Referencia de la API de gRPC

Manejo de errores

La API de gRPC para Zeebe se expone a través de la puerta de enlace, que actúa como un proxy para el intermediario. En general, esto significa que el cliente ejecuta una llamada remota en la puerta de enlace, que luego se traduce a un protocolo binario especial que la puerta de enlace utiliza para comunicarse con el intermediario.

Como resultado de esta representación, cualquier error que ocurra entre la puerta de enlace y el intermediario *para el cual el cliente no tiene la culpa* (por ejemplo, la puerta de enlace no puede deserializar la respuesta del intermediario, el intermediario no está disponible, etc.) se informa al cliente utilizando el siguientes códigos de error.

- `GRPC_STATUS_RESOURCE_EXHAUSTED`: si el corredor recibe demasiadas solicitudes más de lo que puede manejar, inicia la contrapresión y rechaza las solicitudes con este código de error. En este caso, es posible volver a intentar las solicitudes con una estrategia de reintento adecuada. Si recibe muchos de estos errores en un período de tiempo pequeño, indica que el corredor está constantemente bajo una carga alta. Se recomienda reducir la tasa de solicitudes. Cuando se inicia la contrapresión, el corredor puede rechazar cualquier solicitud, excepto *CompleteJob* RPC y *FailJob* RPC. Estas solicitudes están en la lista blanca para contrapresión y siempre son aceptadas por el corredor, incluso si está recibiendo solicitudes por encima de sus límites.
- `GRPC_STATUS_UNAVAILABLE`: si la puerta de enlace está en un estado no válido (por ejemplo, sin memoria)
- `GRPC_STATUS_INTERNAL`: para cualquier otro error interno que haya ocurrido entre la puerta de enlace y el intermediario.

Este comportamiento se aplica a todos los RPC posibles; En estos casos, es posible que el reintento tenga éxito, pero se recomienda hacerlo con una política de reintento adecuada (por ejemplo, una combinación de retroceso exponencial o fluctuación de fase envuelta en un interruptor de circuito).

En la documentación a continuación, los errores documentados son errores de lógica de negocios, es decir, errores que son el resultado de la lógica de procesamiento de solicitudes, y no errores de serialización, red u otros errores más generales.

> Como el servidor / cliente gRPC se basa en el código generado, tenga en cuenta que cualquier llamada realizada al servidor puede devolver errores como se describe en la especificación aquí .

Servicio de puerta de enlace

La API Zeebe gRPC se expone a través de un único servicio de puerta de enlace.

ActivateJobs RPC

Itera a través de todas las particiones conocidas round-robin y se activa hasta el máximo solicitado y las devuelve al cliente a medida que se activan.

Entrada: ActivateJobsRequest

```
message ActivateJobsRequest {
  // the job type, as defined in the BPMN process (e.g. <zeebe:taskDefinition
  // type="payment-service" />)
  string type = 1;
  // the name of the worker activating the jobs, mostly used for logging
purposes
  string worker = 2;
  // a job returned after this call will not be activated by another call until
the
  // timeout has been reached
  int64 timeout = 3;
  // the maximum jobs to activate by this request
  int32 maxJobsToActivate = 4;
  // a list of variables to fetch as the job variables; if empty, all visible
variables at
  // the time of activation for the scope of the job will be returned
  repeated string fetchVariable = 5;
  // The request will be completed when at least one job is activated or after
the requestTimeout.
  // if the requestTimeout = 0, a default timeout is used.
  // if the requestTimeout < 0, long polling is disabled and the request is
completed immediately, even when no job is activated.
  int64 requestTimeout = 6;
}
```

Salida: ActivateJobsResponse

```
message ActivateJobsResponse {
  // list of activated jobs
  repeated ActivatedJob jobs = 1;
}

message ActivatedJob {
  // the key, a unique identifier for the job
  int64 key = 1;
  // the type of the job (should match what was requested)
  string type = 2;
  // the job's workflow instance key
  int64 workflowInstanceKey = 3;
  // the bpmn process ID of the job workflow definition
  string bpmnProcessId = 4;
  // the version of the job workflow definition
  int32 workflowDefinitionVersion = 5;
  // the key of the job workflow definition
  int64 workflowKey = 6;
  // the associated task element ID
  string elementId = 7;
  // the unique key identifying the associated task, unique within the scope of
the
  // workflow instance
  int64 elementInstanceKey = 8;
  // a set of custom headers defined during modelling; returned as a serialized
  // JSON document
  string customHeaders = 9;
  // the name of the worker which activated this job
  string worker = 10;
  // the amount of retries left to this job (should always be positive)
  int32 retries = 11;
  // when the job can be activated again, sent as a UNIX epoch timestamp
  int64 deadline = 12;
  // JSON document, computed at activation time, consisting of all visible
variables to
  // the task scope
  string variables = 13;
}
```

Errores

GRPC_STATUS_INVALID_ARGUMENT

Devuelto si:

- el tipo está en blanco (cadena vacía, nulo)
- el trabajador está en blanco (cadena vacía, nulo)
- tiempo de espera inferior a 1
- la cantidad es menor que 1

CancelWorkflowInstance RPC

Cancela una instancia de flujo de trabajo en ejecución

Entrada: CancelWorkflowInstanceRequest

```
message CancelWorkflowInstanceRequest {
  // the workflow instance key (as, for example, obtained from
  // CreateWorkflowInstanceResponse)
  int64 workflowInstanceKey = 1;
}
```

Salida: CancelWorkflowInstanceResponse

```
message CancelWorkflowInstanceResponse {
}
```

Errores

GRPC_STATUS_NOT_FOUND

Devuelto si:

- no existe una instancia de flujo de trabajo con la clave dada. Tenga en cuenta que, dado que las instancias de flujo de trabajo se eliminan una vez finalizadas, podría significar que la instancia existió en algún momento.

CompleteJob RPC

Completa un trabajo con la carga útil dada, lo que permite completar la tarea de servicio asociada.

Entrada: CompleteJobRequest

```
message CompleteJobRequest {
  // the unique job identifier, as obtained from ActivateJobsResponse
  int64 jobKey = 1;
  // a JSON document representing the variables in the current task scope
  string variables = 2;
}
```

Salida: CompleteJobResponse

```
message CompleteJobResponse {
}
```

Errores

GRPC_STATUS_NOT_FOUND

Devuelto si:

- no existe trabajo con la clave de trabajo dada. Tenga en cuenta que, dado que los trabajos se eliminan una vez completados, podría ser que este trabajo existiera en algún momento.

GRPC_STATUS_FAILED_PRECONDITION

Devuelto si:

- el trabajo se marcó como fallido. En ese caso, el incidente relacionado debe resolverse antes de que el trabajo pueda activarse nuevamente y completarse.

CreateWorkflowInstance RPC

Crea e inicia una instancia del flujo de trabajo especificado. La definición de flujo de trabajo que se utilizará para crear la instancia se puede especificar utilizando su clave única (tal como la devolvió DeployWorkflow) o utilizando el ID de proceso BPMN y una versión. Pase -1 como la versión para usar la última versión implementada.

Tenga en cuenta que solo los flujos de trabajo sin eventos de inicio pueden iniciarse mediante este comando.

Entrada: CreateWorkflowInstanceRequest

```
message CreateWorkflowInstanceRequest {
  // the unique key identifying the workflow definition (e.g. returned from a
workflow
  // in the DeployWorkflowResponse message)
  int64 workflowKey = 1;
  // the BPMN process ID of the workflow definition
  string bpmnProcessId = 2;
  // the version of the process; set to -1 to use the latest version
  int32 version = 3;
  // JSON document that will instantiate the variables for the root variable
scope of the
  // workflow instance; it must be a JSON object, as variables will be mapped in
a
  // key-value fashion. e.g. { "a": 1, "b": 2 } will create two variables, named
"a" and
  // "b" respectively, with their associated values. [{ "a": 1, "b": 2 }] would
not be a
  // valid argument, as the root of the JSON document is an array and not an
object.
  string variables = 4;
}
```

Salida: CreateWorkflowInstanceResponse

```
message CreateWorkflowInstanceResponse {
  // the key of the workflow definition which was used to create the workflow
instance
  int64 workflowKey = 1;
  // the BPMN process ID of the workflow definition which was used to create the
workflow
  // instance
  string bpmnProcessId = 2;
  // the version of the workflow definition which was used to create the
workflow instance
  int32 version = 3;
  // the unique identifier of the created workflow instance; to be used wherever
a request
  // needs a workflow instance key (e.g. CancelWorkflowInstanceRequest)
  int64 workflowInstanceKey = 4;
}
```

CreateWorkflowInstanceWithResult RPC

Similar a `CreateWorkflowInstance RPC`, crea e inicia una instancia del flujo de trabajo especificado. A diferencia `CreateWorkflowInstance RPC`, la respuesta se devuelve cuando se completa el flujo de trabajo.

Tenga en cuenta que solo los flujos de trabajo sin eventos de inicio pueden iniciarse mediante este comando.

Entrada: CreateWorkflowInstanceWithResultRequest

```
message CreateWorkflowInstanceRequest {
   CreateWorkflowInstanceRequest request = 1;
   // timeout in milliseconds. the request will be closed if the workflow is not
completed before
   // the requestTimeout.
   // if requestTimeout = 0, uses the generic requestTimeout configured in the
gateway.
   int64 requestTimeout = 2;
}
```

Salida: CreateWorkflowInstanceWithResultResponse

```
message CreateWorkflowInstanceResponse {
  // the key of the workflow definition which was used to create the workflow
instance
  int64 workflowKey = 1;
  // the BPMN process ID of the workflow definition which was used to create the
workflow
  // instance
  string bpmnProcessId = 2;
  // the version of the workflow definition which was used to create the
workflow instance
  int32 version = 3;
  // the unique identifier of the created workflow instance; to be used wherever
a request
  // needs a workflow instance key (e.g. CancelWorkflowInstanceRequest)
  int64 workflowInstanceKey = 4;
  // consisting of all visible variables to the root scope
  string variables = 5;
}
```

Errores

GRPC_STATUS_NOT_FOUND

Devuelto si:

- no existe flujo de trabajo con la clave dada (si se proporcionó workflowKey)
- no existe flujo de trabajo con el ID de proceso dado (si se proporcionó bpmnProcessId pero la versión fue -1)
- no existe flujo de trabajo con el ID y la versión de proceso dados (si se proporcionaron bpmnProcessId y la versión)

GRPC_STATUS_FAILED_PRECONDITION

Devuelto si:

- la definición del flujo de trabajo no contiene un evento sin inicio; solo los flujos de trabajo sin evento de inicio pueden iniciarse manualmente.

GRPC_STATUS_INVALID_ARGUMENT

Devuelto si:

- el argumento de las variables dadas no es un documento JSON válido; se espera que sea un documento JSON válido donde el nodo raíz sea un objeto.

DeployWorkflow RPC

Implementa uno o más flujos de trabajo en Zeebe. Tenga en cuenta que esta es una llamada atómica, es decir, o todos los flujos de trabajo están implementados o ninguno de

ellos lo está.

Entrada: DeployWorkflowRequest

```
message DeployWorkflowRequest {
  // List of workflow resources to deploy
  repeated WorkflowRequestObject workflows = 1;
}

message WorkflowRequestObject {
  enum ResourceType {
    // FILE type means the gateway will try to detect the resource type
    // using the file extension of the name field
    FILE = 0;
    BPMN = 1; // extension 'bpmn'
    YAML = 2; // extension 'yaml'
  }

  // the resource basename, e.g. myProcess.bpmn
  string name = 1;
  // the resource type; if set to BPMN or YAML then the file extension
  // is ignored
  ResourceType type = 2;
  // the process definition as a UTF8-encoded string
  bytes definition = 3;
}
```

Salida: DeployWorkflowResponse

```
message DeployWorkflowResponse {
  // the unique key identifying the deployment
  int64 key = 1;
  // a list of deployed workflows
  repeated WorkflowMetadata workflows = 2;
}

message WorkflowMetadata {
  // the bpmn process ID, as parsed during deployment; together with the version
forms a
  // unique identifier for a specific workflow definition
  string bpmnProcessId = 1;
  // the assigned process version
  int32 version = 2;
  // the assigned key, which acts as a unique identifier for this workflow
  int64 workflowKey = 3;
  // the resource name (see: WorkflowRequestObject.name) from which this
workflow was
  // parsed
  string resourceName = 4;
}
```

Errores

GRPC_STATUS_INVALID_ARGUMENT

Devuelto si:

- sin recursos dados
- si al menos un recurso no es válido Un recurso se considera inválido si:
  - no es un archivo BPMN o YAML (actualmente detectado a través de la extensión del archivo)
  - los datos del recurso no son deserializables (p. ej., se detectan como BPMN, pero tienen XML roto)
  - el flujo de trabajo no es válido (por ejemplo, una puerta de enlace basada en eventos tiene un flujo de secuencia saliente para una tarea)

FailJob RPC

Marca el trabajo como fallido; Si el argumento de reintentos es positivo, entonces el trabajo se podrá volver a activar inmediatamente, y un trabajador podría intentar nuevamente procesarlo. Sin embargo, si es cero o negativo, se generará un incidente, etiquetado con el mensaje de error dado, y el trabajo no se podrá activar hasta que se resuelva el incidente.

Entrada: FailJobRequest

```
message FailJobRequest {
  // the unique job identifier, as obtained when activating the job
  int64 jobKey = 1;
  // the amount of retries the job should have left
  int32 retries = 2;
  // an optional message describing why the job failed
  // this is particularly useful if a job runs out of retries and an incident is
raised,
  // as it this message can help explain why an incident was raised
  string errorMessage = 3;
}
```

Salida: FailJobResponse

```
message FailJobResponse {
}
```

Errores

GRPC_STATUS_NOT_FOUND

Devuelto si:

- no se encontró trabajo con la clave dada

GRPC_STATUS_FAILED_PRECONDITION

Devuelto si:

- el trabajo no se activó
- el trabajo ya está en un estado fallido, es decir, se quedó sin reintentos

PublishMessage RPC

Publica un solo mensaje. Los mensajes se publican en particiones específicas calculadas a partir de sus claves de correlación.

Entrada: solicitud

```
message PublishMessageRequest {
  // the name of the message
  string name = 1;
  // the correlation key of the message
  string correlationKey = 2;
  // how long the message should be buffered on the broker, in milliseconds
  int64 timeToLive = 3;
  // the unique ID of the message; can be omitted. only useful to ensure only
one message
  // with the given ID will ever be published (during its lifetime)
  string messageId = 4;
  // the message variables as a JSON document; to be valid, the root of the
document must be an
  // object, e.g. { "a": "foo" }. [ "foo" ] would not be valid.
  string variables = 5;
}
```

Salida: respuesta

```
message PublishMessageResponse {
}
```

Errores

GRPC_STATUS_ALREADY_EXISTS

Devuelto si:

- se publicó previamente un mensaje con el mismo ID (y todavía está vivo)

ResolverIncident RPC

Resuelve un incidente dado. Esto simplemente marca el incidente como resuelto; lo más probable es que se necesite una llamada a UpdateJobRetries o UpdateWorkflowInstancePayload para resolver realmente el problema, seguido de esta llamada.

Entrada: solicitud

```
message ResolveIncidentRequest {
  // the unique ID of the incident to resolve
  int64 incidentKey = 1;
}
```

Salida: respuesta

```
message ResolveIncidentResponse {
}
```

Errores

GRPC_STATUS_NOT_FOUND

Devuelto si:

- no existe ningún incidente con la clave dada

SetVariables RPC

Actualiza todas las variables de un ámbito particular (por ejemplo, instancia de flujo de trabajo, instancia de elemento de flujo) del documento JSON dado.

Entrada: solicitud

```
message SetVariablesRequest {
  // the unique identifier of a particular element; can be the workflow instance key (as
  // obtained during instance creation), or a given element, such as a service task (see
  // elementInstanceKey on the job message)
  int64 elementInstanceKey = 1;
  // a JSON serialized document describing variables as key value pairs; the root of the document
  // must be an object
  string variables = 2;
  // if true, the variables will be merged strictly into the local scope (as indicated by
  // elementInstanceKey); this means the variables is not propagated to upper scopes.
  // for example, let's say we have two scopes, '1' and '2', with each having effective variables as:
  // 1 => `{ "foo" : 2 }`, and 2 => `{ "bar" : 1 }`. if we send an update request with
  // elementInstanceKey = 2, variables `{ "foo" : 5 }`, and local is true, then scope 1 will
  // be unchanged, and scope 2 will now be `{ "bar" : 1, "foo" 5 }`. if local was false, however,
  // then scope 1 would be `{ "foo": 5 }`, and scope 2 would be `{ "bar" : 1 }`.
  bool local = 3;
}
```

Salida: respuesta

```
message SetVariablesResponse {
}
```

Errores

GRPC_STATUS_NOT_FOUND

Devuelto si:

- ningún elemento con lo dado `elementInstanceKey` era existe

GRPC_STATUS_INVALID_ARGUMENT

Devuelto si:

- la carga útil dada no es un documento JSON válido; Se espera que todas las cargas útiles sean documentos JSON válidos donde el nodo raíz es un objeto.

RPC de topología

Obtiene la topología actual del clúster del que forma parte la puerta de enlace.

Entrada: TopologyRequest

```
message TopologyRequest {
}
```

Salida: TopologyResponse

```
message TopologyResponse {
  // list of brokers part of this cluster
  repeated BrokerInfo brokers = 1;
  // how many nodes are in the cluster
  int32 clusterSize = 2;
  // how many partitions are spread across the cluster
  int32 partitionsCount = 3;
  // configured replication factor for this cluster
  int32 replicationFactor = 4;
}

message BrokerInfo {
  // unique (within a cluster) node ID for the broker
  int32 nodeId = 1;
  // hostname of the broker
  string host = 2;
  // port for the broker
  int32 port = 3;
  // list of partitions managed or replicated on this broker
  repeated Partition partitions = 4;
}

message Partition {
  // Describes the Raft role of the broker for a given partition
  enum PartitionBrokerRole {
    LEADER = 0;
    FOLLOWER = 1;
  }

  // the unique ID of this partition
  int32 partitionId = 1;
  // the role of the broker for this partition
  PartitionBrokerRole role = 2;
}
```

Errores

Sin errores específicos

UpdateJobRetries RPC

Actualiza el número de reintentos que le queda a un trabajo. Esto es principalmente útil para trabajos que se han quedado sin reintentos, en caso de que se resuelva el problema

subyacente.

Entrada: solicitud

```
message UpdateJobRetriesRequest {
  // the unique job identifier, as obtained through ActivateJobs
  int64 jobKey = 1;
  // the new amount of retries for the job; must be positive
  int32 retries = 2;
}
```

Salida: respuesta

```
message UpdateJobRetriesResponse {
}
```

Errores

GRPC_STATUS_NOT_FOUND

Devuelto si:

- no existe trabajo con la clave dada

GRPC_STATUS_INVALID_ARGUMENT

Devuelto si:

- reintentos no es mayor que 0