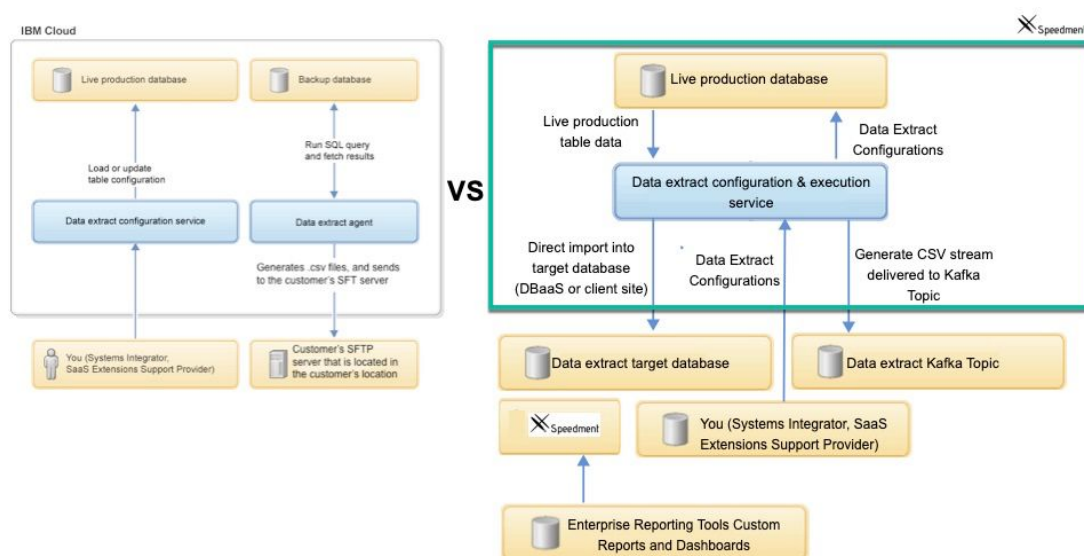


## Speedment Live Data Agent™

*Free your IBM OMoC production data to your Business Users*

Since the release of the IBM Order Management on Cloud (OMoC) customers have been challenged to access their production data in near-real time and have faced the frustration of having to use the only solution made available by IBM that moved table-level entity data over FTP to CSV flat files on an hourly basis at best. Consuming data in this form is not only error-prone, but cumbersome and makes it extremely difficult to expose that data to their business users who need to glean important insights from that data trapped in their IBM OMoC production system.

**Figure 1 - IBM OMoC Data Extract Agent vs. Speedment Live Data Agent**



## 1. The Obstacles

In the world of IBM Order Management on Cloud (OMoC), customers are **not** granted direct access to their production database via JDBC or any other open standard. This creates a major obstacle for customers looking to use their enterprise reporting tools to connect to their order repository to run reports from in near real time. Although IBM provides an agent for customers that will extract data from select tables, the agent does so by creating FTP files and transferring them to the customer site leaving it up to the customer to convert those files into something their reporting tools can generally use.

## 1.1 A Better solution to Access Customer Data

It's apparent that a better solution is needed for IBM OMOC customers and the ideal solution would have the following benefits and capabilities.

1. The solution is a turn-key replacement of the IBM Data Extract Agent that uses cumbersome FTP to transfer table row/column data from the Production IBM OMS system to remote FTP file system with an agent that moves the data either over a *Kafka* Topic Queue or directly into a shadow database. From a Kafka Topic, it can be easily loaded into a database of the customer's choosing (DBaaS or On-Site) that, in-turn will be directly accessible to Speedment via JDBC and *all* it's capabilities.
2. The solution should utilize all the same configuration tables that are already available to set up the IBM Data Extract Agent. This means customers can use existing IBM API's to configure what is fed to the Speedment Live Data Agent. For the most part, the documentation for that IBM Data Extract agent should still apply, although disregarding the FTP-centricity of those docs should be obvious. Those docs are here: [https://www.ibm.com/support/knowledgecenter/en/SSGTJF/com.ibm.help.omcloud.administer.doc/tools/c\\_omc\\_dataextract.html](https://www.ibm.com/support/knowledgecenter/en/SSGTJF/com.ibm.help.omcloud.administer.doc/tools/c_omc_dataextract.html)
3. The solution should utilize Transactional Kafka technology (Kafka is the latest in Scalable Messaging Technology and is used by IBM internally and by many others) to move the data from the IBM OMOC Production DB to a Kafka Topic. The data is delivered to the queue in Comma Delimited Format and the delivery is very fast, very reliable, and very scalable.
4. The solution should allow customers to “**Override**” the part of the ***Speedment Live Data Agent*** that moves the data to ***Kafka*** in CSV format, so the customer can instead, decide to move it over their own ESB technology. The default implementation of this override moves it over Kafka in CSV format.
5. Since the data is typically going to be targeted for a Database vs FTP files, the solution should allow the customer to configure table groups and sequence the extracted tables so they can, for example, send the YFS\_ORDER\_HEADER records before YFS\_ORDER\_LINE records which have a one-to-many relationship and leverage foreign keys to tie the headers to the lines. This grouping should not require any change to the existing YFS\_DATA\_EXTR\_CFG table to accomplish this.
6. The solution should come with a Client Side Server application that can move the data from the Kafka Topic to a **target** Data Base via JDBC. That client application should only read the committed records from the Speedment Live Data Agent topic and should be restartable, and should allow multiple consumers to be active to support one-to-many target db instances. It should also have the ability to create the corresponding table schema on the destination database using the YFS\_DATA\_EXTR\_CFG configurations stored and to re-create any of these tables as new columns are added or others dropped.
7. The solution should extend IBM's DataExtract's “First Run” capabilities that is used to synch back any number of days up to the current date/time. This solution should allow

customers to trigger a Reset to force any Pending or Running tasks refreshed to a given start date dictated by the table's **FirsRunExtractInDays** configuration setting. It should also facilitate a way to tell the downstream client what tables, if any, should be deleted, dropped, created, dropped and created, or left as is in preparation for getting a new full data synch.

## 1.2 Conclusion

The Speedment Live Data Agent lays the groundwork for IBM OMoC customers to realize the full potential of the order management data that is today, **trapped** inside the four walls of the IBM Cloud data centers. Using this revolutionary Data Extract tool that is purpose built for IBM OMoC using all the standard IBM OM Agent frameworks, customers can finally get at the critical transaction data, quickly, efficiently, and with little to no impact on their production system. Once the data is extracted into your own customer-owned databases, you're free to leverage the Speedment tools, or any of your other enterprise reporting frameworks to build real-time reporting capabilities.

## 2 - The Setup of the Speedment Live Data Agent

### 2.1 Step 1 - Create the Sample Data Extract Configuration Records

To create the Sample Data Extract Configurations copy and paste the XML below and paste it into the API Tester set up to call the multiAPI. Don't worry about any existing Configuration Records as these will not be disturbed or used going forward.

```
<?xml version="1.0" encoding="UTF-8"?>
<MultiApi>
  <API FlowName="CocDataExtractConfig">
    <Input>
      <DataExtractConfig Action="MODIFY"
Columns="ORDER_HEADER_KEY,DOCUMENT_TYPE,ENTERPRISE_KEY,BUYER_ORGANIZATION_CODE,SELLER_ORGANIZATION_CODE,BILL_TO_ID,SHIP_T
O_ID,ORDER_DATE,ORDER_NAME,ORDER_NO,ORDER_TYPE,ENTRY_TYPE,AUTHORIZED_CLIENT,REQ_SHIP_DATE,REQ_DELIVERY_DATE,CUSTOMER_FI
RST_NAME,CUSTOMER_LAST_NAME,CUSTOMER_EMAILID,CUSTOMER_PHONE_NO,CUSTOMER_ZIP_CODE,CURRENCY,TAX,PAYMENT_STATUS,TOTAL_A
MOUNT" DataExtractConfigKey="SPEEDMENT-ORDERS-10" FirstRunExtractInDays="30" FrequencyInHours="0" FrequencyInMins="4"
TableName="YFS_ORDER_HEADER" TaskId="SPEEDMENT-ORDERS"/>
    </Input>
  </API>
  <API FlowName="CocDataExtractConfig">
    <Input>
      <DataExtractConfig Action="MODIFY"
Columns="ORDER_LINE_KEY,ORDER_HEADER_KEY,PRIME_LINE_NO,SUB_LINE_NO,ITEM_ID,UOM,PRODUCT_CLASS,DELIVERY_METHOD,ITEM_DESCRIPTI
ON,ITEM_SHORT_DESCRIPTION,SHIPNODE_KEY,SCAC,CARRIER_SERVICE_CODE,RECEIVING_NODE,ORDERED_QTY,OTHER_CHARGES,LINE_TOTAL"
DataExtractConfigKey="SPEEDMENT-ORDERS-20" FirstRunExtractInDays="30" FrequencyInHours="0" FrequencyInMins="4" TableName="YFS_ORDER_LINE"
TaskId="SPEEDMENT-ORDERS"/>
    </Input>
  </API>
  <API FlowName="CocDataExtractConfig">
    <Input>
      <DataExtractConfig Action="MODIFY"
Columns="ITEM_KEY,ORGANIZATION_CODE,ITEM_ID,UOM,DEFAULT_PRODUCT_CLASS,SHORT_DESCRIPTION,DESCRIPTION,EXTENDED_DESCRIPTION,IM
AGE_LOCATION,IMAGE_ID" DataExtractConfigKey="SPEEDMENT-ITEMS-10" FirstRunExtractInDays="3650" FrequencyInHours="0" FrequencyInMins="4"
TableName="YFS_ITEM" TaskId="SPEEDMENT-ITEMS"/>
    </Input>
  </API>
  <API FlowName="CocDataExtractConfig">
    <Input>
      <DataExtractConfig Action="MODIFY" Columns="INVENTORY_ITEM_KEY,ORGANIZATION_CODE,ITEM_ID,UOM,PRODUCT_CLASS"
DataExtractConfigKey="SPEEDMENT-INVENTORY-10" FirstRunExtractInDays="30" FrequencyInHours="0" FrequencyInMins="4"
TableName="YFS_INVENTORY_ITEM" TaskId="SPEEDMENT-INVENTORY"/>
    </Input>
  </API>
  <API FlowName="CocDataExtractConfig">
    <Input>
      <DataExtractConfig Action="MODIFY"
Columns="INVENTORY_SUPPLY_KEY,INVENTORY_ITEM_KEY,SHIPNODE_KEY,SUPPLY_TYPE,QUANTITY,ETA,SEGMENT,SEGMENT_TYPE,AVAILABILITY_TY
PE" DataExtractConfigKey="SPEEDMENT-INVENTORY-20" FirstRunExtractInDays="30" FrequencyInHours="0" FrequencyInMins="4"
TableName="YFS_INVENTORY_SUPPLY" TaskId="SPEEDMENT-INVENTORY"/>
    </Input>
  </API>
  <API FlowName="CocDataExtractConfig">
    <Input>
      <DataExtractConfig Action="MODIFY"
Columns="INVENTORY_DEMAND_KEY,INVENTORY_ITEM_KEY,OWNER_KEY,SHIPNODE_KEY,DEMAND_TYPE,QUANTITY,SEGMENT,SEGMENT_TYPE,DEMAN
D_SHIP_DATE,DEMAND_CANCEL_DATE" DataExtractConfigKey="SPEEDMENT-INVENTORY-30" FirstRunExtractInDays="30" FrequencyInHours="0"
FrequencyInMins="4" TableName="YFS_INVENTORY_DEMAND" TaskId="SPEEDMENT-INVENTORY"/>
    </Input>
  </API>
</MultiApi>
```

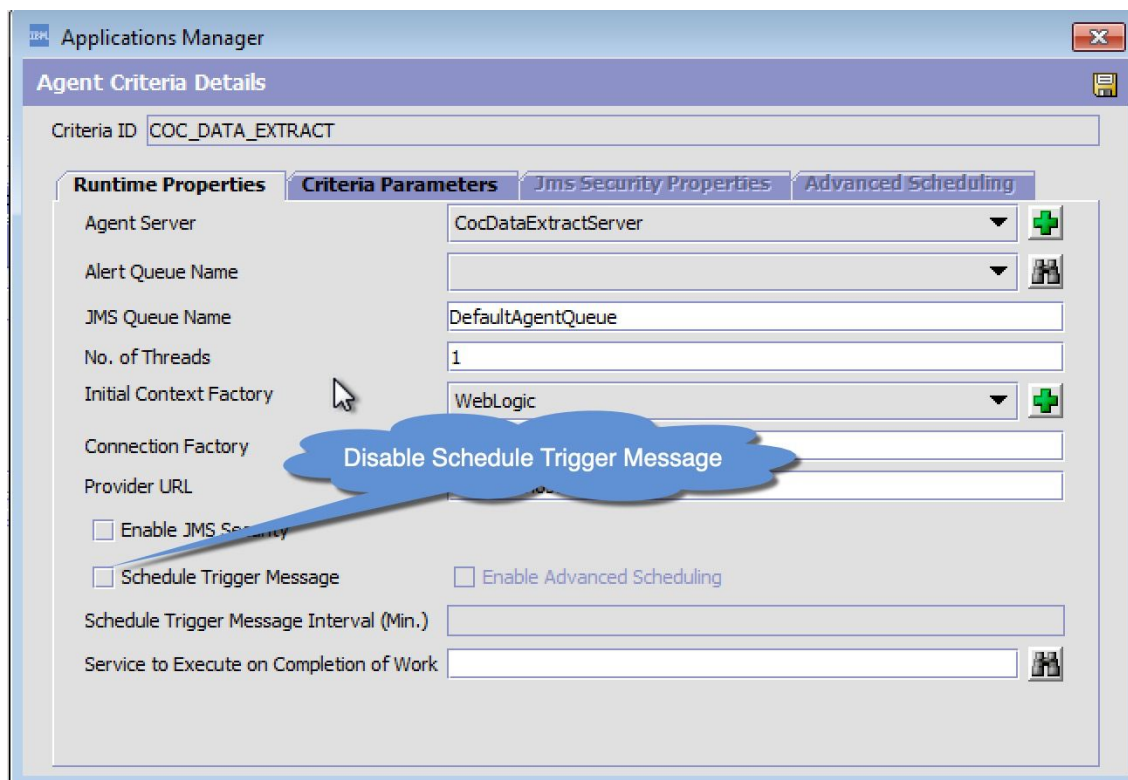
This XML can also be found in the file manageDataExtractConfigInput.xml inside the SpeedmentLiveDataAgent.jar file under install.

Name
com
docs
install
META-INF

## 2.2 Step 2 - Disable Triggering the Original COC\_DATA\_EXTRACT Agent provided by IBM

**IMPORTANT NOTE:** You ***MUST*** disable the Schedule Trigger Message option for the original **Data Extract Agent** provided by IBM as you don't want to run this agent any longer.

Figure 2 - COC\_DATA\_EXTRACT Transaction



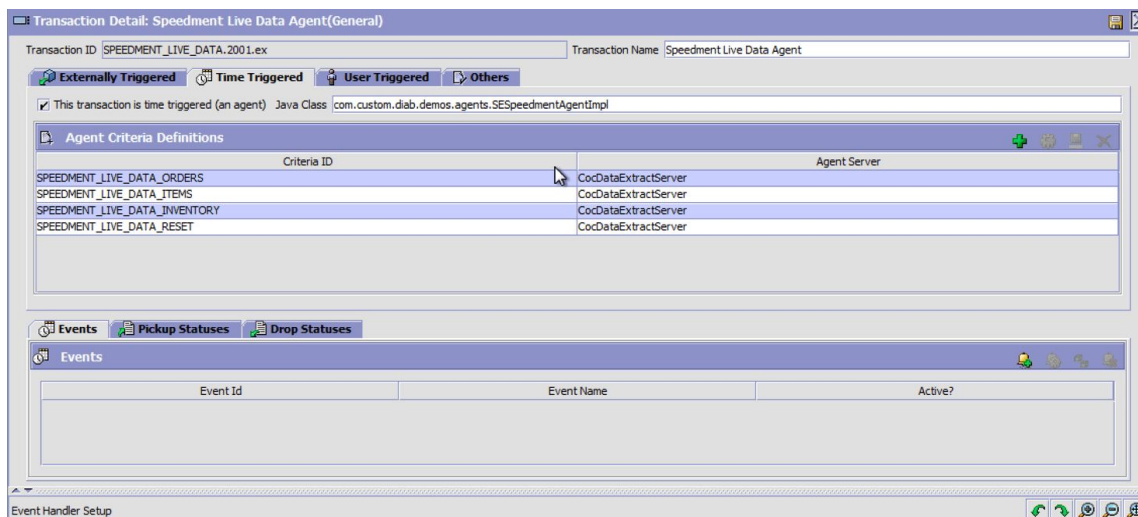
## 2.3 Step 3 - Create a Custom Agent in the General Process Type

Below is an image of what the agent setup looks like in the IBM OMS Applications Manager UI. Effectively we're setting up one Triggerable Task for each table or table group we want to extract. Doing this we can ensure master data is synched separate from transaction data and on schedules that make sense for each entity.

You will start by creating a new transaction in the **General** Repository with **SPEEDMENT\_LIVE\_DATA.0002.ex** as the **Transaction ID**. Note that you don't need to enter the 'ex' portion as that's done for you. Make sure to set it up as a new transaction vs. an extended transaction in the **General Process Type** Repository of **Transactions**. In the **Time Triggered** tab you'll enter the class name for the Speedment Live Data Agent which is:

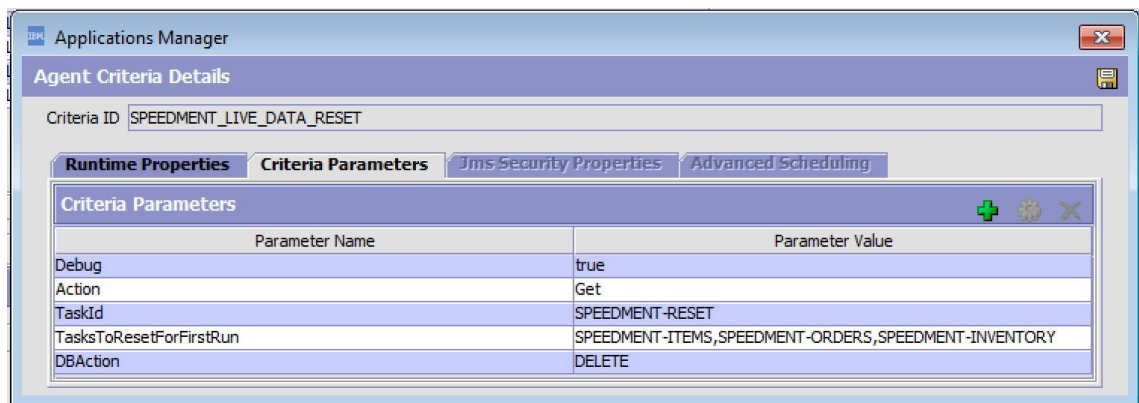
Java Class: **com.speedment.livedata.agent.SpeedmentLiveDataAgentImpl**.

**Figure - 3 - Agent Configuration -1**



**IMPORTANT NOTE:** that the SPEEDMENT-RESET Criteria shown above is not configured as an extract task but instead is an Internal task that the Speedment Live Data Agent uses to facilitate resetting jobs that are not in a valid terminal status, or to force the “First Run” of a job (i.e. extracts **FirstRunExtractDays** worth of records). The Client-side Consumer application will process these RESET requests and performs a specific DB Action prior to executing the “First Run RESET”

**Figure 4 - SPEEDMENT\_LIVE\_DATA\_RESET Transaction Setup**



Initially you would want to setup the SPEEDMENT-RESET agent to trigger manually and you should always check to make sure the **TasksToResetForFirstRun** are correct before manually or scheduling trigger messages.

### Runtime Properties Setup

The Runtime configuration should resemble all your other agents and should use the already provided **CocDataExtratServer**. The Schedule Trigger Message Interval should be set to reflect the minimum frequency of the all the data extract config records who share the same **TaskId**. So, for example, both the YFS\_ORDER\_HEADER and YFS\_ORDER\_LINE tables have a default **FrequencyInMins=4**, so it would make sense to trigger this agent every 4 minutes. It won't hurt to trigger it more or less frequently as the agent will ignore triggers that occur inside the 4 minute interval, and will roll-up all the changes in 4 minute increments (never to exceed the current time) if triggers occur outside the 4 minute interval.

## Criteria Parameters Setup

The **Criteria Parameter** configuration should consist of one agent criteria record for each **TaskId** configured and created in **Step 1** using the **CocDataExtractConfig** service as defined in the Service Definition Framework (SDF).

In the example provided here we have three **TaskId**'s configured. Assuming you've created the Data Extract Configuration Records in **Step 1** to capture the YFS\_ORDER\_HEADER and YFS\_ORDER\_LINE tables, the data extract config records would look as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<DataExtractConfigList TotalNumberOfRecords="2">
  <DataExtractConfig
    Columns="ORDER_HEADER_KEY,DOCUMENT_TYPE,ENTERPRISE_KEY,BUYER_ORGANIZATION_CODE,SELLER_ORGANIZATION_CODE,BILL_TO_ID,SHIP_TO_ID,ORDER_DATE,ORDER_NAME,ORDER_NO,ORDER_TYPE,ENTRY_TYPE,AUTHORIZED_CLIENT,REQ_SHIP_DATE,REQ_DELIVERY_DATE,CUSTOMER_FIRST_NAME,CUSTOMER_LAST_NAME,CUSTOMER_EMAILID,CUSTOMER_PHONE_NO,CUSTOMER_ZIP_CODE,CURRENCY,TAX,PAYMENT_STATUS,TOTAL_AMOUNT"
    DataExtractConfigKey="SPEEDMENT-ORDERS-10" FirstRunExtractInDays="30" FrequencyInHours="0" TableName="YFS_ORDER_HEADER"
    TaskId="SPEEDMENT-ORDERS"/>
  <DataExtractConfig
    Columns="ORDER_HEADER_KEY,ORDER_LINE_KEY,PRIME_LINE_NO,SUB_LINE_NO,ITEM_ID,UOM,PRODUCT_CLASS,DELIVERY_METHOD,ITEM_DESCRIPTION,ITEM_SHORT_DESCRIPTION,SHIPNODE_KEY,SCAC,CARRIER_SERVICE_CODE,RECEIVING_NODE,ORDERED_QTY,OTHER_CHARGES,LINE_TOTAL"
    DataExtractConfigKey="SPEEDMENT-ORDERS-20" FirstRunExtractInDays="30" FrequencyInHours="0" TableName="YFS_ORDER_LINE"
    TaskId="SPEEDMENT-ORDERS"/>
</DataExtractConfigList>
```

Or viewed another way:

Node	Content
?? xml	version="1.0" encoding="UTF-8"
DataExtractConfigList	
TotalNumberOfRecords	2
DataExtractConfig	
Columns	ORDER_HEADER_KEY,DOCUMENT_TYPE,ENTERPRISE_KEY,BUYER_ORGANIZATION_CODE,SELLER_ORGANIZAT...
DataExtractConfigKey	SPEEDMENT-ORDERS-10
FirstRunExtractInDays	30
FrequencyInHours	0
TableName	YFS_ORDER_HEADER
TaskId	SPEEDMENT-ORDERS
DataExtractConfig	
Columns	ORDER_HEADER_KEY,ORDER_LINE_KEY,PRIME_LINE_NO,SUB_LINE_NO,ITEM_ID,UOM,PRODUCT_CLASS,DELIV...
DataExtractConfigKey	SPEEDMENT-ORDERS-20
FirstRunExtractInDays	30
FrequencyInHours	0
TableName	YFS_ORDER_LINE
TaskId	SPEEDMENT-ORDERS

Note that the **TaskId** for each of the Data Extract Configuration records is identical (SPEEDMENT-ORDERS) which makes them part of a table group. The **DataExtractConfigKey** attribute is explicitly set to include the TASKID-SEQNO, where TASKID=the TaskId and SEQNO=Sequence of the table in the Extract Job. So for example, in this case the SPEEDMENT-ORDERS-10 (YFS\_ORDER\_HEADER) job is always extracted before SPEEDMENT-ORDERS-20 (YFS\_ORDER\_LINE) job to ensure the header records will exist on the target database before the order lines are provided.



## SPEEDMENT Criteria Parameter Data Used by Live Data Agent Tasks

<b>EXTRACT TASK CRITERIA</b>	<b>All tasks configured in YFS_DATA_EXTR_CFG</b>
<b>Action</b>	<b>Get (Always use this value)</b>
<b>Debug</b>	<b>false (Will output debugging messages if true)</b>
<b>TaskId</b>	<b>Task Id from Data Extract Config Record</b>
<b>SimulatedRunTime</b>	<b>0 (Always use this value - speedment use only)</b>
<b>SPEEDMENT-RESET ONLY</b>	<b>Additional Criteria is Optional</b>
<b>DBAction</b>	<b>DELETE   DROP   CREATE   DROPANDCREATE   NONE</b>
<b>TasksToResetForFirstRun</b>	<b>TaskId-1,TaskId-2,TaskId-3</b>

**Note the DBAction is only used by the SPEEDMENT-RESET transaction.** This action will be taken on the client side database for the tasks listed in the **TasksToResetForFirstRun**.

**Note the TasksToResetForFirstRun is only used by the SPEEDMENT-RESET transaction.** Include any task id that you want to take the corresponding **DBAction** on. All tables associated to that task id will be included in the DBAction.

**Figure 4 - SPEEDMENT-RESET Criteria**

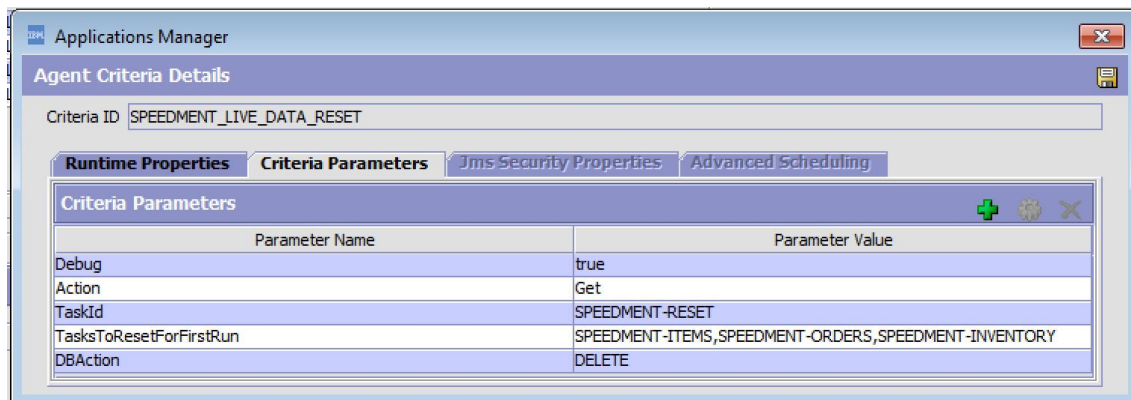


Figure 5 - SPEEDMENT-ORDERS Criteria

Applications Manager

Agent Criteria Details

Criteria ID: SPEEDMENT\_LIVE\_DATA\_ORDERS

Runtime Properties Criteria Parameters Jms Security Properties Advanced Scheduling

Criteria Parameters

Parameter Name	Parameter Value
Debug	false
Action	Get
TaskId	SPEEDMENT-ORDERS
SimulatedRunTime	0

Figure - 6 - SPEEDMENT-ITEMS Criteria

Applications Manager

Agent Criteria Details

Criteria ID: SPEEDMENT\_LIVE\_DATA\_ITEMS

Runtime Properties Criteria Parameters Jms Security Properties Advanced Scheduling

Criteria Parameters

Parameter Name	Parameter Value
Debug	false
Action	Get
TaskId	SPEEDMENT-ITEMS
SimulatedRunTime	0

Figure 7 - SPEEDMENT-INVENTORY Criteria

Applications Manager

Agent Criteria Details

Criteria ID: SPEEDMENT\_LIVE\_DATA\_INVENTORY

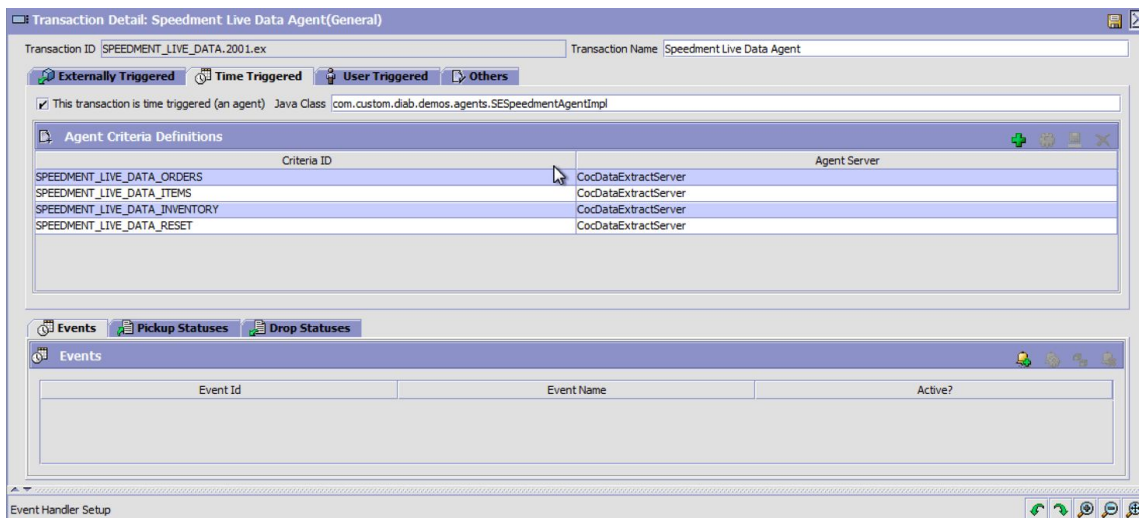
Runtime Properties Criteria Parameters Jms Security Properties Advanced Scheduling

Criteria Parameters

Parameter Name	Parameter Value
Debug	false
Action	Get
TaskId	SPEEDMENT-INVENTORY
SimulatedRunTime	0

When completed your agent configuration should look as follows:

**Figure - 8 - Finalized Agent Configuration**



## 2.4 Step 4 - Install Properties for the Live Data Agent

A set of **customer\_overrides.properties** is available for legacy customers that still use the customer\_overrides method to extend the System Properties. Note these properties are not manageable if added to the **customer\_overrides.properties**.

To make these properties manageable via the Systems Management Console, there is a **managePropertiesInput.xml** file that you will find in the install folder that can be used to add these properties to the database as manageable properties. If you do this you don't have to add them to the **customer\_overrides.properties**.

```
#
#SPEEDMENT Live Data Agent Properties
#This should be copied into customer_overrides.properties when ever it changes.
#

#####
# IMPORTANT: PRODUCER PROPERTIES MUST BE COPIED TO customer_overrides.properties
#####
yfs.speedment.producer.kafka.enabled=true
yfs.speedment.producer.kafka.topic=speedment-topic
yfs.speedment.producer.kafka.bootstrap.servers=<KafkaServer DNS>:<Port>
yfs.speedment.producer.kafka.transactional.id=SPEEDMENT_LIVE_DATA.2001.ex

# SAMPLE DB connection for a DB2 database
yfs.speedment.producer.database.enabled=false
yfs.speedment.producer.database.DstDBType=DB2
yfs.speedment.producer.database.DstDBServer=<DBServer DNS>
yfs.speedment.producer.database.DstDBPort=50000
yfs.speedment.producer.database.DSTDatabase=OMDB
yfs.speedment.producer.database.DstDBUsername=<DB UserName>
#our database password has been encrypted by the live data agent's encrypter class
yfs.speedment.producer.database.DstDBPassword=<DB Password>
yfs.speedment.producer.database.DstDBSchema=OMDB

# columns that need to be url decoded - typically description columns
yfs.speedment.urlencoded.columns=ITEM_DESCRIPTION,ITEM_SHORT_DESCRIPTION,SHORT_DESCRIPTION,DESCRIPTION,EXTENDED_DESCRIPTION,IMAGE_ID

# encryption interfaces
#yfs.security.encrypter.class=<value>
#yfs.security.propertyencrypter.class=com.speedment.livedata.encrypter

#####
# IMPORTANT: PRODUCER PROPERTIES MUST BE COPIED TO customer_overrides.properties
#####
```

Note the **yfs.speedment.producer.database.DstDBPassword** property can be encrypted by the Live Data Client application. Run the live data client with the **-help** option to see how this is done.

**IMPORTANT: These two properties determine what mode(s) the live data agent runs in**

**yfs.speedment.producer.kafka.enabled=true|false**

**yfs.speedment.producer.database.enabled=true|false**

If Kafka option is Enabled then all the data records will be written to a Kafka Topic determined by the properties. If the Database option is Enabled, then in all records will be directly written from the Live Data Agent directly to a database, however there you should keep in mind that there is additional overhead to writing to a Database vs. writing to Kafka.

**IMPORTANT: Use of Kafka vs. Database Mode is Strongly Encouraged**

Using the **Database mode**, writes data directly to the target database from the Live Data Agent and eliminates the need for using the client-side server. While this mode is allowed, this mode can be prone to more extract failures if the client database is not available, and it can degrade the performance of the Live Data Agent if there is significant latency in accessing the target database. Using the **Kafka mode** instead eliminates any issues that may result from the Client Side database not being available or accessible and any latency issues that might result depending on where the destination database is located relative to your OMS instance.

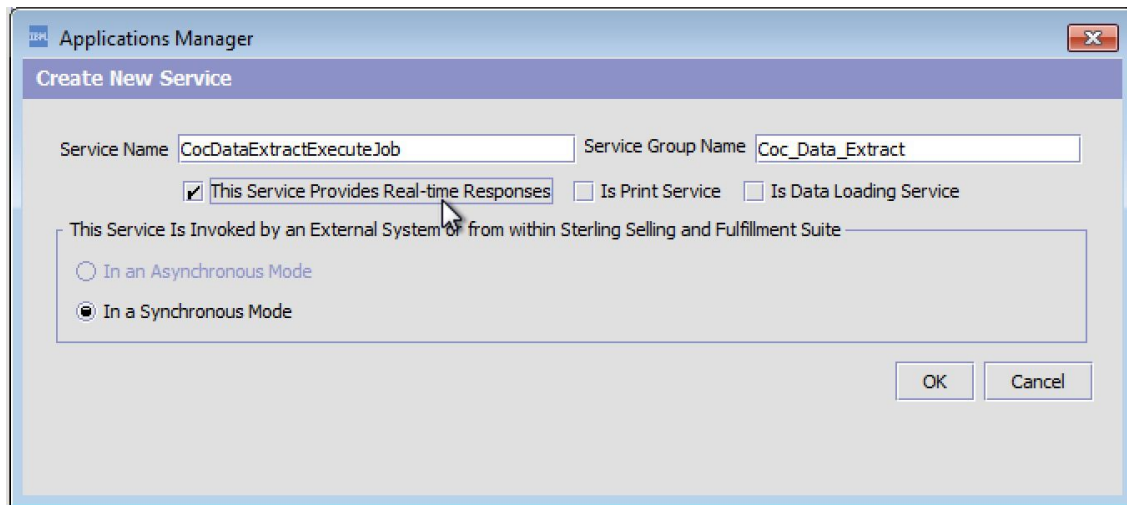
## 2.5 - Step 5 - Create a new Custom API to Call the Extract Worker API

The Live Data Agent allows you to configure your own “**executeJob Worker**” API which will get called every time the Live Data Agent has a record to write to the target. The worker provided for you will write all the data records over a **Kafka** topic named “**speedment-topic**”, or write directly to an alternative database based on the server’s properties. These live data properties should be added to your customer\_overrides.properties.

Open up the SDF services in the General Repository and look for the CocDataExtract Group of services.



Create a new Synchronous Service in the Coc\_Data\_Extract Group of Services as shown here:



**Applications Manager**

**Create New Service**

Service Name:  Service Group Name:

☒ This Service Provides Real-time Responses ☐ Is Print Service ☐ Is Data Loading Service

This Service Is Invoked by an External System or from within Sterling Selling and Fulfillment Suite

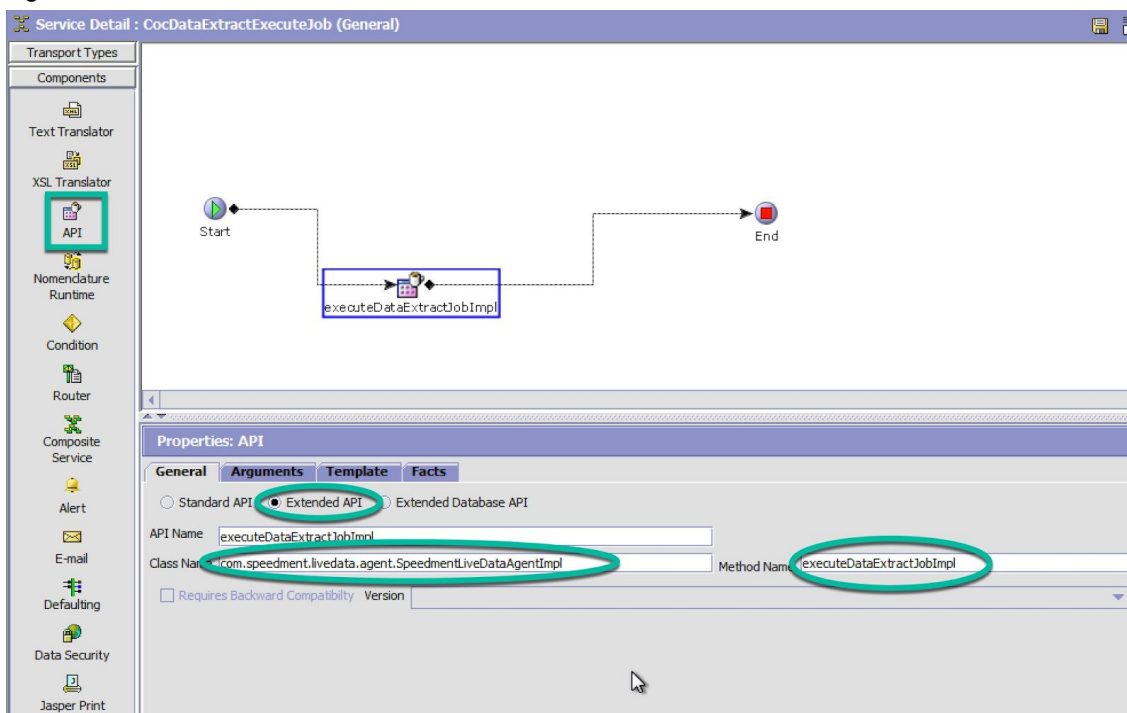
☐ In an Asynchronous Mode

☒ In a Synchronous Mode

OK Cancel

This service will, by default, be setup to call the built-in Custom API provided as part of the Speedment Live Data Extract Agent and so you'll configure the service to call an API as shown here:

Figure 7 - CocDataExtractExecuteJob Custom API



Drag API component into SDF, and select the Extended API and then enter in:

**Api Name:** *executeDataExtractJobImpl*

**Class Name:** *com.speedment.livedata.agent.SpeedmentLiveDataAgentImpl*

**Method Name:** *executeDataExtractJobImpl*

## 3.0 - Running the Live Data Agent Server

TBD

## 4.0 - Running the Live Data Client Server

TBD