**HCL Commerce V9.1**

**Native Kafka Connector Integration Developer Guide**

**Draft - Version 1.0**

**January 19, 2023**

****

**Document Source**

|  |  |  |  |
| --- | --- | --- | --- |
| Version # | Date | Version Description and Summary of changes | Editor |
| 1.0 | January19, 2023 | Draft- First version of review | Anuj Kumar |

**Setup the Kafka**

1. Install the Kafka
2. Go to installation directory of Kafka
3. Launch Zookeeper with: **.\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties**
4. Open a second command line in your Kafka installation folder
5. Launch single Kafka broker: **.\bin\windows\kafka-server-start.bat .\config\server.properties**
6. Open a third command line in your Kafka installation folder
7. Create a topic for producer for testing purpose: **.\bin\windows\kafka-topics.bat --create --bootstrap-server kafka:9092 --replication-factor 1 --partitions 1 --topic HCLC\_ORDER**
8. Create a topic for registry consumer for testing purpose: **.\bin\windows\kafka-topics.bat --create --bootstrap-server kafka:9092 --replication-factor 1 --partitions 1 --topic HCLC\_INV**
9. Create a topic for scheduler consumer for testing purpose: **.\bin\windows\kafka-topics.bat --create --bootstrap-server kafka:9092 --replication-factor 1 --partitions 1 --topic HCLC\_INV\_SCH**
10. Open a fourth command line in your Kafka installation folder for consuming the message from the topic this consumer will consume the OrderItemAdd event data in this PoC: **.\bin\windows\kafka-console-consumer.bat --bootstrap-server kafka:9092 --topic HCLC\_ORDER**
11. Open a fifth command line in your Kafka installation folder for producing the message to Topic that will be consumed by Our InventoryConsumer class in this POC: **.\bin\windows\kafka-console-producer.bat --broker-list kafka:9092 –topic HCLC\_INV**
12. Open a sixth command line in your Kafka installation folder for producing the message to Topic that will be consumed by Our Scheduler class in this POC: **.\bin\windows\kafka-console-producer.bat --broker-list kafka:9092 –topic HCLC\_INV\_SCH**
13. **Note:** In this example we have Kafka hostname as “**Kafka**” because we have put hostname as kaka in Kafka config files. So if yours is different then use that one for example if your Kafka host is localhost then use localhost:port as bootstrap server.
14. In case if you are running Kafka docker container then in order to connect from ts-app container to Kafka container we need to do following config changes and where hostname will be Kafka and port is 9092.

Login inside Kafka docker container and go to following location and search for **bootstrap.servers** and change its value as below.

Filename: ./config/producer.properties

PropertyToBeModified: bootstrap.servers=kafka:9092

Login inside Kafka docker container and go to following location and search for **listeners** and change its value as below.  
Filename: ./config/server.properties

PropertyToBeModified: listeners=PLAINTEXT://kafka:9092

Login inside Kafka docker container and go to following location and search for **advertised.listeners** and change its value as below  
Filename : ./config/server.properties

PropertyTobeModified: advertised.listeners=PLAINTEXT://kafka:9092

Login inside Kafka docker container and go to following location and search for **bootstrap.servers** and change its value as below.  
Filename : ./config/consumer.properties

PropertyToBeModified : bootstrap.servers=kafka:9092

Login inside Kafka docker container and go to following location and search for **listeners** and change its value as below.  
FileName : ./config/kraft/server.properties

PropertyToBeModified : listeners=PLAINTEXT://kafka:9092,CONTROLLER://:9093

Login inside Kafka docker container and go to following location and search for **advertised.listeners** and change its value as below.  
FileName : ./config/kraft/server.properties

PropertyToBeModified : advertised.listeners=PLAINTEXT://kafka:9092

Login inside Kafka docker container and go to following location and search for **listeners** and change its value as below.  
FileName : ./config/kraft/broker.properties

PropertyToBeModified : listeners=PLAINTEXT://kafka:9092

Login inside Kafka docker container and go to following location and search for **advertised.listeners** and change its value as below.  
FileName : ./config/kraft/broker.properties

PropertyToBeModified : advertised.listeners=PLAINTEXT://kafka:9092

**Note :** Above steps is for windows if you have installed Kafka in windows if it is installed on Linux system then just go to the path : **.\bin\** and here you will find the .sh commands and now follow same 1 to 12 steps as above.

**Integration Steps**

Once you have approval to integrate the Native Kafka Connector. HCL provide the Native Kafka Connector folder. This folder contains following projects-

1. WebSphereCommerceServerExtensionsLogic
2. WC
3. Stores
4. Dataload
5. LOBTools
6. tooling-web
7. **Deploy WebSphereCommerceServerExtensionsLogic**
   1. Copy the Java Class classes from WebSphereCommerceServerExtensionsLogic/src and place it on to your {HCL\_Commerce\_Workspace}/ WebSphereCommerceServerExtensionsLogic/src folder

Graphical user interface, text, application, email

Description automatically generated

1. **Deploy WC Changes**
   1. Merge the we-server.xml file from WC/xml/config to your {HCL\_Commerce\_Workspace}/WC/xml/config folder. The following code needs to be merged to create Registry Class.

|  |
| --- |
| **<registry** name="KafkaConsumerRegistry" regClassName="com.hcl.commerce.kafka.consumer.KafkaConsumerRegistry"**/>** |

Graphical user interface, text, application

Description automatically generated

* 1. If com.ibm.commerce.foundation-ext folder does not exist on your installation directory then create this com.ibm.commerce.foundation-ext folder on your {HCL\_Commerce\_Workspace}WC/xml/config installation directory and copy files from WC/xml/config/com.ibm.commerce.foundation-ext to your created folder

Else Do following

* 1. Merge the wc-component.xml file from WC/xml/config/com.ibm.commerce.foundation-ext to your {HCL\_Commerce\_Workspace}WC/xml/config/com.ibm.commerce.foundation-ext. The following changes needs to be merged.

**Note**: In this code snippet we have placed encrypted passwords of wcsadmin user. So, you need to first encrypt the password for user using wcs\_encrypt.bat and then place that encrypted password here.

|  |
| --- |
| **<\_config:configgrouping** name="kafkaConfig"**>**  **<\_config:property** name="com.hcl.kafka.consumer.inventory.topic" value="HCLC\_INV"**/>**  **<\_config:property** name="com.hcl.kafka.producer.order.topic" value="HCLC\_ORDER"**/>**  **<\_config:property** name="enableRestCall" value="true"**/>**  **<\_config:property** name="enableJdbcCall" value="false"**/>**  **</\_config:configgrouping>**  **<\_config:configgrouping** name="RestConfig"**>**  **<\_config:property** name="username" value="wcsadmin"**/>**  **<\_config:property** name="password" value="BiJRGDv39K6N85I4ANQpmrf5WwR/DCGOUuq6Uw5OeX4="**/>**  **</\_config:configgrouping>**  **<\_config:configgrouping** name="SchedulerConfig"**>**  **<\_config:property** name="com.hcl.kafka.consumer.inventory.topic" value="HCLC\_INV\_SCH"**/>**  **</\_config:configgrouping>** |

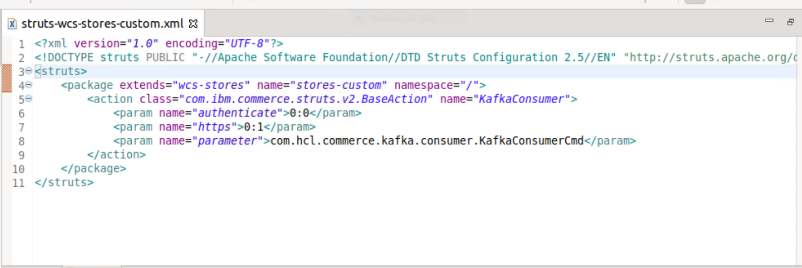


1. **Deploy Stores**
   1. If struts-wcs-stores-custom.xml file does not exist on your installation directory then create this struts-wcs-stores-custom.xml file on your {HCL\_Commerce\_Workspace}Stores/src installation directory and copy files from Stores/src to your directory.

Else Do following

* 1. Merge the struts-wcs-stores-custom.xml file from Stores/src to your {HCL\_Commerce\_Workspace} Stores/src the following changes needs to be merged.

|  |
| --- |
| **<action** class="com.ibm.commerce.struts.v2.BaseAction" name="KafkaConsumer"**>**  **<param** name="authenticate"**>**0:0**</param>**  **<param** name="https"**>**0:1**</param>**  **<param** name="parameter"**>**com.hcl.commerce.kafka.consumer.KafkaConsumerCmd**</param>**  **</action>** |



1. **Deploy DataLoad**
   1. Execute the SQL available inside Dataload/Sql to enable or disable the producer. We need to make entry for each new consumer and producer class in STORECONF. So that these consumer and producer will be available in CMC and from there we can make them enable and disable.
   2. Make sure that each entry of producer and consumer in STORECONF TABLE must have start with this prefix as given below.

**Example:** producer prefix “kafka.producer.{nameofproducer}” and consumer prefix “kafka.consumer.{nameofconsumer}”

1. **Deploy LOBTools**
   1. Merge the properties file from LOBTools/src/com/ibm/commerce/foundation/client/lobtools/properties to your {HCL\_Commerce\_Workspace} LOBTools/src/com/ibm/commerce/foundation/client/lobtools/properties/. The following changes needs to be merged.

|  |
| --- |
| integrationsManagementDisplayName=Integrations |

* 1. Merge the ApplicationMenuItems .xml file from LOBTools/WebContent/WEB-INF/src/xml/commerce/shell to your {HCL\_Commerce\_Workspace} LOBTools/WebContent/WEB-INF/src/xml/commerce/shell. The following changes needs to be merged

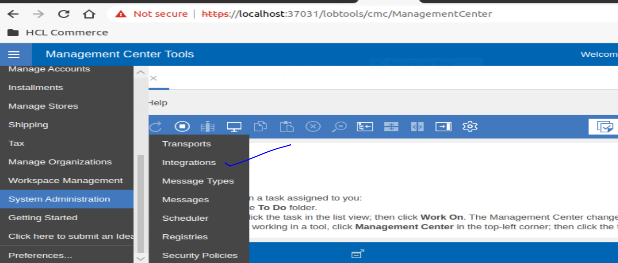
|  |
| --- |
| **<ApplicationMenuItem** actionName="openBusinessObjectEditor" displayName="${shellResources.integrationsManagementDisplayName}" id="integrationsManagement" package="cmc/shell" toolDefinition="cmc/integrations/IntegrationsManagement"**>**  **<EnablementAndCondition>**  **<DarkFeatureEnablementCondition** featureName="integrationsManagement" **/>**  **<EnablementOrCondition>**  **<EnablementCondition** contextName="siteAdministrator" enablementValue="true" **/>**  **</EnablementOrCondition>**  **</EnablementAndCondition>**  **</ApplicationMenuItem>** |

* 1. Copy the integration folder from /LOBTools/WebContent/WEB-INF/src/xml/commerce to your {HCL\_Commerce\_Workspace} /LOBTools/WebContent/WEB-INF/src/xml/commerce.

1. **Deploy tooling-web**
   1. Copy the **integration** folder from /tooling-web/commerce-tooling /src/app/features to your {HCL\_Commerce\_Workspace} /tooling-web/commerce-tooling /src/app/features.
   2. Merge the app-routing.module.ts file from /tooling-web/commerce-tooling /src/app to your {HCL\_Commerce\_Workspace} /tooling-web/commerce-tooling /src/app. Following code needs to be merged.
2. Now Build these changes and deploy on your local environment.

**Test the producer and consumer communication with kafka server.**

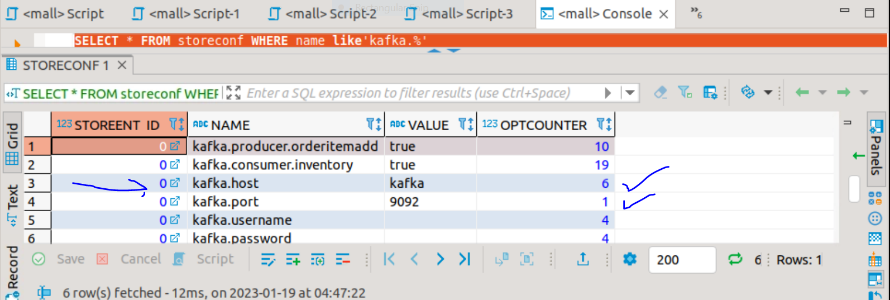
So Let`s test first Producer Example. We will be publishing the OrderItems details on Kafka topic. So whenever item gets added to the cart means after successful post processing of orderitemadd we are publishing the message of orderitems details on Kafka topic. Before that we need setup Kafka server details from CMC for producer and consumer to connect with Kafka.

1. **Setup the Kafka configuration via CMC**
   1. Go to CMC click on System Administration->Integrations this following page will be open.  
        
      

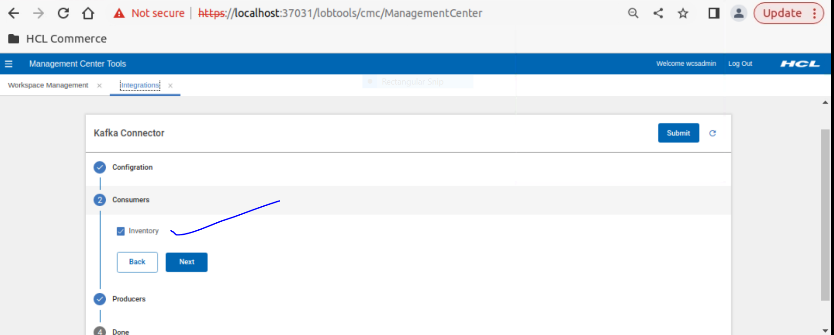
Graphical user interface, application

Description automatically generated

* 1. Enter the Kafka server details and submit it. Once submitted you can see entries will be saved into STORECONF table as below.



* 1. Also, you can see the consumer and producer tab from there you can enable or disable the consumer and producer. We have one producer and consumer details in STORECONF table that`s why we can see one producer and consumer in dropdown.

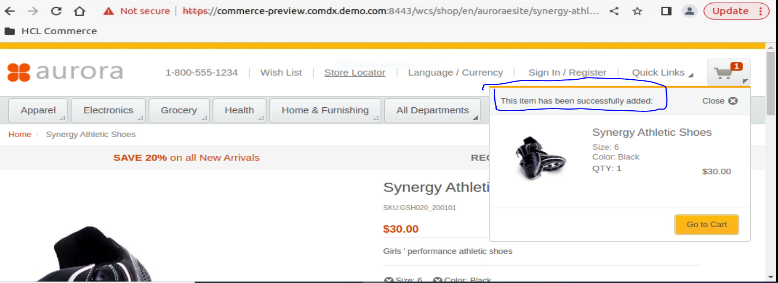
****

**Graphical user interface, application

Description automatically generated**

1. **Publish Message on Kafka Topic**.
   1. Go to PDP page of any product and click on add to cart button. Once an item has been added to the cart after that message has been published to the Kafka Topic to verify that switch to the command line terminal that we have opened in step 10 in Setup Kafka.

Added item to Cart.



Message Consumed from Kafka topic post orderitem added to cart.

Text

Description automatically generated

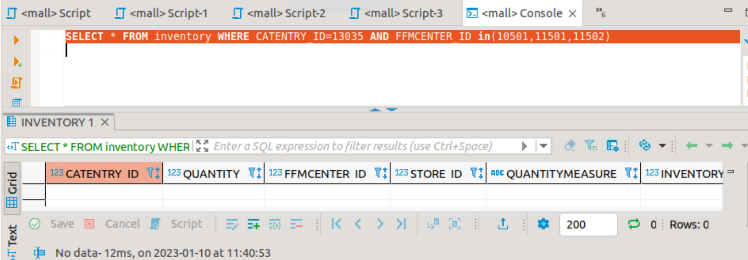
1. **Consume Message from Kafka Topic.**

For this consumer example we have taken an example of Inventory. We will be publishing the inventory record on Kafka topic from command line and our consumer class will consume that message and will save the inventory record into the INVENTORY table of the database.

For this consumer PoC we have two approaches using which we can start the consumer class that will keep consuming the data whenever a publish event is triggered on Kafka topic.

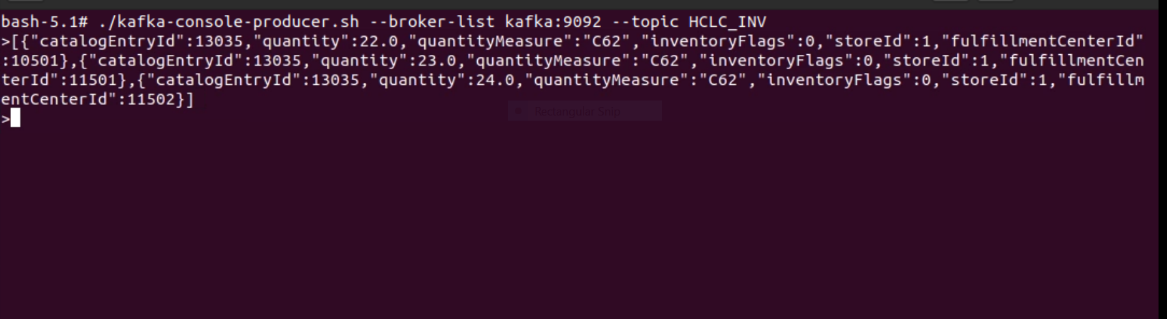
* 1. **Consume message using Registry class approach**.
     1. So, under this registry approach we are initializing the consumer class at the server start time. So, whenever server will be started at that time our registry class will create the Kafka server connection to consume the message from Kafka topic whenever publish event will be triggered on topic and this Kafka server connection will be alive until something misshapen.
     2. Now once we have consumed message from Kafka topic under our consumer registry class, we need to save that consumer record into inventory table. To save that consumer record we are using two approaches to save data into INVENTORY table as below.
        1. First, we are calling inventory Rest API to save that consumer record into INVENTORY table. For this example, we have enabled Rest API call to save consumer record.
        2. Second, we are using JDBCHelper Access Bean to save the consumer record into INVENTORY table if flag is enabled.
     3. Now let`s verify the consumer go to the Kafka producer terminal that we have opened in step no 11 under Setup Kafka and publish the below message.

As you can see INVENTORY table has no record for CatentryId 13035 and ffmcenter (10501,11501,11502)

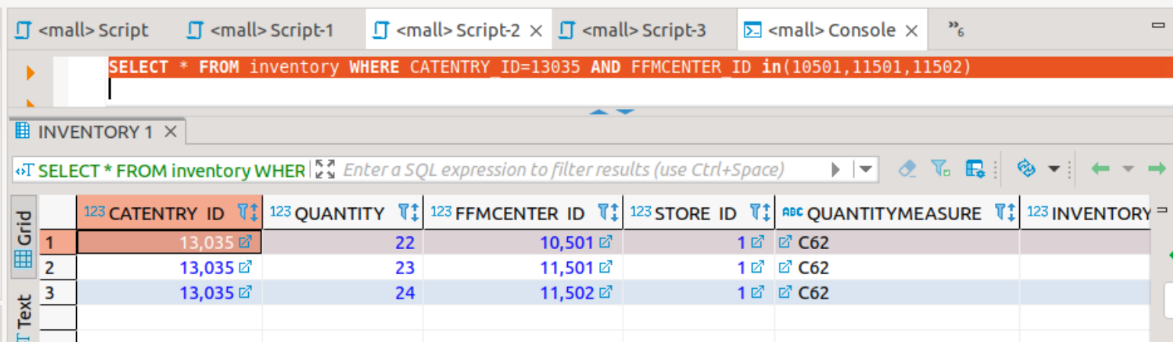


Now from producer terminal we are going to publish following inventory record in Kafka Topic. Change inventory accordingly.

|  |
| --- |
| [{"catalogEntryId":13035,"quantity":22.0,"quantityMeasure":"C62","inventoryFlags":0,"storeId":1,"fulfillmentCenterId":10501},{"catalogEntryId":13035,"quantity":23.0,"quantityMeasure":"C62","inventoryFlags":0,"storeId":1,"fulfillmentCenterId":11501},{"catalogEntryId":13035,"quantity":24.0,"quantityMeasure":"C62","inventoryFlags":0,"storeId":1,"fulfillmentCenterId":11502}] |



Now once we have published the message of kafka topic. We can see INVENTORY table has been populated with same data.

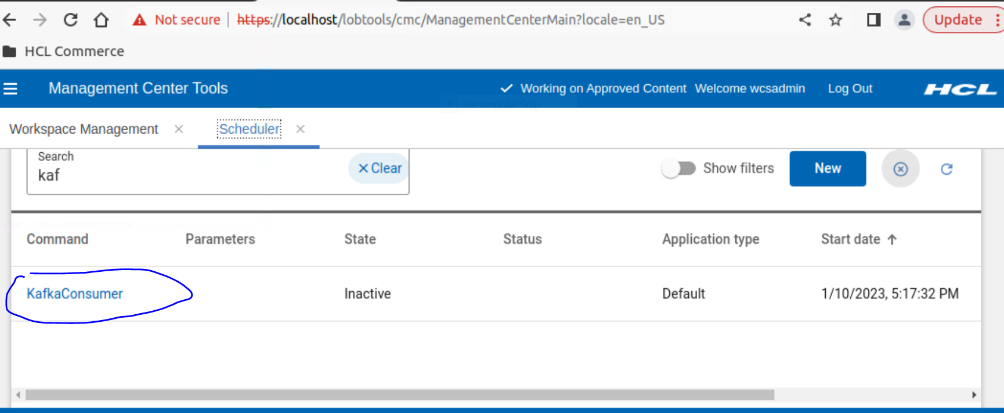


To save the consumer record from our Kafka Consumer class we are using Rest API call for this example as you can see we have enabled Rest API call and JDBC call is disabled. You can use any of these two by enabling the flag.



**Note:** If we are enabling or disabling the Kafka consumer that has been started via Registry class in that case, we need to start the server in order to reflect the changes. Because Kafka consumer started via registry class was started using thread and we can`t get that same thread during registry refresh to kill that thread if flag is disabled.

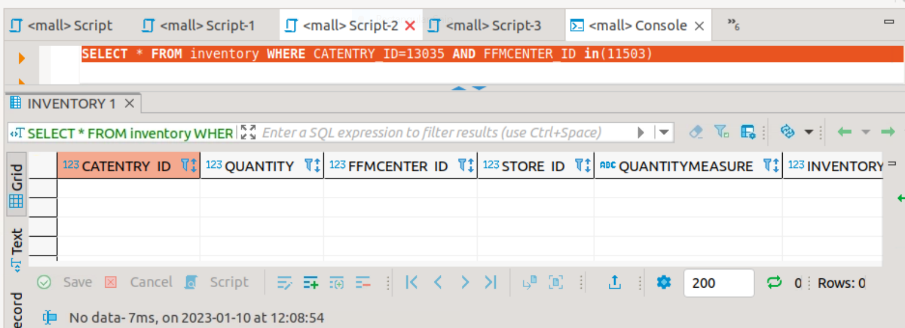
* 1. **Consume message using scheduler approach.**
     1. So, this is another approach using this approach we can also initialize the consumer class to consume the message from Kafka topic.
     2. In this example we have created a scheduler class that will be run on specific interval of time to initialize the Kafka consumer class connection with Kafka server and this scheduler class will consume the message from Kafka topic and uses the Access Bean to save the message record into INVENTORY table.
     3. Now go to CMC and create new scheduler as shown in image.

****

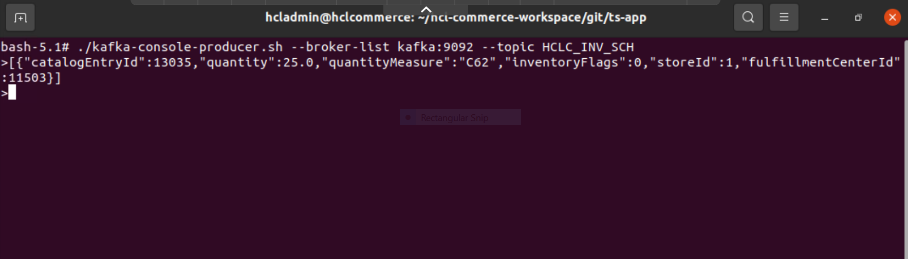
* + 1. Now open Kafka producer terminal that we have opened in step no 11 under Setup Kafka and publish the below message.

|  |
| --- |
| [{"catalogEntryId":13035,"quantity":25.0,"quantityMeasure":"C62","inventoryFlags":0,"storeId":1,"fulfillmentCenterId":11503}] |

As you can see INVENTORY table has no record for above catentryId 13035 and FFMCenterId 11503



As you can see in below Kafka producer terminal, we have published above message on Kafka topic.



* + 1. Once scheduler has run successfully you can verify the INVENTORY has created the new record we have published in previous step.

Graphical user interface, application, website

Description automatically generated

Graphical user interface, text, application

Description automatically generated with medium confidence