
Q-flow 3.0: Scripting Interface Reference

| Manual code: Qf3034003ENG
| Version: 1.0
| Applies to: Q-flow 3.034
| Latest revision: 167/3/2009

Q-flow 3.0

Scripting Interface Reference

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

Table of Contents

Introduction	4
Q-flow scripting interface reference	4
Scripts for code steps, event handlers and integrations	4
CodeScriptBase	4
Scripts for code evaluation steps	5
CodeEvaluationScriptBase	5
Classes handled by the scripts	6
FlowContext	6
ThreadContext	7
Attachment	8
Data	8
DataInstance	9
Parameter	10
Role	10
RoleMember	11
IScriptHost.....	11

Introduction

Code steps, event handlers and other elements of Q-flow make it possible to develop scripts. Q-flow includes a set of classes which make it possible to manipulate Q-flow information inside these scripts. These classes are inside the Qflow.Steps.Configuration.Scripting namespace. This manual describes these classes.

Q-flow scripting interface reference

Scripts must be written in C# or Visual Basic .Net, and can access the classes of the following .NET Framework namespaces:

- System.IO
- System.Data (connections can be created to SQL or Oracle databases)

Scripts for code steps, event handlers and integrations

A script for a code step, event manager or integration must declare a class derived from the CodeScriptBase class. When a code step is added to a template, Q-flow automatically creates in the script the code that declares the class and its inheritance relation, as well as the Execute() method. The same occurs when an event manager is created. For integration scripts, Q-flow creates the entire script code, although the user can modify it if he wants to.

The Execute() method is an abstract method from the CodeScriptBase class, and therefore should be implemented in the script. When Q-flow executes a code step in a workflow, it creates an object of the class declared in the script and invokes the Execute() method for this object. The same happens when it executes an integration and when it executes an event manager.

CodeScriptBase

The CodeScriptBase class is the base class for all the scripts used in code steps. It contains a group of methods and properties which are protected and, therefore, available in its derived classes. These methods and properties return objects from other classes which also belong to the Qflow.Steps.Configuration.Scripting namespace, and which are described further on.

Properties

- **Binder: QflowBinder:** Returns an object of the QflowBinder class, which performs conversions in methods that iterate through collections.
- **Host: IScriptHost:** Returns an object of the IScriptHost type, which contains functions that allow one to retrieve general information. It serves the same purpose as the Info object of Q-flow 2.x.

- **Context: ThreadContext:** It represents the thread to which the code step belongs, and allows it to access and manipulate information belonging to that thread.
- **Flow: FlowContext:** Represents the workflow to which the code step belongs, and allows it to access and manipulate information regarding this workflow.
- **Parameters: List<Parameter>:** Returns a list of parameters. These parameters are useful only in scripts belonging to integrations. They are used in these scripts.
- **Data: List<Data>:** Allows one to retrieve the list of the workflow's application data.
- **Roles: List<Roles>:** Allows one to retrieve the list of the workflow's roles.
- **Attachments: List<Attachments>:** Allows one to obtain the list of the workflow's attached files. This list can be used to obtain the workflow's attached files, but not to attach new files.

Methods

- **Initialize(IScriptHost host, ThreadContext context):** initializes the class with the specified Host object and ThreadContext object. This method is invoked by Q-flow before executing the script.
- **GetData(Guid guid): Data:** allows you to obtain the application datum corresponding to the specified Guid.
- **GetData(string name): Data:** allows you to obtain an application datum whose name is the one specified in the parameter. If there is more than one application datum with that name, it returns the first one it finds.
- **GetRole(Guid guid): Role:** allows you to obtain the role corresponding to the specified Guid.
- **GetRole(string name): Role:** allows you to obtain a role whose name is the one specified in the parameter. If there is more than one role with this name, it returns the first one it finds.
- **GetParameterData(string paramName): Data:** this method is only useful in integration scripts. It is used internally by Q-flow and allows you to obtain the application datum associated with a certain parameter.

Scripts for code evaluation steps

The script for a code evaluation step should declare a class derived from the CodeEvaluationScriptBase class. When a code step is added to a template, Q-flow automatically creates in the script the code that declares the class and inheritance relation, as well as the Evaluate() method.

The Evaluate() method is an abstract method from the CodeEvaluationScriptBase class, and therefore should be implemented in the code step script. When Q-flow executes the code step in a workflow, it will create an object from the class declared in the script and will invoke this object's Evaluate(). The result of this invocation, which is "true" or "false", determines the result of the step's evaluation.

CodeEvaluationScriptBase

The CodeEvaluationScriptBase class is the base class for all the scripts used in code evaluation steps. It contains a group of methods and properties which are protected and, therefore, available in its derived classes. These methods and properties return objects from other classes which also belong to the Qflow.Steps.Configuration.Scripting namespace and which are described further on.

Properties

The properties of the CodeEvaluationScriptBase class are the same as those of the CodeScriptBase class, with the exception that it does not have the Parameters:List<Parameters> and Host:IScriptHost properties.

Methods

The CodeEvaluationScriptBase class methods are the same as those of the CodeScriptBase class with the exception that it does not have the GetParameterData(string paramName):Data method.

Classes handled by the scripts

The properties of the classes to which scripts belong return objects of other classes. These objects represent workflows, threads, data and other elements. This section describes these classes.

FlowContext

The FlowContext class allows you to manipulate information from the workflow that is executing the script. The Flow property available in the scripts returns an object of this type.

Constructors

The FlowContext class contains two constructors that Q-flow uses internally and which are not intended to be used in the scripts. Q-flow creates these objects automatically when creating the objects of the script classes. Using these constructors should not be necessary during the development of the scripts, and is not recommended.

- **FlowContext():** creates an empty FlowContext object.
- **FlowContext(Guid flowId, long flowCorrelativeld, Guid templated, Guid templateVersionId):** creates a FlowContext object, assigning to it the specified workflow identifier, the workflow's relative identifier, the template identifier, and the version identifier. It does not load information from the workflow whose identifier is specified.

Properties

- **FlowCorrelativeld: Long:** returns the workflow's relative identifier. The relative identifier is a number that is assigned to the workflow when it is created. It is sequential, that is, the number of a workflow is that of the previous workflow's number plus one.
- **FlowID: Guid:** returns the workflow identifier.
- **TemplateID: Guid:** returns the identifier of the template upon which the workflow is based.
- **VersionID: Guid:** returns the identifier of the version upon which the workflow is based.
- **Name: string:** allows you to specify or obtain the workflow's name.
- **Description: string:** allows you to specify or obtain the workflow's description.
- **Importance: byte:** allows you to specify or obtain the workflow's importance. The importance indicates the priority.
- **Progress: byte:** allows you to specify or obtain the workflow's progress.

- **Flag: string:** allows you to specify or obtain the workflow's flag. The flag is what is set in the milestone steps and in many other steps. It is a label that describes the workflow's status with any text.
- **Data: List<Data>:** allows you to obtain a list which contains the workflow's application data.
- **Roles: List<Roles>:** allows you to obtain a list which contains the workflow's roles.
- **Attachments: List<Attachment>:** allows you to obtain a list which contains the workflow's attached files. This list can be used to obtain a workflow's attached files, but not for adding attached files.

Methods

- **GetData(Guid guid): Data:** allows you to obtain the application datum corresponding to the specified Guid.
- **GetData(string name): Data:** allows you to obtain an application datum whose name is specified in the parameter. If there is more than one application datum with this name, it returns the first one it finds.
- **GetRole(Guid guid): Role:** allows you to obtain the role corresponding to the specified Guid.
- **GetRole(string name): Role:** allows you to obtain a role whose name is specified in the parameter. If there is more than one role with this name, it returns the first one it finds.

ThreadContext

The ThreadContext class allows the script to manipulate information about the thread that is executing the script. The Thread property, which is available to the scripts, returns an object of this type.

Constructors

The ThreadContext class has two constructors that Q-flow uses internally and which are not intended to be used in the scripts. Q-flow creates these objects automatically when creating the objects of the script classes. Using these constructors should not be necessary during the development of the scripts.

- **ThreadContext(Guid threadID, Guid flowStepID, FlowContext flow):** creates a ThreadContext object with the specified Guid, Guid step and FlowContext object. It does not load information about the thread.
- **ThreadContext(Guid threadID, Guid flowStepID, FlowContext flow, List<Parameter> parameters):** creates a ThreadContext object in the same manner as the previous one, but additionally allows you to specify a list of parameters. It does not load information about the thread.

Properties

- **Parameters: List<Parameter>:** allows you to specify or obtain a list of parameters. Parameters are used in the integration scripts, but are not used in other scripts.
- **ThreadID: Guid:** allows you to obtain the Guid from the thread.
- **Flow: FlowContext:** allows you to obtain an object that represents the workflow to which the thread belongs.
- **FlowStepID: Guid:** allows you to obtain the Guid from the thread's actual step.

Methods

- **GetParameterData(string paramName): Data:** returns the application datum corresponding to the parameter whose name is indicated.
- **GetParameter(string paramName): Parameter:** returns the parameter with the specified name.

Attachment

The objects of the Attachment class represent files attached to the workflow, and are the objects Q-flow keeps in the Attachments collection of the FlowContext objects.

Constructors

The Attachment class has two constructors that Q-flow uses internally and which are not intended to be used in the scripts. Q-flow creates these objects automatically when creating the objects of the scripts classes. Using these constructors should not be necessary during the development of scripts, and is not recommended.

- **Attachment():** creates an empty Attachment object.
- **Attachment(Guid attachID):** creates an Attachment object with the specified Guid.
- **Attachment(Guid attachID, int attachVersion):** creates an Attachment object with the specified Guid and version.
- **Attachment(Guid attachID, int attachVersion, string name):** creates an Attachment object with the specified Guid, version and name.

Properties

- **AttachID: Guid:** allows you to obtain the attached file's identifier.
- **AttachVersion: int:** allows you to obtain the version of the attached file.
- **Name: string:** allows you to specify or obtain the name of the attached file.
- **Description: string:** allows you to specify or obtain the description of the attached file.
- **IsLink: bool:** if the attached file is a link, it returns "true". Otherwise, it returns "false".
- **Url: string:** allows you to obtain the URL address of the attached file.

Data

The Data class objects represent application data and are the objects that Q-flow keeps in the Data collection of the FlowContext objects.

Constructors

The Data class has three constructors that Q-flow uses internally and which are not intended to be used in the scripts. Q-flow creates these objects automatically when creating the objects of the script classes. Using these constructors should not be necessary during the development of scripts, and is not recommended.

- **Data():** creates an empty Data object.

- **Data(Guid dataID, string name, Type dataType, bool isArray):** creates a Data object with the specified parameters.
- **Data(Guid dataID, string name, Type dataType, bool isArray, string lineBlock):** similar to the previous one, but allows you to specify a line block for the datum.

Properties

- **DataType: Type:** allows you to specify or obtain the datum's type.
- **DataTypeName:DomainDataType:** allows you to specify or obtain the type of the datum's domain.
- **isArray: bool:** indicates whether the datum is multi-valued or not.
- **LineBlock: string:** allows you to find out to which line block the datum belongs to.
- **DataID: Guid:** allows you to obtain the datum's identifier.
- **Name: string:** allows you to obtain the datum's name.
- **Value: object:** allows you to obtain the datum's value. If the datum has multiple values, it returns the first one.
- **Values:List<DataInstance>:** for data with multiple values or belonging to a line block, it allows you to specify or obtain the list of the datum's values.

Methods

- **set_Value(string value):** assigns a string type value to the datum.
- **set_Value(bool value):** assigns a bool type value to the datum.
- **set_Value(DateTime value):** assigns a DateTime type value to the datum.
- **set_value(decimal value):** assigns a decimal type value to the datum.
- **AddValue(object xValue):** adds a value to the datum (if the datum belongs to a line block, it adds a line).
- **DelValue(int instance):** erases a datum from the position specified by the "instance" parameter.

DataInstance

Objects belonging to the DataInstance class represent values of a datum that has multiple values or of a datum that belongs to a line block. Objects saved in the Values collection of objects belonging to the Data class are instances of the DataInstance class.

Constructors

The DataInstance class has two constructors that Q-flow uses internally and which are not intended to be used in scripts. Q-flow creates these objects automatically when creating the objects of the script classes. Using these constructors should not be necessary during the development of scripts, and is not recommended.

- **DataInstance():** creates a DataInstance object, without loading anything in it.
- **DataInstance(Data data, long instance, object value):** creates a DataInstance object, assigning to it the datum specified by the "data" parameter, the instance number specified by the "instance" parameter and the value specified by the "value" parameter. Keep in mind that it does not add the object to the values list of any datum.

Properties

- **Data: Data:** allows you to specify or obtain the datum to which the DataInstance belongs.
- **Instance: long:** allows you to specify or obtain the position of the DataInstance in the collection of values of the datum to which it belongs.
- **Value: object:** allows you to specify or obtain the value of the DataInstance.

Parameter

Parameter class objects are used internally by Q-flow in the integration scripts. The Parameters list of the ThreadContext objects keeps objects of this class. They represent the parameters of the integrations and their association with the workflow's application data. The Parameter class was developed only for Q-flow's internal use, and using it is not recommended. Moreover, using it is unnecessary, since it is not useful outside the use given to it by Q-flow.

Constructor

- **Parameter(string name, Data data):** creates a parameter with the name specified by "name" associated to the datum represented by "data".

Properties

- **Name: string:** allows you to specify or obtain the parameter's name.
- **Data: Data:** allows you to specify or obtain the object that represents the datum associated with the parameter.

Role

Objects of the Role class represent roles of workflows and contain information that indicates what user or users perform them. These are the objects that Q-flow keeps in the Roles collection of FlowContext objects.

Constructors

The Role class has two constructors that Q-flow uses internally and which are not intended to be used in scripts. Q-flow creates these objects automatically when creating the objects of the script classes. Using these constructors should not be necessary during the development of scripts, and is not recommended.

- **Role():** creates an empty role.
- **Role(Guid roleID, string roleName):** creates a role with the specified identifier and name.

Properties

- **RoleID: Guid:** returns the role's identifier.
- **Name: string:** returns the role's name.
- **IsMultiUser: bool:** allows you to specify or find out whether the role has multiple members.
- **Members: List<RoleMember>:** allows you to specify or obtain the list of the role's members.

RoleMember

Objects of the RoleMember class represent role members. They indicate, for a role, which user performs it. These are the objects that Q-flow keeps in the Members collection of Role objects.

Constructors

The RoleMember class has three constructors that Q-flow uses internally and which are not intended to be used in scripts. Q-flow creates these objects automatically when creating the objects of the script classes. Using these constructors should not be necessary during the development of scripts, and is not recommended.

- **RoleMember():** creates an empty role member.
- **RoleMember(Guid memberID):** creates a role member with the specified identifier.
- **RoleMember(Guid memberID, string name):** creates a role member with the specified identifier and name.

Properties

- **MemberID: Guid:** allows you to obtain the role member identifier.
- **Name: string:** allows you to obtain the role member name.

IScriptHost

IScriptHost objects expose the following methods:

- **AddAttachment(string name, byte[] content): void:** adds an attachment to the workflow. The *name* parameter specifies the name of the new attachment, and the *content* parameter is the binary contents of this file.
- **AddAttachment(string path): void:** adds to the workflow the file whose path is the one specified by the *path* parameter.
- **AddSubstitution(Guid userID, DateTime from, DateTime to, Guid substituteID): void:** creates a substitution for the user corresponding to the userID identifier, from the date specified by the “from” parameter until the date indicated by the “to” parameter, with the user corresponding to the substituteID identifier as the substitute.
- **GetAttachmentContent(string name):byte[]:** given the name of an attachment, it returns the contents of that file in the form of a sequence of bytes.
- **GetCalendarDate(Guid calendarID, DateTime date): DateTime:** given a day (date), if that day is a work day according to the calendar indicated by the calendarID parameter, it returns that same day. If that day is not a work day, it returns the first work day after that day. For example, if the day indicated in the date parameter is a Saturday, and the calendar specifies that work is done from Monday to Friday, the function returns an object that represents the first Monday after that Saturday. It also considers the hours. For example, if the end of the workday is 6:00 pm and the date parameter indicates Thursday at 7:00 pm, the method returns the Friday after that Thursday.
- **GetCalendarDate(Guid calendarID, DateTime date, TimeSpan span): DateTime:** given a day (date) and a time period (span), that time period is added to that day. If the result is a workday according to the work calendar, it returns that same day. Otherwise, it returns the first

workday after that day. It also considers the work hours, and therefore, for example, if the result is a DateTime that indicates an hour after the working day, it returns the next day with the hour of the work day's beginning.

- **GetManagedUsers(Guid userID): Guid[]:** returns an array of Guid objects containing the identifiers of the users supervised by the user whose identifier coincides with the value of the userID parameter.
- **GetManagersOf(Guid userID): Guid[]:** returns an array of Guid objects that contains the identifiers of the supervisors of the user whose identifier coincides with the value of the userID parameter.
- **GetSystemParameter(string propertyKey): string:** obtains the value of the installation parameter indicated by the propertyKey parameter.
- **GetTask(Guid flowStepID): Task:** returns the task that corresponds to the step whose identifier coincides with the value of the flowStepID parameter.
- **GetTask(string flowStepName): Task:** returns the task that corresponds to the step whose name coincides with the value of the flowStepName parameter. If there are two steps with the same name, it returns the one that was created most recently. This can happen if the template has a loop. There can be several workflow steps that correspond to the same template step. It can also happen if two steps of the template have the same name.
- **GetUser(Guid userID): User:** given a user's identifier, it returns a User object that represents the user corresponding to that identifier.
- **GetUser(string loginName): User:** does the same as the previous function, but instead of receiving the logon name and security provider separately, it receives both in the same parameter in the "provider\logon" format (if you are using C#, you must write "provider\\logon" in the code) or in the "provider@logon" format.
- **GetUser(string loginName, string securityProviderName): User:** given a user's logon name (logonName) and the name of his security provider (securityProviderName), it returns a User object that represents that user.
- **GetUserID(string logonName): Guid:** does the same as the previous function, but instead of receiving the logon name and the security provider separately, it receives both in the same parameter in the "provider\logon" format (if you are using C#, you must write "provider\\logon" in the code) or in the "provider@logon" format.
- **GetUserID(string logonName, string securityProviderName): Guid:** given the logon name of a user (logonName) and the name of his security provider (securityProviderName), it returns the identifier of that user.
- **GetUsersByExtendedProperty(string propertyName, int value):Guid[]:** returns the ids of those users for which the property specified by the *propertyName* parameter has the value specified by the *value* parameter.
- **GetUsersByExtendedProperty(string propertyName, string value):Guid[]:** returns the ids of those users for which the property specified by the *propertyName* parameter has the value specified by the *value* parameter.
- **GetUsersByMemberID(Guid memberID):Guid[]:** given the id of an organizational model member (a user, a node or a group), it returns the ids of those users which belong to that member and of the users which belong to descendants of that member. For example, if the id belongs to a group, it returns all users of that group, plus all users of groups that belong to that group, and so forth. If the id belongs to a user, it returns the id of that user.
- **GetUsersTaskLoad(Guid[] users): Dictionary<Guid, int>:** given a set of users specified by their ids, it returns a dictionary whose key is each user's id and whose value is the number of active tasks each user has been assigned.
- **GetUsersTaskLoadForTemplate(Guid[] users): Dictionary<Guid, int>:** given a set of users specified by their ids, it returns a dictionary whose key is each user's id and whose value is the number of active tasks assigned to each user in the template to which the current workflow belongs.

- **GetUserWithLeastTasks(Guid[] users):Guid:** given a set of users specified by their ids, it returns the id of the one which has the smaller number of tasks assigned to him.
- **GetUserWithLeastTasksForTemplate(Guid[] users): Guid:** given a set of users specified by their ids, it returns the id of the one which has the smaller number of tasks assigned to him in the current template.
- **ResetFlowCounter(Guid flowStepID):** given the id of a "Repeat" step, it sets to zero the counter that counts the number of iterations made by that step. This is useful in cases in which errors occur during an iteration and one wants the workflow to retry the execution of the step as if it had never been executed.
- **ResetFlowCounter(string flowStepName):** given the name of a "Repeat" step, it sets to zero the counter that counts the number of iterations made by that step. This is useful in cases in which errors occur during an iteration and one wants the workflow to retry the execution of the step as if it had never been executed.
- **ResolveAddressees(Guid templateStepID): Guid[]:** given the identifier of a template step, it returns an array with the identifiers of the users to whom that step is addressed, substituting the roles with the identifiers of the users that represent them.
- **ResolveAddressees(string templateStepName): Guid[]:** given the name of a template step, it returns an array with the identifiers of the users to whom that step is addressed, substituting the roles with the identifiers of the users that represent them.
- **RespondTask(Guid flowStepID, string responseKey): void:** responds to the task whose identifier coincides with the value of the flowStepID parameter. The value of the response is the value of the responseKey parameter. The task is registered as answered by the first of its addressees.
- **RespondTask(Guid flowStepID, string responseKey, byte progress): void:** responds to the task whose identifier coincides with the value of the flowStepID parameter. The value of the response is the value of the responseKey parameter. The task is registered as answered by the first of its addressees. The progress percentage of the task is updated with the value of the progress parameter.
- **RespondTask(string flowStepName, string responseKey): void:** responds to the task corresponding to the step whose name coincides with the flowStepName value. If there is more than one step in these conditions, it considers the one that was most recently initiated. The value of the response is the value of the responseKey parameter. The task is registered as answered by the first of its addressees.
- **RespondTask(string flowStepName, string responseKey, byte progress): void:** responds to the task corresponding to the step whose name coincides with the flowStepName value. If there is more than one step in these conditions, it considers the one that was most recently initiated. The value of the response is the value of the responseKey parameter. The task is registered as answered by the first of its addressees. The progress percentage of the task is updated with the value of the progress parameter.