LIBRA
EVENT DRIVEN ENTERPRISE

Gestión de Eventos 6.5.0
29/08/2025

Gestión de Eventos 6.5.0

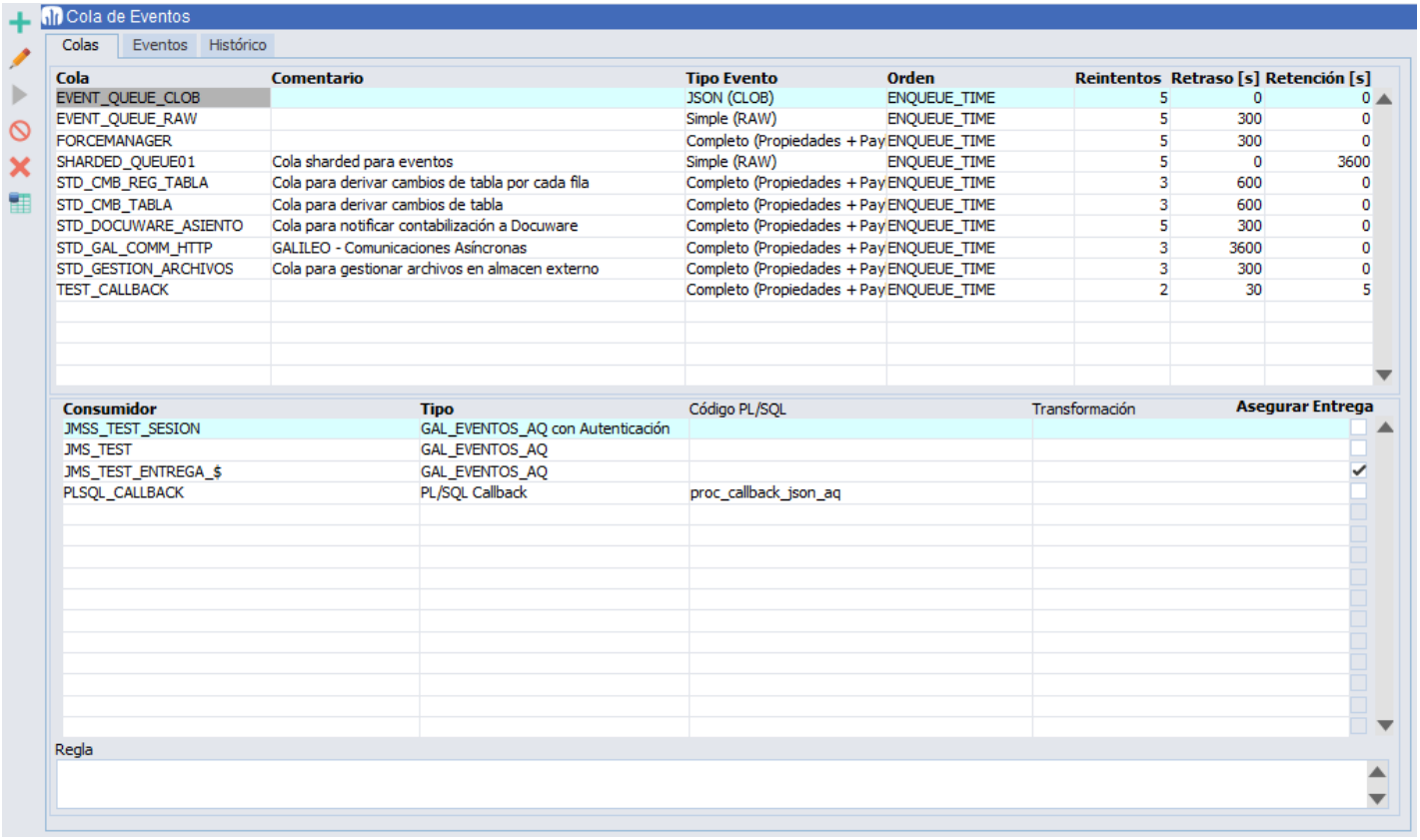
Introducción

Se ha implementado un sistema de registro de eventos que sigue el patrón “Publicador - Consumidores” mediante Oracle Advanced Queuing, que permite de forma sencilla registrar un evento al que se puedan registrar múltiples suscriptores, a quien serán entregados los eventos de forma asíncrona.

En este documento se detalla cómo se administran las colas de eventos y el código PL/SQL que realiza el registro de evento y su consumo.

Configuración

Desde el programa “Cola de Eventos” [U_EVT_AQ] se realiza la gestión de eventos y su monitorización.



Pestaña “Colas”

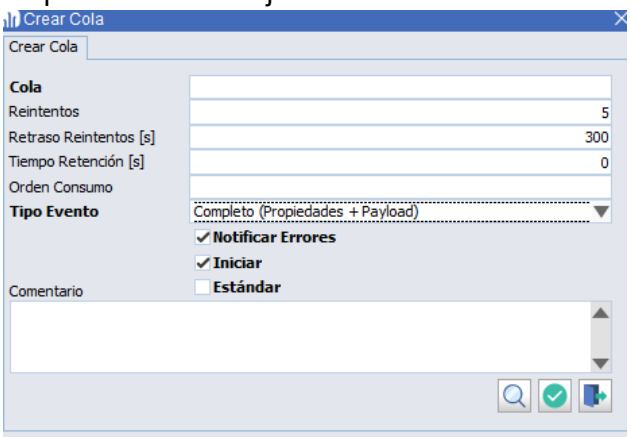
En esta pestaña se muestra un maestro-detalle de las colas existentes y sus suscripciones (procedimientos a los que serán redirigidos los eventos).
Estos multiregistros son bloques de sólo consulta, que tienen deshabilitado el borrado, inserción y edición, ya que tienen como origen de datos vistas relacionadas con las colas AQ de Oracle.
Para su gestión haremos uso de los Plug-Ins en la barra vertical desde la que podremos crear, borrar, arrancar, parar una cola, crear y borrar una suscripción y exportar dicha configuración mediante SQLs.

Crear Cola

Tras pulsar en el Plug-In “Crear Cola” se mostrará una ventana modal en la que solicita

- **Nombre:** Nombre de la cola [obligatorio]
- **Reintentos:** Número máximo de reintentos previo a enviar el evento a la cola de excepciones
- **Retraso Reintentos [s]**: Tiempo de espera hasta lanzar un nuevo reintento [en segundos]

- **Tiempo Retención [s]:** Tiempo que permanece en la cola un evento tras su correcto consumo [en segundos]
- **Orden Consumo:** Como se ordena la entrega de los eventos (Lista de valores)
- **Tipo Evento:** Las colas AQ permiten enviar una amplia variedad de tipos de datos como carga útil de eventos. Para facilitar su uso por parte de los desarrolladores, se ha implementado un subconjunto de formatos de mensaje:
    - **Tipo Simple (RAW)**: Permite enviar un valor de tipo VARCHAR2 como carga directa. Es un tipo adecuado para suscriptores Forms o JMS
    - **Tipo JSON (CLOB)**: Utiliza el tipo de dato AQ$_JSM_TEXT_MESSAGE para enviar contenido en formato JSON, almacenado como CLOB. Es un tipo adecuado para suscriptores Forms o JMS
    - **Tipo Completo (Propiedades + Payload):** Emplea el tipo AQ$_JSM_MAP_MESSAGE, que permite incluir tanto propiedades como datos de carga. Es el tipo recomendado para suscriptores PL/SQL Callback.
        - Las propiedades pueden utilizarse como filtros en la generación de suscriptores.
        - El payload admite múltiples tipos de datos PL/SQL: VARCHAR2, NUMBER, DATE, BOOLEAN, CLOB y BLOB.
- **Notificar Errores:** Registra una notificación cuando un evento termina en la cola de excepciones
- **Iniciar:** Iniciar la cola tras su creación
- **Estándar:** Check que indica si esta cola es o no estándar
- **Comentario:** Texto descripción sobre el objeto de esta cola



Suscribirse a una cola

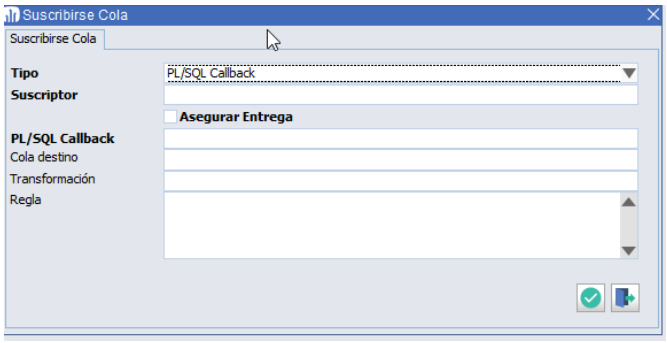
Tras pulsar en el Plug-In “Suscribirse” se mostrará una ventana modal en la que solicita

- **Tipo:** Cada evento encolado en AQ puede ser consumido por distintos tipos de suscriptores. Para facilitar su identificación, se utiliza un sistema de prefijos en el nombre del suscriptor, dependiendo del tipo de consumidor
    - **PL/SQL Callback**
        - **Descripción:** El evento se entrega automáticamente al procedimiento PL/SQL registrado como callback.

        - **Acceso a datos:** Se proporcionan métodos específicos para recuperar la información contenida en el evento.
        - **Prefijo:** *Sin prefijo*
    - **GAL_EVENTOS_AQ**
        - **Descripción:** Estos eventos son consumidos por el servicio GAL_EVENTOS_AQ.
        - **Con Autenticación:** Requiere el uso de un **TOKEN** durante la conexión al WebSocket, gestionado mediante el paquete PK_GAL_EVENTOS_AQ.
        - **Prefijos:**
            - JMS_ para eventos de tipo **GAL_EVENTOS_AQ**
            - JMSS_ para eventos de tipo **GAL_EVENTOS_AQ con Autenticación**
    - **Oracle Forms**
        - **Descripción:** Eventos destinados a ser consumidos por programas **Oracle Forms**.
        - **Prefijo:** FRM_
- **Suscritor:** Identificador del suscritor [obligatorio]
- **Asegurar Entrega:** Por defecto, el servicio ***gal_eventos_aq*** desencola los eventos de tipo **JMS** sin verificar si existe un cliente WebSocket conectado para recibirlos. Esto puede provocar que eventos se retiren de la cola sin haber sido entregados efectivamente. Para evitar esta situación, se ha incorporado una propiedad de **aseguramiento de entrega**, la cual, al activarse, garantiza que los eventos **solo serán desencolados si existe al menos un WebSocket conectado** y listo para recibir el contenido. Esta funcionalidad mejora la fiabilidad del sistema en escenarios donde la entrega en tiempo real es crítica.
- **PL/SQL Callback:** procedimiento al que se redirige el evento, cuya firma es:

  ```
  PROCEDURE callback(CONTEXT RAW, reginfo SYS.AQ$_REG_INFO, descr SYS.AQ$_DESCRIPTOR, payload RAW, payloadl NUMBER)
  ```
- **Cola Destino:** Propagar el evento a otra cola (Deshabilitado)
- **Transformación:** Método de transformación tras realizar el desencolado del evento.
- **Regla:** Es posible suscribirse a un evento que cumpla con un filtro relativo a las propiedades registradas en el objeto JMS entregado. Ejemplo:

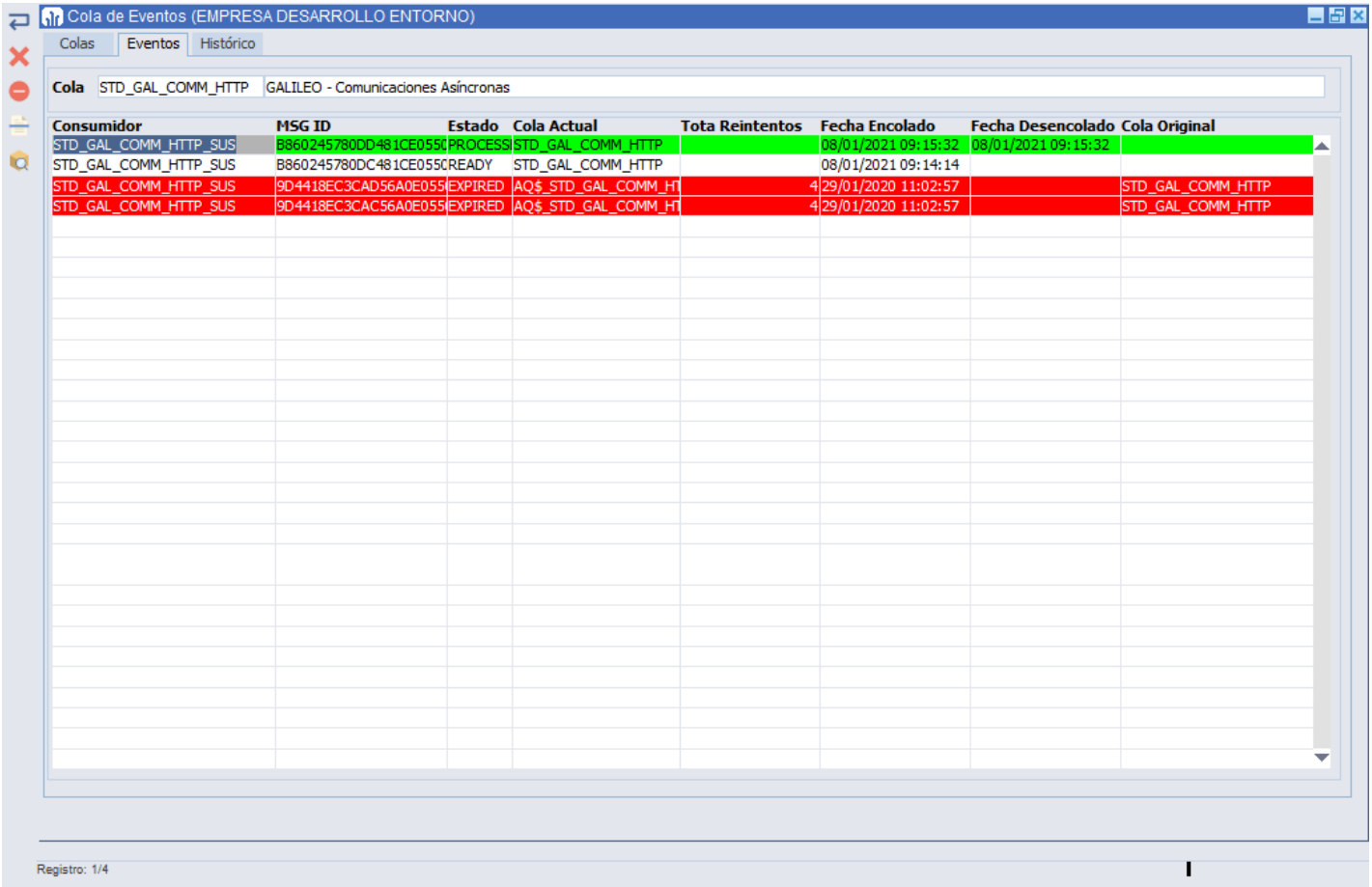
  ```
  tab.user_data.get_string_property('L$TABLA') = 'DIARIOS'
  ```



Pestaña “Eventos”

En esta pestaña se muestran los eventos encolados y el estado del consumo por sus suscriptores. Los eventos que han sido consumidos correctamente (no han terminado con una EXCEPTION) son retirados tras el tiempo de “Retención” especificado en la definición de la cola.

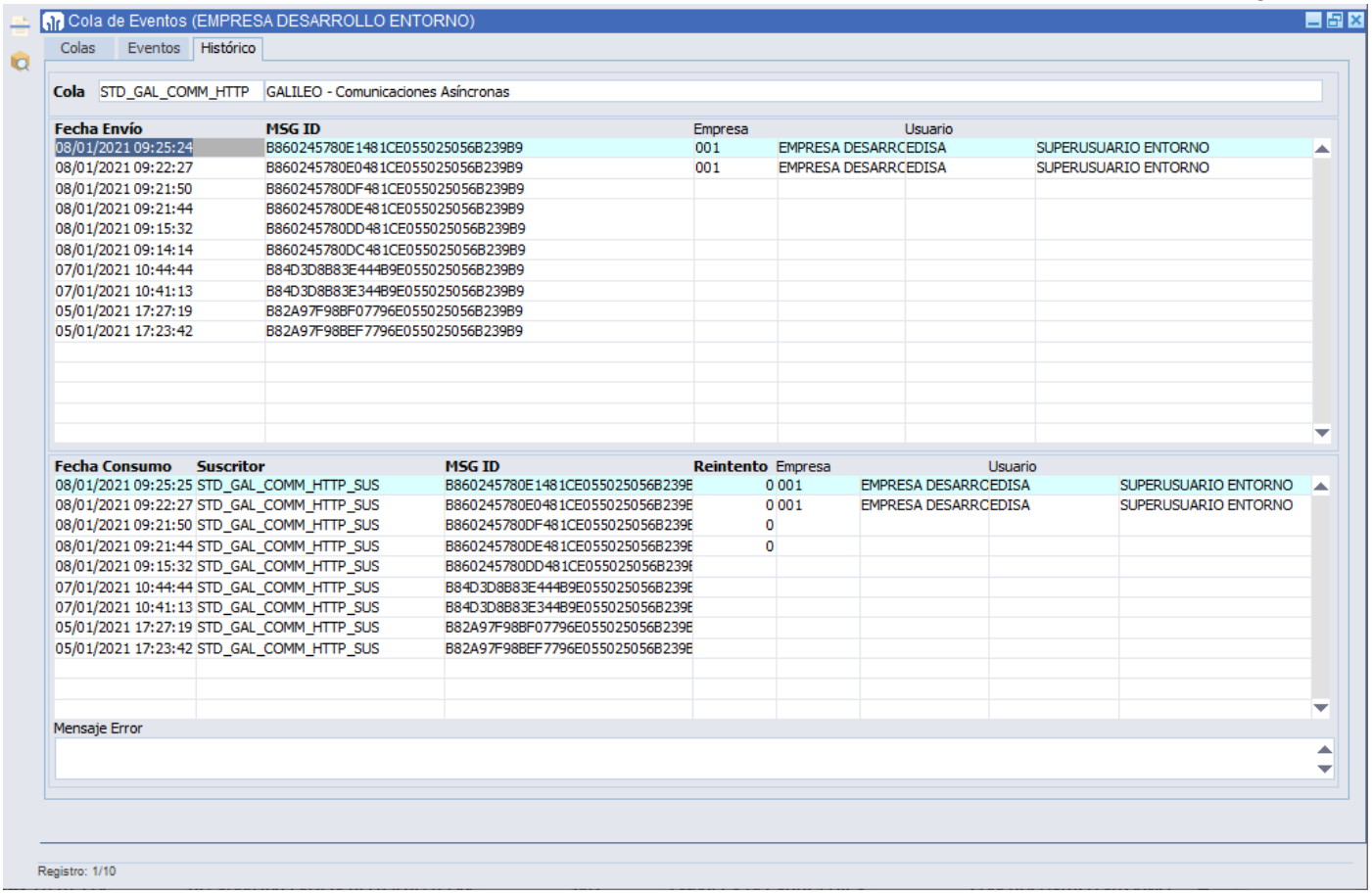
Aquellos que han fallado en su entrega y han consumido el número máximo de reintentos son enviados a la cola de excepción. Para su gestión se ha agregado una serie de Plug-Ins que permite: borrar un evento, reencolar un evento (que haya caído en la cola de excepción) y su purgado, así como consultar las propiedades y datos entregados en su payload.



Pestaña “Históricos”

En esta pestaña se muestran como multiregistros el histórico de eventos enviados y consumidos por sus suscriptores.

El motivo de su existencia reside en que tal y como se indica en el apartado anterior, los eventos que han sido consumidos son retirados de la cola, por lo que fue necesario crear unas tablas auxiliares donde registrarlos.



Código PL/SQL

En esta sección se indica el código a utilizar para comunicarnos con las colas registradas.
Para facilitar la gestión y centralizar un modo único de trabajo nos apoyaremos en el paquete ***"PK_GESTION_EVENTOS_AQ"*** el cual nos abstrae del código específico de gestión de colas mediante los paquetes "DBMS_AQ" y "DBMS_ADMAQ"

Publicar Evento

Según el tipo de cola, se utilizará un método de encole específico.

Publicar Evento Completo (AQ$_JMS_MAP_MESSAGE)

El envío de un evento a una cola se realiza mediante este método

```
FUNCTION encola_evento(p_cola VARCHAR2, p_propiedades IN OUT NOCOPY pk_gestion_eventos_aq_st.tt_propiedades, p_payload IN OUT NOCOPY pk_gestion_eventos_aq_st.tt_payload, p_immediate_enqueue BOOLEAN DEFAULT FALSE, p_wk_asegurar_orden BOOLEAN DEFAULT FALSE) RETURN RAW;
```

Dónde

- **p_cola:** Identificador de la cola a la que se envía el evento.
- **p_propiedades:** Array de propiedades. Permite filtrado por regla de suscripción.
  Para su registro, utilizar el PROCEDURE según corresponda.

```
PROCEDURE add_propiedad(t_propiedades IN OUT NOCOPY pk_gestion_eventos_aq.tt_propiedades, p_propiedad VARCHAR2, p_valor VARCHAR2);
PROCEDURE add_propiedad(t_propiedades IN OUT NOCOPY pk_gestion_eventos_aq.tt_propiedades, p_propiedad VARCHAR2, p_valor NUMBER);
PROCEDURE add_propiedad(t_propiedades IN OUT NOCOPY pk_gestion_eventos_aq.tt_propiedades, p_propiedad VARCHAR2, p_valor DATE);
PROCEDURE add_propiedad(t_propiedades IN OUT NOCOPY pk_gestion_eventos_aq.tt_propiedades, p_propiedad VARCHAR2, p_valor BOOLEAN);
```

- **p_payload:** Array que contiene los datos a entregar en el evento (NO permite filtrado).
  Para su registro, utilizar el PROCEDURE según corresponda.

```
PROCEDURE add_payload(t_payload IN OUT NOCOPY pk_gestion_eventos_aq.tt_payload, p_propiedad VARCHAR2, p_valor VARCHAR2);
PROCEDURE add_payload(t_payload IN OUT NOCOPY pk_gestion_eventos_aq.tt_payload, p_propiedad VARCHAR2, p_valor NUMBER);
PROCEDURE add_payload(t_payload IN OUT NOCOPY pk_gestion_eventos_aq.tt_payload, p_propiedad VARCHAR2, p_valor DATE);
PROCEDURE add_payload(t_payload IN OUT NOCOPY pk_gestion_eventos_aq.tt_payload, p_propiedad VARCHAR2, p_valor BOOLEAN);
PROCEDURE add_payload(t_payload IN OUT NOCOPY pk_gestion_eventos_aq.tt_payload, p_propiedad VARCHAR2, p_valor CLOB);
PROCEDURE add_payload(t_payload IN OUT NOCOPY pk_gestion_eventos_aq.tt_payload, p_propiedad VARCHAR2, p_valor BLOB);
```

- **p_immediate_enqueue:** Indica si los eventos se encolan de forma inmediata o requieren de un COMMIT posterior para su visibilidad
- **p_wk_asegurar_orden**: Se aplica un WORKAROUND que corrige un BUG por el cual los PL/SQL Callback no son entregado en ORDEN cuando se registran en el mismo segundo. Si se activa, el proceso agrega un SLEEP de 1s y a mayores un COMMIT si "p_immediate_enqueue" está desactivado.

Publicar Evento JSON (AQ$_JMS_TEXT_MESSAGE)

Método 1: Envío de un JSON en un dato CLOB

FUNCTION encola_evento_json(p_cola VARCHAR2, p_json_payload CLOB, p_registrar_variables_globales BOOLEAN DEFAULT FALSE, p_immediate_enqueue BOOLEAN DEFAULT FALSE, p_wk_asegurar_orden BOOLEAN DEFAULT FALSE) RETURN RAW;

Dónde

- **p_cola:** Identificador de la cola a la que se envía el evento.
- **p_json_payload:** Texto a enviar en el evento. Lo adecuado es que sea un JSON aunque nada impide enviar por ejemplo un **p_payload:** Array que contiene los datos a entregar en el evento (NO permite filtrado).
- **p_registrar_variables_globales**: Permite trasladar las variables globales de sesión en el CLOB de envio. Para ello generará un nuevo JSON con el formato
  "{"sesion_vars":{…},"payload":<p_json_payload>}"
  El método de consumo_evento_json se encarga de detectar este modo y aplicar las variables globales en la sesión donde se desencola.
- **p_immediate_enqueue:** Indica si los eventos se encolan de forma inmediata o requieren de un COMMIT posterior para su visibilidad
- **p_wk_asegurar_orden**: Se aplica un WORKAROUND que corrige un BUG por el cual los PL/SQL Callback no son entregado en ORDEN cuando se registran en el mismo segundo. Si se activa, el proceso agrega un SLEEP de 1s y a mayores un COMMIT si "p_immediate_enqueue" está desactivado.

Método 2: Envío de un JSON a partir de un payload

FUNCTION encola_evento_json(p_cola VARCHAR2, p_payload IN OUT NOCOPY pk_gestion_eventos_aq_st.tt_payload, p_registrar_variables_globales BOOLEAN DEFAULT FALSE, p_immediate_enqueue BOOLEAN DEFAULT FALSE, p_wk_asegurar_orden BOOLEAN DEFAULT FALSE) RETURN RAW;

Dónde

- **p_cola:** Identificador de la cola a la que se envía el evento.
- **p_payload:** Array que contiene los datos a entregar en el evento. Para su registro, utilizar el PROCEDURE según corresponda.

PROCEDURE add_payload(t_payload IN OUT NOCOPY pk_gestion_eventos_aq.tt_payload, p_propiedad VARCHAR2, p_valor VARCHAR2);
PROCEDURE add_payload(t_payload IN OUT NOCOPY pk_gestion_eventos_aq.tt_payload, p_propiedad VARCHAR2, p_valor NUMBER);
PROCEDURE add_payload(t_payload IN OUT NOCOPY pk_gestion_eventos_aq.tt_payload, p_propiedad VARCHAR2, p_valor DATE);

PROCEDURE add_payload(t_payload IN OUT NOCOPY pk_gestion_eventos_aq.tt_payload, p_propiedad VARCHAR2, p_valor BOOLEAN);
PROCEDURE add_payload(t_payload IN OUT NOCOPY pk_gestion_eventos_aq.tt_payload, p_propiedad VARCHAR2, p_valor CLOB);
PROCEDURE add_payload(t_payload IN OUT NOCOPY pk_gestion_eventos_aq.tt_payload, p_propiedad VARCHAR2, p_valor BLOB);

- **p_registrar_variables_globales**: Permite trasladar las variables globales de sesión en el CLOB de envio. Para ello generará un nuevo JSON con el formato
  "{"sesion_vars":{…},"json_data":<p_json_payload>}"
  El método de consumo_evento_json se encarga de detectar este modo y aplicar las variables globales en la sesión donde se desencola.
- **p_immediate_enqueue:** Indica si los eventos se encolan de forma inmediata o requieren de un COMMIT posterior para su visibilidad
- **p_wk_asegurar_orden**: Se aplica un WORKAROUND que corrige un BUG por el cual los PL/SQL Callback no son entregado en ORDEN cuando se registran en el mismo segundo. Si se activa, el proceso agrega un SLEEP de 1s y a mayores un COMMIT si "p_immediate_enqueue" está desactivado.

Publicar Evento RAW

El envío de un evento a una cola se realiza mediante este método

FUNCTION encola_evento_raw(p_cola VARCHAR2, p_dato VARCHAR2, p_registrar_variables_globales BOOLEAN DEFAULT FALSE, p_immediate_enqueue BOOLEAN DEFAULT FALSE, p_wk_asegurar_orden BOOLEAN DEFAULT FALSE) RETURN RAW;

Dónde

- **p_cola:** Identificador de la cola a la que se envía el evento.
- **p_dato:** Texto a enviar en el evento.
- **p_registrar_variables_globales**: Permite trasladar las variables globales de sesión en el CLOB de envio. Para ello generará un nuevo JSON con el formato
  "{"sesion_vars":{…},"json_data":<p_json_payload>}"
  El método de consumo_evento_json se encarga de detectar este modo y aplicar las variables globales en la sesión donde se desencola.
- **p_immediate_enqueue:** Indica si los eventos se encolan de forma inmediata o requieren de un COMMIT posterior para su visibilidad
- **p_wk_asegurar_orden**: Se aplica un WORKAROUND que corrige un BUG por el cual los PL/SQL Callback no son entregado en ORDEN cuando se registran en el mismo segundo. Si se activa, el proceso agrega un SLEEP de 1s y a mayores un COMMIT si "p_immediate_enqueue" está desactivado.

Consumir Evento

Las colas de eventos “Oracle Advanced Queueing” permiten redirigir un evento a un PROCEDURE que cumpla la siguiente firma

```
PROCEDURE callback(CONTEXT RAW, reginfo SYS.AQ$_REG_INFO, descr SYS.AQ$_DESCRIPTOR, payload RAW, payloadl NUMBER)
```

Dentro de este PROCEDURE el desarrollador tiene que lanzar el código que realiza el “desencolado” del evento y extracción de las propiedades registradas en su envío.

Para facilitar el desarrollo y evitar posibles errores, el paquete “***PK_GESTION_EVENTOS_AQ***” incluye los métodos “***callback_plantilla***”, “***callback_plantilla_json***” y “***callback_plantilla_raw***”; que sirvan de punto de para a continuación continuar con el proceso del dato entregado en el registro del evento, según el tipo de cola configurada (Completo, JSON y RAW respectivamente).

```
PROCEDURE callback(CONTEXT RAW, reginfo SYS.AQ$_REG_INFO, descr SYS.AQ$_DESCRIPTOR, payload RAW, payloadl NUMBER) IS
  PRAGMA AUTONOMOUS_TRANSACTION;
  v_valor_evento  CLOB;
  vt_propiedades  pk_gestion_eventos_aq.tt_propiedades;
  vt_payload      pk_gestion_eventos_aq.tt_payload;
BEGIN
  --Consumir el evento -> Extrae el PAYLOAD del evento, registra las variables globales y retorna las propiedades
  pk_gestion_eventos_aq.consumir_evento(CONTEXT => CONTEXT, reginfo => reginfo, descr => descr, payload => payload, payloadl => payloadl, p_propiedades => vt_propiedades, p_payload => vt_payload);

  --Continuar a partir de aquí
  --
  --.....
  --

  --Finalizar con COMMIT!!!
  COMMIT;
EXCEPTION
  WHEN OTHERS THEN
    --Registrar consumo del evento tabla de histórico
    pk_gestion_eventos_aq.registrar_consumo_evento(p_empresa => pkpantallas.get_empresa, p_usuario => pkpantallas.usuario_validado, p_cola => descr.queue_name, p_suscritor => descr.consumer_name, p_msg_id => descr.msg_id, p_reintento => descr.msg_prop.attempts, p_mensaje_error => SQLERRM);
    ROLLBACK;
    RAISE;
END callback;
```

Eventos estándar

Con el entorno se distribuyen colas estándar las cuales gestionan los siguientes eventos.

Comunicaciones HTTP asíncronas

El paquete pk_galileo permite registrar una petición HTTP (mediante pk_galileo.tr_peticion) de forma asíncrona, a través de la cola estandar “STD_GAL_COMM_HTTP”.

Esta cola tiene la particularidad de que la función PL/SQL CALLBACK está suscrita al propio paquete, el cual se encarga del desencolado y comunicación HTTP, permitiendo redirigir la respuesta a otra función PL/SQL.

Las funciones que se encargan para el registro de estos eventos son las siguientes.

FUNCTION aq_encola_peticion_http(p_peticion tr_peticion, p_plsql_callback VARCHAR2 DEFAULT NULL) RETURN RAW;
FUNCTION aq_encola_peticion_http_at(p_peticion tr_peticion, p_plsql_callback VARCHAR2 DEFAULT NULL) RETURN RAW;
PROCEDURE aq_encola_peticion_http(p_peticion tr_peticion, p_plsql_callback VARCHAR2 DEFAULT NULL);
PROCEDURE aq_encola_peticion_http_at(p_peticion tr_peticion, p_plsql_callback VARCHAR2 DEFAULT NULL);

Propagar cambios en tablas

La cola “STD_CMB_TABLA” gestiona los eventos de cambios de una tabla.
Este evento usualmente se registra desde un TRIGGER el cual está a nivel de cambios de tabla.

Para facilitar su creación se debe utilizar este método

```
pk_gestion_eventos_aq.snippet_trigger_cambios_tabla(p_nombre_tabla VARCHAR2, p_nombre_corto VARCHAR2 DEFAULT NULL) RETURN CLOB;
```

Propagar cambios en tablas a nivel de row

La cola “STD_CMB_REG_TABLA” gestiona los eventos de cambios de una tabla a nivel de registro.
Para este tipo de eventos tenemos dos formas de gestionarlos.

Registro evento inmediato

Si la tabla no suele tener mucho volumen de cambios, el evento se puede registrar desde el TRIGGER del tipo de cambio de fila.

Para facilitar su creación se debe utilizar este método

```
pk_gestion_eventos_aq.snippet_trigger_cambios_reg(p_nombre_tabla VARCHAR2, p_nombre_corto VARCHAR2 DEFAULT NULL, p_solo_pk VARCHAR2 DEFAULT 'N', p_usar_payload VARCHAR2 DEFAULT 'N') RETURN CLOB;
```

Registro evento diferido

Si la tabla suele tener muchos cambios simultáneos que involucran la modificación de varias filas simultáneas y debido a que el registro de cada evento es una tarea pesada, se recomienda utilizar el método registro en diferido, el cual consta de la creación de dos TRIGGERs.

- TRIGGER a nivel de fila: Registra los datos del evento como un evento en diferido
- TRIGGER a nivel de tabla: Itera por los eventos registrados en diferido y los inserta en la cola.

Para facilitar su creación se debe utilizar este método

```
pk_gestion_eventos_aq.snippet_triggers_cmb_reg_dif(p_nombre_tabla VARCHAR2, p_nombre_corto VARCHAR2 DEFAULT NULL, p_solo_pk VARCHAR2 DEFAULT 'N', p_usar_payload VARCHAR2 DEFAULT 'N') RETURN CLOB;
```

POSIBLES MEJORAS

Integrar la gestión de eventos a nivel de trigger con AUDITORIA, de forma que se puedan registrar el trigger a nivel de columnas a controlar y el poder indicar qué columnas van como property(permiten filtrar) y cuales como payload

Servicio GAL_EVENTOS_AQ

El servicio **GAL_EVENTOS_AQ** permite entregar eventos desde una cola AQ de Oracle a una aplicación web mediante WebSocket, como por ejemplo un módulo de **LIBRA Movilidad**.

Para utilizarlo, basta con instalar el servicio y añadir un suscriptor del tipo **GAL_EVENTOS_AQ** a la cola correspondiente. Este suscriptor puede configurarse para requerir credenciales de conexión, lo que garantiza que los eventos críticos no se entreguen a aplicaciones no autorizadas que simplemente capturen la URL del WebSocket.

Para facilitar el acceso a la URL del WebSocket, se ha desarrollado el paquete **PK_GAL_EVENTOS_AQ**, que incluye el método get_url_websocket. Este método genera automáticamente una URL que incorpora un **token de un solo uso** para autenticación, y selecciona el protocolo adecuado (ws o wss) según si el servicio está instalado con SSL o si la URL pública utiliza https.

FUNCTION get_url_websocket(p_empresa VARCHAR2, p_usuario VARCHAR2, p_cola VARCHAR2, p_consumidor VARCHAR2) RETURN VARCHAR2

Recuperar motor AQ caído

En ocasiones hemos tenido caídas del motor de eventos AQ el cual en sólo conseguimos recuperar tras un reinicio de la BBDD.

Nota

Hay que tener en cuenta que las colas AQ internamente utilizan JOBS de BBDD en el usuario SYS, por lo que cualquier configuración de la BBDD que tenga los JOBS parados (por ejemplo job_queue_procesess = 0) evitará que se propaguen los eventos.

Tras abrir un caso en soporte, nos han facilitado las siguientes instrucciones

La diferencia entre el método 1 y 2 es que en el primero se trata de realizar parada controlada de los schedulers mientras que el segundo implica matar sesiones.

Método 1

1. Cambiar el parámetro “aq_tm_processes” a 0
   conn / as sysdba
   alter system set aq_tm_processes = 0;

2. Detener los SCHEDULER_JOB relacionados con AQ

```
DECLARE
  CURSOR cur_aq IS
    SELECT job_name
      FROM dba_scheduler_running_jobs
     WHERE job_name LIKE 'AQ$_%';
BEGIN
  FOR reg IN cur_aq LOOP
    dbms_scheduler.stop_job(reg.job_name);
  END LOOP;
END;
```

3. Volver a dejar el parámetro “aq_tm_processes” a 1

Método 2

1. Cambiar el parámetro “aq_tm_processes” a 0
   conn / as sysdba
   alter system set aq_tm_processes = 0;

2. Matar todos los procesos AQ (q01,q02,…) en el SO o esperar a que finalicen.

3. Volver a dejar el parámetro “aq_tm_processes” a 1

Método 3

Ejecutar script que vuelca los datos de la BBDD a un HTML
https://support.oracle.com/epmos/faces/DocumentDisplay?parent=SrDetailText&sourceId=3-30462384631&id=1193854.1
https://support.oracle.com/epmos/faces/DocumentDisplay?_afrLoop=35064435889613&parent=SrDetailText&sourceId=3-30462384631&id=1958196.1&_afrWindowMode=0&_adf.ctrl-state=186a36y16z_70

En caso de que alguno de estos parámetros tenga estos valores continuar

Parámetro	**Valor no válido**
_client_enable_auto_unregister	FALSE
streams_pool_size	0

Pasos a realizar

- connect / as sysdba
- alter system set "_client_enable_auto_unregister"=true scope=spfile ;
- shutdown immediate
- startup

Asignar un valor al parámetro STEAMS_POOL_SIZE, por ejemplo: 50M
El valor adecuado resultaría en el valor obtenido por esta SQL

```
SELECT component, current_size/1048576, max_size/1048576
   FROM v$sga_dynamic_components
WHERE component='streams pool';
```

Bibliografía

https://docs.oracle.com/en/database/oracle/oracle-database/12.2/arpls/DBMS_AQ.html
https://docs.oracle.com/en/database/oracle/oracle-database/12.2/arpls/DBMS_AQADM.html

EDISA

LATINOAMÉRICA

COLOMBIA
ECUADOR
MÉXICO
REP. DOMINICANA

ESPAÑA

MADRID
BARCELONA
VALENCIA
VIGO
OVIEDO
LAS PALMAS
OURENSE (CENTRO I+D)