permite especificar u obtener la descripción del workflow.
- **Importance: byte:** permite especificar u obtener la importancia del workflow. La importancia indica la prioridad.
- **Progress: byte:** permite especificar u obtener el progreso del workflow.
- **Flag: string:** permite especificar u obtener la bandera del workflow. La bandera es lo que se coloca en los pasos de hito y en muchos otros pasos. Es una etiqueta que describe el estado del workflow con un texto cualquiera.
- **Data: List<Data>:** permite obtener una lista que contiene los datos de aplicación del workflow.
- **Roles: List<Roles>:** permite obtener una lista que contiene los roles del workflow.
- **Attachments: List<Attachment>:** permite obtener una lista que contiene los archivos adjuntos al workflow. Esta lista sirve para obtener los archivos adjuntos de un workflow, pero no para agregarle archivos adjuntos.

Métodos

- **GetData(Guid guid): Data:** permite obtener el dato de aplicación correspondiente al Guid indicado.
- **GetData(string name): Data:** permite obtener un dato de aplicación cuyo nombre es el indicado en el parámetro. Si hay más de un dato de aplicación con ese nombre, devuelve el primero que encuentra.
- **GetRole(Guid guid): Role:** permite obtener el rol correspondiente al Guid indicado.
- **GetRole(string name): Role:** permite obtener un rol cuyo nombre es el indicado en el parámetro. Si hay más de un rol con ese nombre, devuelve el primero que encuentra.

ThreadContext

La clase ThreadContext permite manipular información del hilo que está ejecutando el script. La propiedad Thread disponible en los scripts devuelve un objeto de este tipo.

Constructores

La clase ThreadContext tiene dos constructores que Q-flow utiliza internamente y no están pensados para ser utilizados en los scripts. Q-flow crea estos objetos automáticamente al crear los objetos de las clases de los scripts. Utilizar estos constructores no debería ser necesario durante el desarrollo de scripts.

- **ThreadContext(Guid threadID, Guid flowStepID, FlowContext flow):** crea un objeto ThreadContext con el Guid, el Guid de paso y el objeto FlowContext indicados. No carga información del hilo.
- **ThreadContext(Guid threadID, Guid flowStepID, FlowContext flow, List<Parameter> parameters):** crea un objeto ThreadContext de la misma forma que el anterior, pero además permite especificar una lista de parámetros. No carga información del hilo.

Propiedades

- **Parameters: List<Parameter>:** permite especificar u obtener una lista de parámetros. Los parámetros son utilizados en los scripts de integraciones, pero no son útiles en otros scripts.
- **ThreadID: Guid:** permite obtener el Guid del hilo.

- **Flow: FlowContext:** permite obtener un objeto que representa el workflow al que pertenece el hilo.
- **FlowStepID: Guid:** permite obtener el Guid del paso actual del hilo.

Métodos

- **GetParameterData(string paramName): Data:** devuelve el dato de aplicación correspondiente al parámetro cuyo nombre se indica.
- **GetParameter(string paramName): Parameter:** devuelve el parámetro con el nombre especificado.

Attachment

Los objetos de la clase Attachment representan archivos adjuntos al workflow, y son los objetos que Q-flow guarda en la colección Attachments de los objetos FlowContext.

Constructores

La clase Attachment tiene dos constructores que Q-flow utiliza internamente y no están pensados para ser utilizados en los scripts. Q-flow crea estos objetos automáticamente al crear los objetos de las clases de los scripts. Utilizar estos constructores no debería ser necesario durante el desarrollo de los scripts, y no es recomendable.

- **Attachment():** crea un objeto Attachment vacío.
- **Attachment(Guid attachID):** crea un objeto Attachment con el Guid indicado.
- **Attachment(Guid attachID, int attachVersion):** crea un objeto Attachment con el Guid y la versión indicados.
- **Attachment(Guid attachID, int attachVersion, string name):** crea un objeto Attachment con el Guid, la versión y el nombre indicados.

Propiedades

- **AttachID: Guid:** permite obtener el identificador del archivo adjunto.
- **AttachVersion: int:** permite obtener la versión del archivo adjunto.
- **Name: Name:** permite especificar u obtener el nombre del archivo adjunto.
- **Description: string:** permite especificar u obtener la descripción del archivo adjunto.
- **IsLink: bool:** si el archivo adjunto es un vínculo, devuelve "true". De lo contrario, devuelve "false".
- **Url: string:** permite obtener la dirección URL del archivo adjunto.

Data

Los objetos de la clase Data representan datos de aplicación y son los objetos que Q-flow guarda en la colección Data de los objetos FlowContext.

Constructores

La clase Data tiene tres constructores que Q-flow utiliza internamente y no están pensados para ser utilizados en los scripts. Q-flow crea estos objetos automáticamente al crear los objetos de las clases de los scripts. Utilizar estos constructores no debería ser necesario durante el desarrollo de scripts, y no es recomendable.

- **Data():** crea un objeto Data vacío.
- **Data(Guid dataID, string name, Type dataType, bool isArray):** crea un objeto Data con los parámetros indicados.
- **Data(Guid dataID, string name, Type dataType, bool isArray, string lineBlock):** similar al anterior, pero permite especificar un bloque de línea para el dato.

Propiedades

- **DataType: Type:** permite especificar u obtener el tipo del dato.
- **DataTypeName:DomainDataType:** permite especificar u obtener el tipo del dominio del dato.
- **isArray: bool:** indica si el dato es multivaluado o no.
- **LineBlock: string:** permite averiguar a qué bloque de líneas pertenece el dato.
- **DataID: Guid:** permite obtener el identificador del dato.
- **Name: string:** permite obtener el nombre del dato.
- **Value: object:** permite obtener el valor del dato. Si el dato tiene múltiples valores, devuelve el primero.
- **Values:List<DataInstance>:** en el caso de datos con valores múltiples o pertenecientes a un bloque de línea, permite especificar u obtener la lista de valores del dato.

Métodos

- **set_Value(string value):** asigna un valor de tipo string al dato.
- **set_Value(bool value):** asigna un valor de tipo bool al dato.
- **set_Value(DateTime value):** asigna un valor de tipo DateTime al dato.
- **set_value(decimal value):** asigna un valor de tipo decimal al dato.
- **AddValue(object xValue):** agrega un valor al dato (si el dato pertenece a un bloque de líneas, agrega una línea).
- **DelValue(int instance):** borra el dato de la posición indicada por el parámetro "instance".

DataInstance

Los objetos de la clase DataInstance representan valores de un dato que tiene múltiples valores o de un dato que pertenece a un bloque de líneas. La clase DataInstance es la clase de los objetos que son guardados en la colección Values de los objetos de la clase Data.

Constructores

La clase DataInstance tiene dos constructores que Q-flow utiliza internamente y no están pensados para ser utilizados en los scripts. Q-flow crea estos objetos automáticamente al crear los objetos de las clases de los scripts. Utilizar estos constructores no debería ser necesario durante el desarrollo de scripts, y no es recomendable.

- **DataInstance():** crea un objeto DataInstance, sin cargarle nada.

- **DataInstance(Data data, long instance, object value):** crea un objeto DataInstance asignándole el dato indicado por el parámetro "data", el número de instancia indicado por el parámetro "instance" y el valor indicado por el parámetro "value". Tenga en cuenta que no agrega el objeto a la lista de valores de ningún dato.

Propiedades

- **Data: Data:** permite especificar u obtener el dato al que pertenece el DataInstance.
- **Instance: long:** permite especificar u obtener la posición que tiene el DataInstance en la colección de valores del dato al que pertenece.
- **Value: object:** permite especificar u obtener el valor del DataInstance.

Parameter

Los objetos de la clase Parameter son utilizados internamente por Q-flow en los scripts de integraciones. Son los objetos que se guardan en la lista Parameters de los objetos ThreadContext. Representan los parámetros de las integraciones y su asociación con datos de aplicación del workflow. La clase Parameter fue desarrollada sólo para uso interno de Q-flow, y no se recomienda utilizarla. No tiene, además, aplicación fuera del ámbito de Q-flow, por lo que utilizarla en un script es absolutamente innecesario.

Constructor

- **Parameter(string name, Data data):** crea un parámetro con el nombre indicado por el argumento "name" y asociado al dato representado por el argumento "data".

Propiedades

- **Name: string:** permite especificar u obtener el nombre del parámetro.
- **Data: Data:** permite especificar u obtener el objeto que representa el dato asociado al parámetro.

Role

Los objetos de la clase Role representan roles de los workflows y contienen la información que indica que usuario o usuarios los están desempeñando. Son los objetos que Q-flow guarda en la colección Roles de los objetos FlowContext.

Constructores

La clase Role tiene dos constructores que Q-flow utiliza internamente y no están pensados para ser utilizados en los scripts. Q-flow crea estos objetos automáticamente al crear los objetos de las clases de los scripts. Utilizar estos constructores no debería ser necesario durante el desarrollo de scripts, y no es recomendable.

- **Role():** crea un rol vacío.
- **Role(Guid roleID, string roleName):** crea un rol con el identificador y nombre indicados.

Propiedades

- **RoleID: Guid:** devuelve el identificador del rol.
- **Name: string:** devuelve el nombre del rol.
- **IsMultiUser: bool:** permite especificar si el rol tiene múltiples miembros, o averiguar si tiene múltiples miembros.
- **Members: List<RoleMember>:** permite especificar u obtener la lista de miembros del rol.

RoleMember

Los objetos de la clase RoleMember representan miembros de roles. Indican, para un rol, qué usuario lo desempeña. Son los objetos que Q-flow guarda en la colección Members de los objetos de la clase Role.

Constructores

La clase RoleMember tiene tres constructores que Q-flow utiliza internamente y no están pensados para ser utilizados en los scripts. Q-flow crea estos objetos automáticamente al crear los objetos de las clases de los scripts. Utilizar estos constructores no debería ser necesario durante el desarrollo de scripts, y no es recomendable.

- **RoleMember():** crea un miembro de rol vacío.
- **RoleMember(Guid memberID):** crea un miembro de rol con el identificador indicado.
- **RoleMember(Guid memberID, string name):** crea un miembro de rol con el indicador y el nombre indicados.

Propiedades

- **MemberID: Guid:** permite obtener el identificador del miembro de rol.
- **Name: string:** permite obtener el nombre del miembro de rol.

IScriptHost

Los objetos de la interfaz IScriptHost expone los siguientes métodos:

- **AddAttachment(string name, byte[] content): void:** agrega al workflow un archivo adjunto. El parámetro *name* especifica el nombre que tendrá el nuevo adjunto, y el parámetro *content* es el contenido binario de ese archivo.
- **AddAttachment(string path): void:** agrega al workflow como archivo adjunto el archivo cuya ruta es indicada por el parámetro *path*.
- **AddSubstitution(Guid userID, DateTime from, DateTime to, Guid substituteID): void:** crea una suplencia para el usuario correspondiente al identificador *userID*, desde la fecha especificada por el parámetro *from* hasta la fecha indicada por el parámetro *to*, con el usuario correspondiente al identificador *substituteID* como suplente.
- **GetAttachmentContent(string name):byte[]:** dado el nombre de un archivo adjunto, devuelve el contenido de ese archivo en la forma de una secuencia de bytes.
- **GetCalendarDate(Guid calendarID, DateTime date): DateTime:** dado un día (*date*), si ese día es un día de trabajo según el calendario indicado por el parámetro *calendarID*, devuelve ese

mismo día. Si ese día no es un día de trabajo, devuelve el primer día de trabajo posterior a ese día. Por ejemplo: si el día indicado en el parámetro `date` es un sábado, y el calendario especifica que se trabaja de lunes a viernes, la función devuelve un objeto que representa el primer lunes posterior a ese sábado. También toma en cuenta las horas. Por ejemplo: si la hora de salida es las 18:00, y el parámetro `date` indica jueves a las 19:00, el método devuelve el viernes siguiente a ese jueves.

- **GetCalendarDate(Guid calendarID, DateTime date, TimeSpan span): DateTime:** dado un día (*date*) y un período de tiempo (*span*), le suma a ese día ese período de tiempo. Si el resultado es un día de trabajo según el calendario de trabajo, devuelve ese mismo día. De lo contrario, devuelve el primer día de trabajo posterior a ese día. También toma en cuenta el horario de trabajo, por lo que, por ejemplo, si el resultado es un `DateTime` que indica una hora posterior a la jornada laboral, devuelve el día siguiente con la hora de inicio de la jornada.
- **GetManagedUsers(Guid userID): Guid[]:** devuelve en un *array* de objetos `Guid` los identificadores de los usuarios supervisados por el usuario cuyo identificador coincide con el valor del parámetro *userID*.
- **GetManagersOf(Guid userID): Guid[]:** devuelve en un *array* de objetos `Guid` los identificadores de los supervisores del usuario cuyo identificador coincide con el valor del parámetro *userID*.
- **GetSystemParameter(string propertyKey): string:** obtiene el valor del parámetro de instalación indicado por el parámetro *propertyKey*.
- **GetTask(Guid flowStepID): Task:** devuelve la tarea que corresponde al paso cuyo identificador coincide con el valor del parámetro *flowStepID*.
- **GetTask(string flowStepName): Task:** devuelve la tarea que corresponde al paso cuyo nombre coincide con el valor del parámetro *flowStepName*. Si hay dos pasos con el mismo nombre, devuelve el que fue creado más recientemente. Esto puede pasar si el *template* tiene un bucle: puede haber varios pasos del *workflow* que correspondan al mismo paso del *template*. También puede pasar si dos pasos del *template* tienen el mismo nombre.
- **GetUser(Guid userID): User:** dado el identificador de un usuario, devuelve un objeto de tipo `User` que representa al usuario que corresponde a ese identificador.
- **GetUser(string loginName): User:** dado un texto que indica el proveedor de seguridad y el *logon* (nombre de usuario) de un usuario, devuelve un objeto `User` que representa a ese usuario. El proveedor de seguridad y el nombre de usuario deben ser provistos en el formato “proveedor@logon” o en el formato “proveedor\logon” (en este último caso, si el código está en C#, hay que escribir “proveedor\\logon”, dado que “\” es el carácter de escape).
- **GetUser(string loginName, string securityProviderName): User:** dado el nombre de *logon* de un usuario (*logonName*) y el nombre de su proveedor de seguridad (*securityProviderName*) devuelve un objeto de tipo `User` que representa a ese usuario.
- **GetUserID(string logonName): Guid:** dado un texto que indica el proveedor de seguridad y el *logon* (nombre de usuario) de un usuario, devuelve el identificador de ese usuario. El proveedor de seguridad y el nombre de usuario deben ser provistos en el formato “proveedor@logon” o en el format “proveedor\logon” (en este último caso, si el código está en C#, hay que escribir “proveedor\\logon”, dado que “\” es el carácter de escape).
- **GetUserID(string logonName, string securityProviderName): Guid:** dado el nombre de *logon* de un usuario (*logonName*) y el nombre de su proveedor de seguridad (*securityProviderName*) devuelve el identificador de ese usuario.
- **GetUsersByExtendedProperty(string propertyName, int value):Guid[]:** devuelve los identificadores de aquellos usuarios para los que la propiedad indicada por el parámetro *propertyName* tiene el valor indicado por el parámetro *value*.
- **GetUsersByExtendedProperty(string propertyName, string value):Guid[]:** devuelve los identificadores de aquellos usuarios para los que la propiedad indicada por el parámetro *propertyName* tiene el valor indicado por el parámetro *value*.

- **GetUsersByMemberID(Guid memberID):Guid[]**: dado el identificador de algún miembro del organigrama (usuario, nodo o grupo), devuelve los identificadores de los usuarios que pertenecen a ese miembro y de los usuarios que pertenecen a los descendientes de ese miembro. Por ejemplo, si el identificador corresponde a un grupo, devuelve todos los usuarios de ese grupo, todos los usuarios de los grupos que pertenecen a ese grupo, y así sucesivamente. Si el identificador corresponde a un usuario, devuelve el identificador de ese usuario.
- **GetUsersTaskLoad(Guid[] users): Dictionary<Guid, int>**: dado un conjunto de usuarios especificados por sus identificadores, devuelve un diccionario cuya clave es el identificador de cada usuario y cuyo valor es la cantidad de tareas que ese usuario tiene asignadas.
- **GetUsersTaskLoadForTemplate(Guid[] users): Dictionary<Guid, int>**: dado un conjunto de usuarios especificados por sus identificadores, devuelve un diccionario cuya clave es el identificador de cada usuario y cuyo valor es la cantidad de tareas que ese usuario tiene asignadas en el template al que pertenece el workflow que ejecuta el script.
- **GetUserWithLeastTasks(Guid[] users):Guid**: dado un conjunto de usuarios, indicados éstos por sus identificadores, devuelve el identificador de aquél de esos usuarios que tenga menos tareas.
- **GetUserWithLeastTasksForTemplate(Guid[] users): Guid**: dado un conjunto de usuarios, indicados éstos por sus identificadores, devuelve el identificador de aquél de esos usuarios que tenga menos tareas del template al que pertenece el workflow en el que se ejecuta el script.
- **ResetFlowCounter(Guid flowStepID)**: dado el identificador de un paso “Repetir”, pone en 0 el contador que cuenta la cantidad de iteraciones realizadas por ese paso. Es útil para casos en los que ocurre un error durante una iteración y se quiere volver a reintentar la ejecución del paso como si éste nunca se hubiera ejecutado.
- **ResetFlowCounter(string flowStepName)**: dado el nombre de un paso “Repetir”, pone en 0 el contador que cuenta la cantidad de iteraciones realizadas por ese paso. Es útil para casos en los que ocurre un error durante una iteración y se quiere volver a reintentar la ejecución del paso como si éste nunca se hubiera ejecutado.
- **ResolveAddressees(Guid templateStepID): Guid[]**: dado el identificador del paso de un template, devuelve un array con los identificadores de los usuarios a los que ese paso está destinado, sustituyendo los roles por los identificadores de los usuarios que los desempeñan.
- **ResolveAddressees(string templateStepName): Guid[]**: dado el nombre de un paso de un template, devuelve un array con los identificadores de los usuarios a los que ese paso está destinado, sustituyendo los roles por los identificadores de los usuarios que los desempeñan.
- **RespondTask(Guid flowStepID, string responseKey): void**: responde la tarea cuyo identificador coincide con el valor del parámetro *flowStepID*. El valor de la respuesta es el valor del parámetro *responseKey*. La tarea queda registrada como contestada por el primero de sus destinatarios.
- **RespondTask(Guid flowStepID, string responseKey, byte progress): void**: responde la tarea cuyo identificador coincide con el valor del parámetro *flowStepID*. El valor de la respuesta es el valor del parámetro *responseKey*. La tarea queda registrada como contestada por el primero de sus destinatarios. El porcentaje de progreso de la tarea queda actualizado con el valor del parámetro *progress*.
- **RespondTask(string flowStepName, string responseKey): void**: responde la tarea correspondiente al paso cuyo nombre coincide con el valor de *flowStepName*. Si hay más de un paso en esas condiciones, toma en cuenta el que fue iniciado más recientemente. El valor de la respuesta es el valor del parámetro *responseKey*. La tarea queda registrada como contestada por el primero de sus destinatarios.
- **RespondTask(string flowStepName, string responseKey, byte progress): void**: responde la tarea correspondiente al paso cuyo nombre coincide con el valor de *flowStepName*. Si hay más de un paso en esas condiciones, toma en cuenta el que fue iniciado más recientemente. El valor de la respuesta es el valor del parámetro *responseKey*. La tarea queda registrada como

contestada por el primero de sus destinatarios. El porcentaje de progreso de la tarea queda actualizado con el valor del parámetro *progress*.