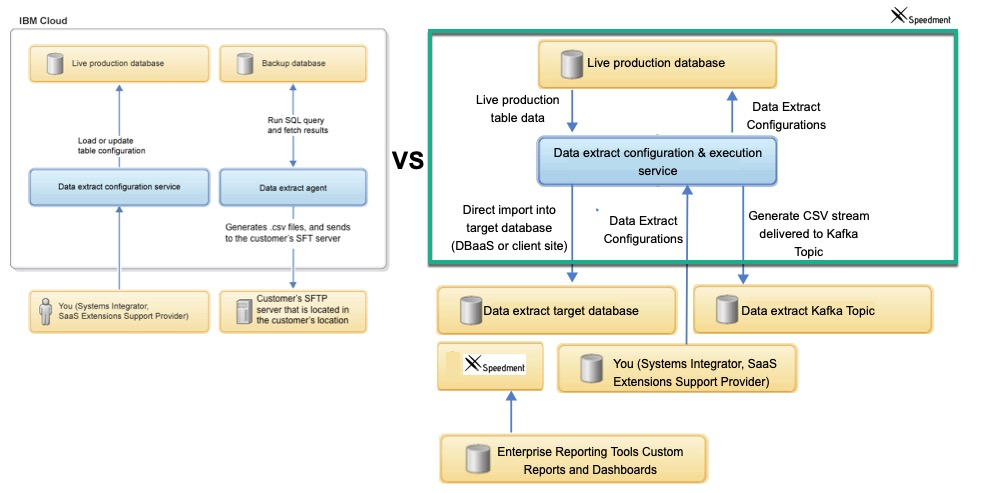
**Speedment Live Data Agent™**

*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. Furthermore, there is no way to schedule these FTP extracts any more frequently than one hour at a time, which means, at best, the data is more than an hour old, while further, significant efforts are required to to adjust the routines processing the FTP files to adapt to any changes to the data extract configurations. This is just plain difficult and customers often wonder why they can’t have their own data extracted directly into their own shadow copy for use in analytics and reporting and on a near-real time basis. **Well now that’s possible with Speedment Live Data Agent for IBM OMoC!**

# 1.1 A Better solution to Access Customer Data

It’s apparent that a better solution is needed for IBM OMoC customers to access their production data in near-real time and the ideal solution would have the following benefits and capabilities.

1. The solution would be a turn-key replacement of the IBM Data Extract Agent that uses cumbersome FTP to transfer table row/column data from the **Backup** IBM OMS system to a remote FTP file system with an agent that moves the data from the **Production** instance over a *Kafka* Topic Queue, directly into a shadow database or both. From a Kafka Topic, it can be easily loaded into a database of the customer’s choosing (DBaaS or On-Site) using a Client-Side Kafka Consumer application (provided with the solution) that can automatically move the data on the Kafka Topic directly into a client-side shadow database. That database, in-turn will be directly accessible to your enterprise reporting tools and by Speedment’s enhanced streaming tools via JDBC and ***all*** it’s capabilities.

1. 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, disregarding the FTP-centricity aspects since FTP is not used.. Those docs are here:<https://www.ibm.com/support/knowledgecenter/en/SSGTJF/com.ibm.help.omcloud.administer.doc/tools/c_omc_dataextract.html>

1. 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. Using Kafka Transactions, the clients can request only the committed records from the topic and can communicate to the Kafka Partition which records have been consumed. This virtually eliminates any issues with data loss and ensures only what was intended to be consumed to be consumed.
2. 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 and/or directly into a target database you can configure via **customer\_override.properties**.
3. 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.
4. The solution should come with a Client Side Server application that can move the data from the Kafka Topic to a **target** to a database (DBaaS or Local) via JDBC. That client application should only read the committed records from the Speedment Live Data Agent topic and should be stoppable, and 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.
5. 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, or 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 their 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 framework, customers can finally get at their critical transaction data, in near real-time, quickly, efficiently, and with little to no impact on their production system. Once the data is extracted into the customer-owned databases, they’re free to leverage the Speedment tools, or any of their other enterprise reporting frameworks to build the real-time reports..

Finally, if you’re an IBM OMoC customer leveraging the IBM Cognos on Cloud solution, you now have a much more flexible alternative that will allow you to use your own internal enterprise reporting tools. Your reports will be more real-time and you can reduce the resources needed to care and feed this less than optimal reporting solution.

# 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\_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" 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\_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" 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,IMAGE\_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\_TYPE" DataExtractConfigKey="SPEEDMENT-INVENTORY-20" FirstRunExtractInDays="30" FrequencyInHours="0" FrequencyInMins="4" TableName="YFS\_INVENTORY\_SUPPLY" TaskId="SPEEDMENT-INVENTORY"/>

</Input>

</API>

<API FlowName="CocDataExtractConfig">

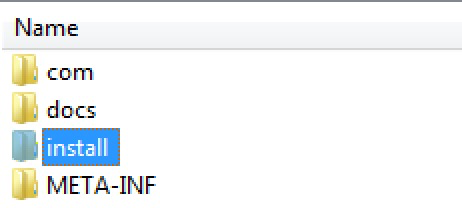
<Input>

<DataExtractConfig Action="MODIFY" Columns="INVENTORY\_DEMAND\_KEY,INVENTORY\_ITEM\_KEY,OWNER\_KEY,SHIPNODE\_KEY,DEMAND\_TYPE,QUANTITY,SEGMENT,SEGMENT\_TYPE,DEMAND\_SHIP\_DATE,DEMAND\_CANCEL\_DATE" DataExtractConfigKey="SPEEDMENT-INVENTORY-30" FirstRunExtractInDays="30" FrequencyInHours="0" FrequencyInMins="4" TableName="YFS\_INVENTORY\_DEMAND" TaskId="SPEEDMENT-INVENTORY"/>

</Input>

</API>

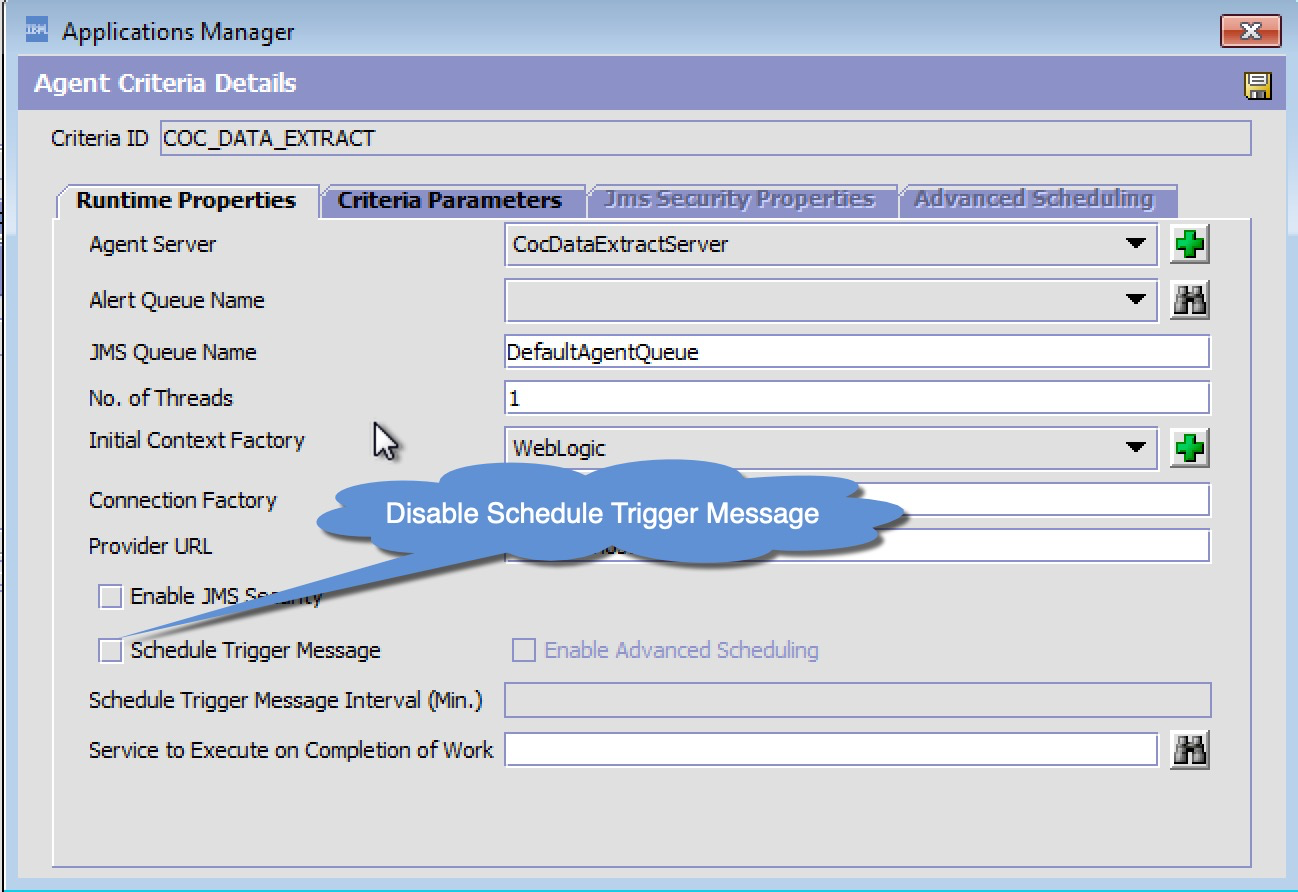
</MultiApi

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


## 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



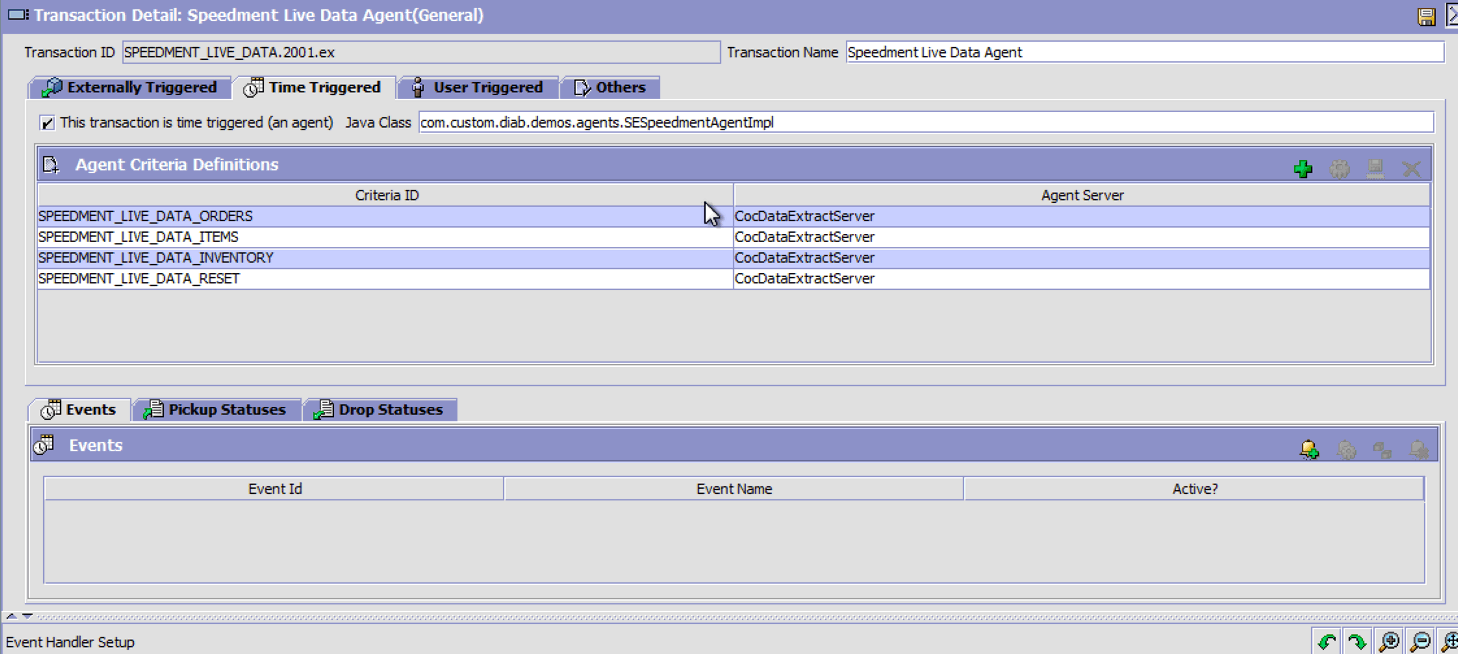
## 2.3 Step 3 - Create a Custom Agent in the General Process Type

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:

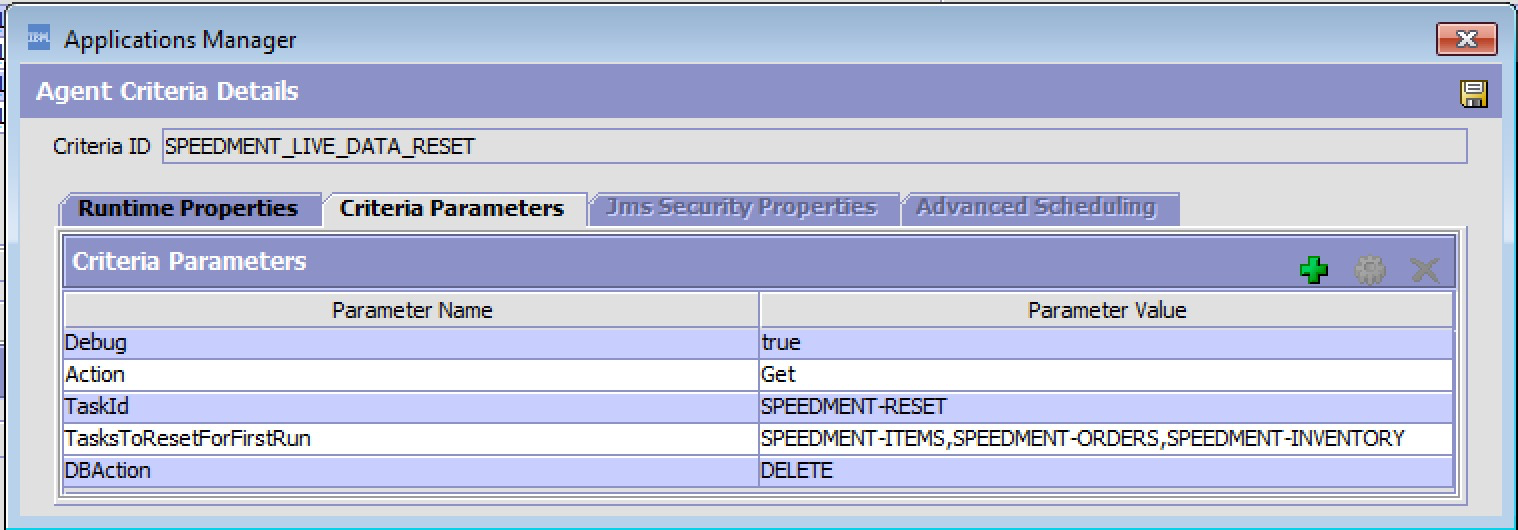
Java Class: **com.speedment.livedata.agent.SpeedmentLiveDataAgentImpl.**

**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 extract jobs that are not in a valid terminal status (SUCCESS/FAILED), 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. It’s worth pointing out that each client consumer can override the **DBAction** using -DDABction=<DBAction> on the JVM arguments when starting the client-side consumer.

#### **Runtime Properties Setup**

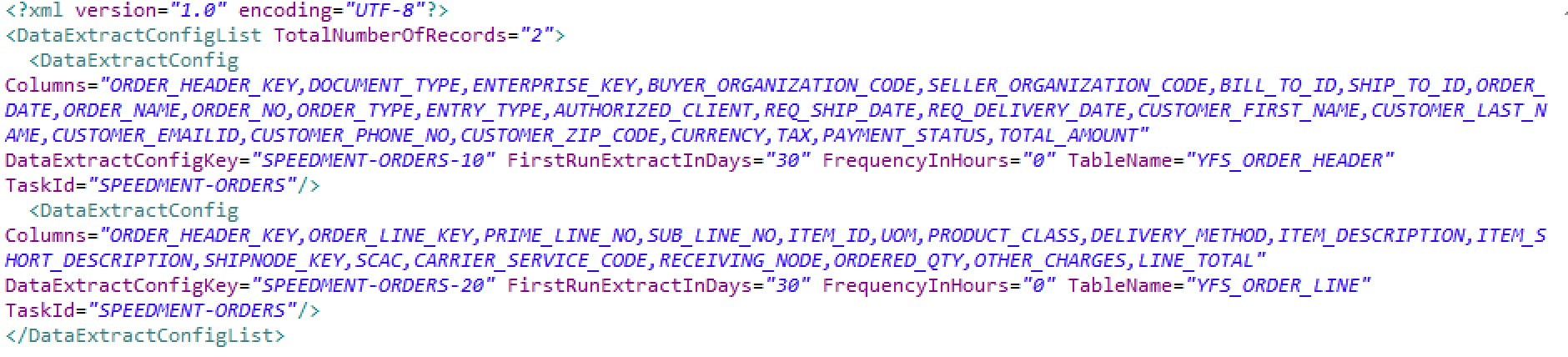
The Runtime configuration should resemble all your other agents and should use the already provided **CocDataExtratServer**. The Schedule Trigger Message Interval should be set to reflect the minimum frequency of the all the data extract config records who share the same **TaskId**. So, for example, 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**

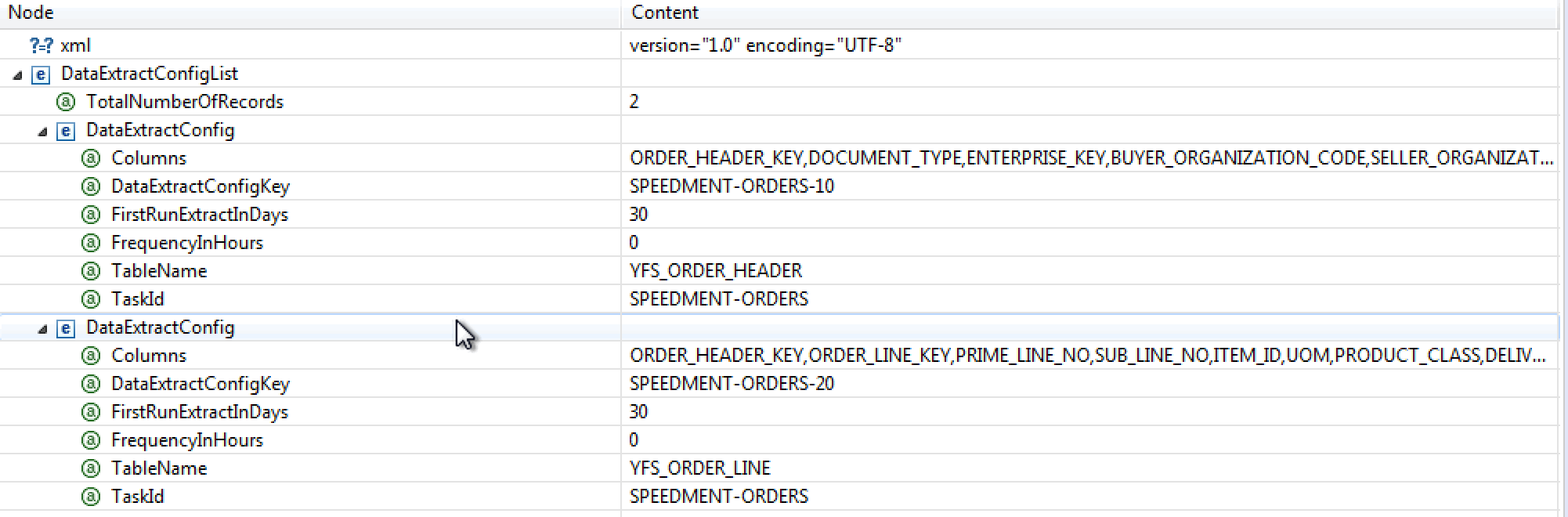
The **Criteria Parameter** configurations should consist of one agent criteria record for each **TaskId** configured and created in **Step 1** using the **CocDataExtractConfig** service as defined in the Service Definition Framework (SDF). The Initial sample extract records includes 3 Tasks Id’s (excluding the SPEEDMENT-RESET task). They include the SPEEDMENT-ITEMS, SPEEDMENT-INVENTORY and SPEEDMENT-ORDERS Tasks. A sample of the SPEEDMENT-ORDERS task is depicted below

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:

### Figure 5 - Sample SPEEDMENT-ORDERS Task Configuration



Or viewed another way:



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.

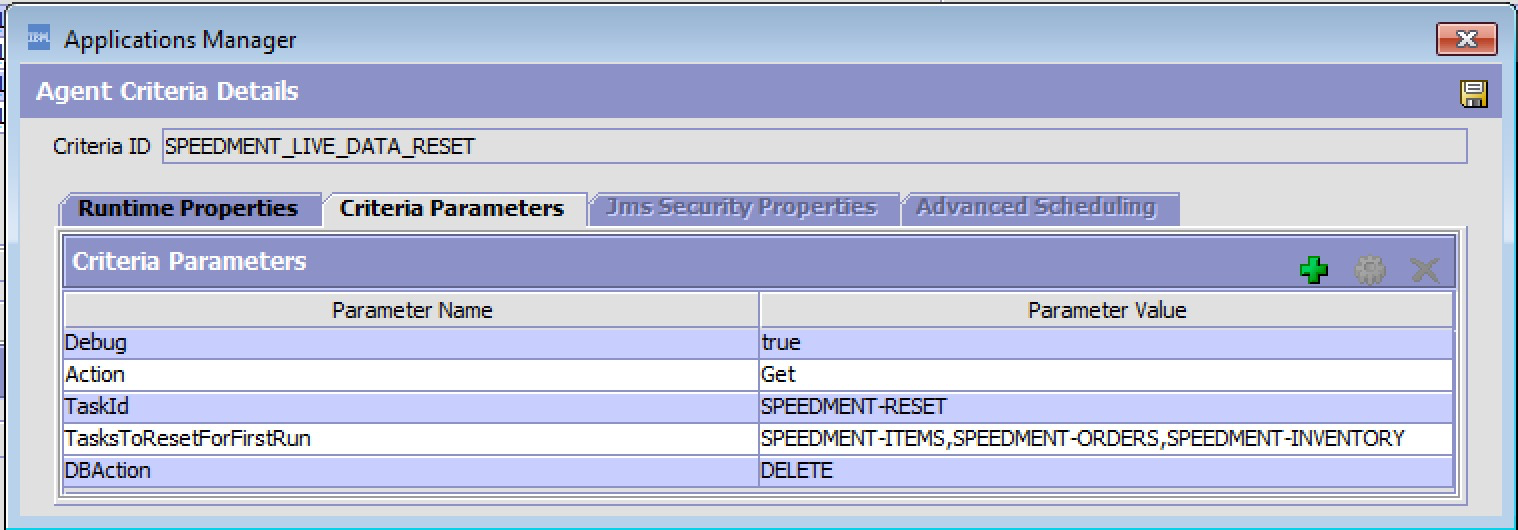
**SPEEDMENT Criteria Parameter Data Used by Live Data Agent Tasks**

|  |  |
| --- | --- |
| **EXTRACT TASK CRITERIA** | **All tasks configured in YFS\_DATA\_EXTR\_CFG** |
| **Action** | **Get (Always use this value)** |
| **Debug** | **false (WIll output helpful INFO Log messages if true)** |
| **TaskId** | **Task Id from Data Extract Config Record** |
| **SimulatedRunTime** | **0 (Always use this value - speedment use only)** |
| **SPEEDMENT-RESET ONLY** | **Additional Criteria is Optional** |
| **DBAction** | **DELETE | DROP | CREATE | DROPANDCREATE | NONE** |
| **TasksToResetForFirstRun** | **TaskId-1,TaskId-2,TaskId-3** |

**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** unless that action is overridden by the Client-Side consumer.

**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’s listed will be included in the requested **DBAction**.

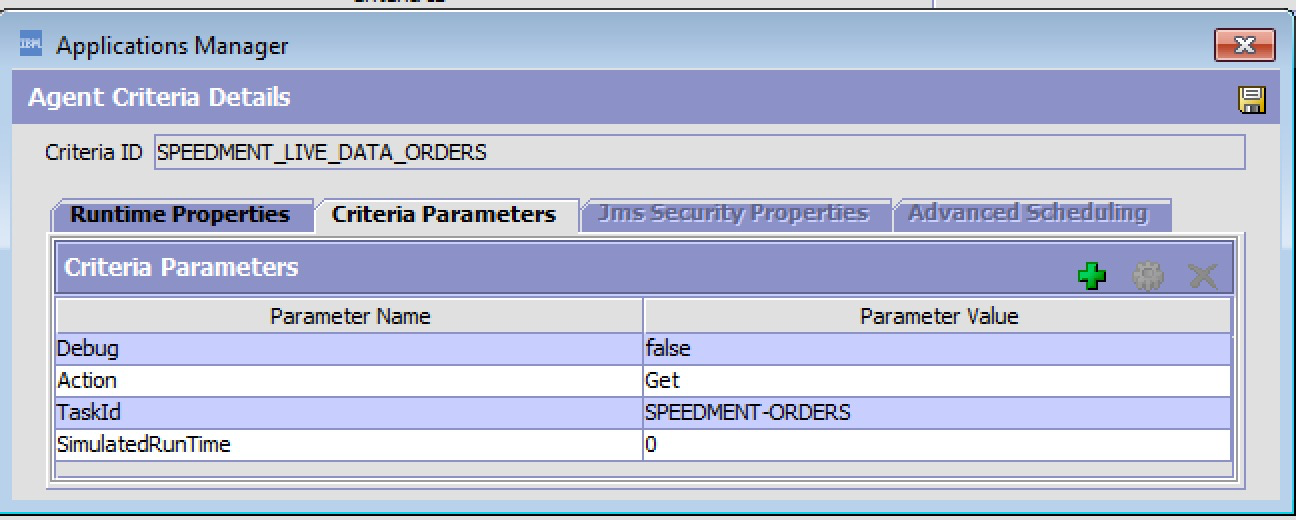
**Figure 4 - SPEEDMENT-RESET Criteria**

****

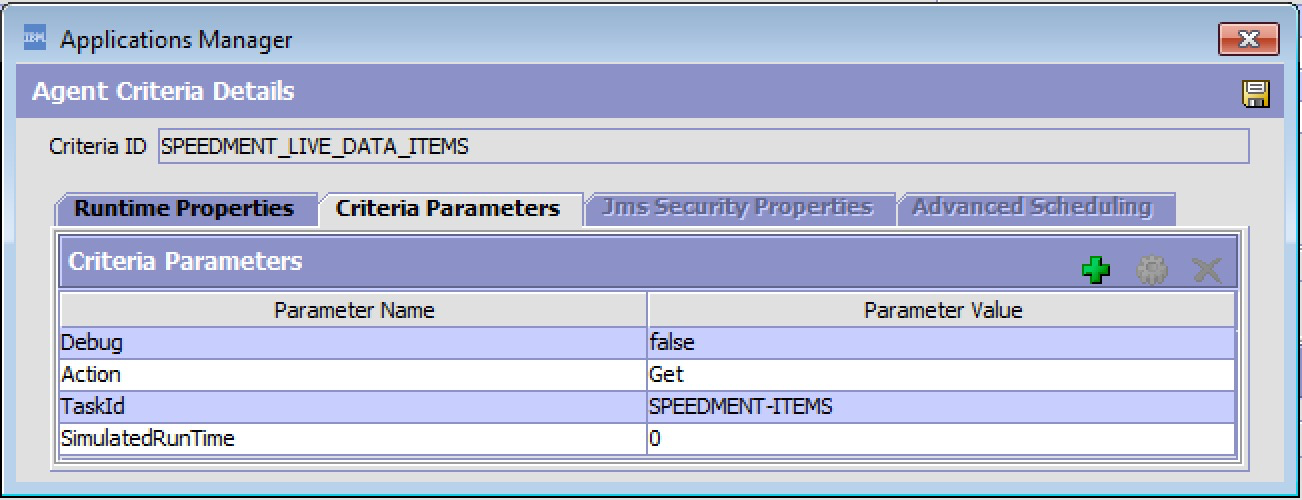
The **CREATE** only operates if the tables associated with the provided Task Id’s don’t exist and the **DROP** only operates on tables that do exist. The **DROPANDCREATE** combines bothe a **DROP** and a **CREATE** action in one **RESET** trigger.

**IMPORTANT NOTE** - The CREATE task on the client side application automatically generates the SQL CREATE TABLE schema based on the **Columns** chosen for the given Data Extract configuration so if you change the set of columns (add or remove) or add new tables to a given Task Id, you should always trigger the **SPEEDMENT-RESET** for those tasks that are affected using the **DROPANDCREATE** **DBAction**.

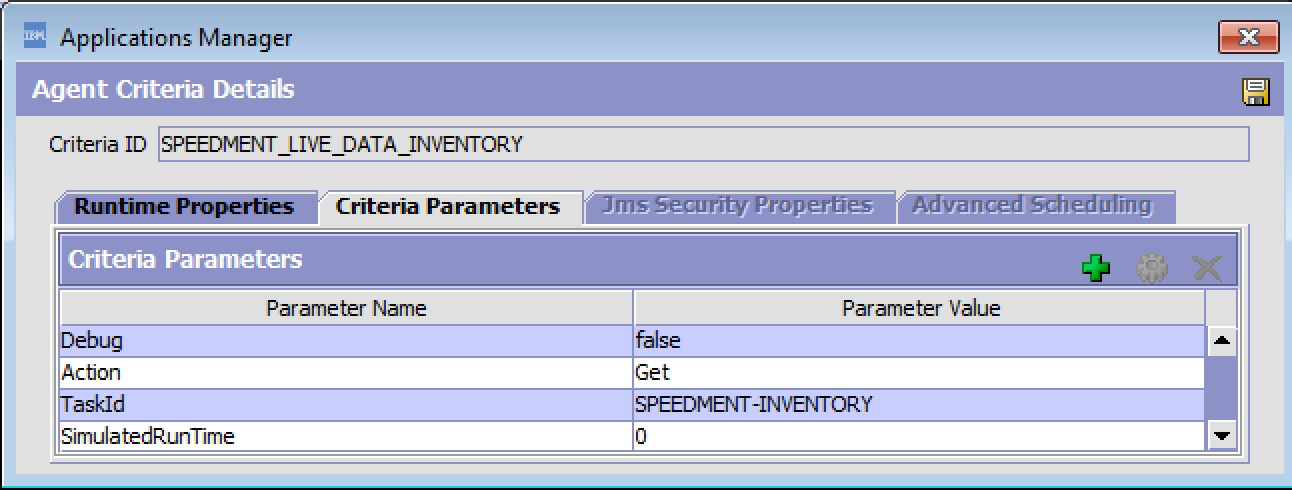
**Figure 5 - SPEEDMENT-ORDERS Criteria**



### Figure - 6 - SPEEDMENT-ITEMS Criteria

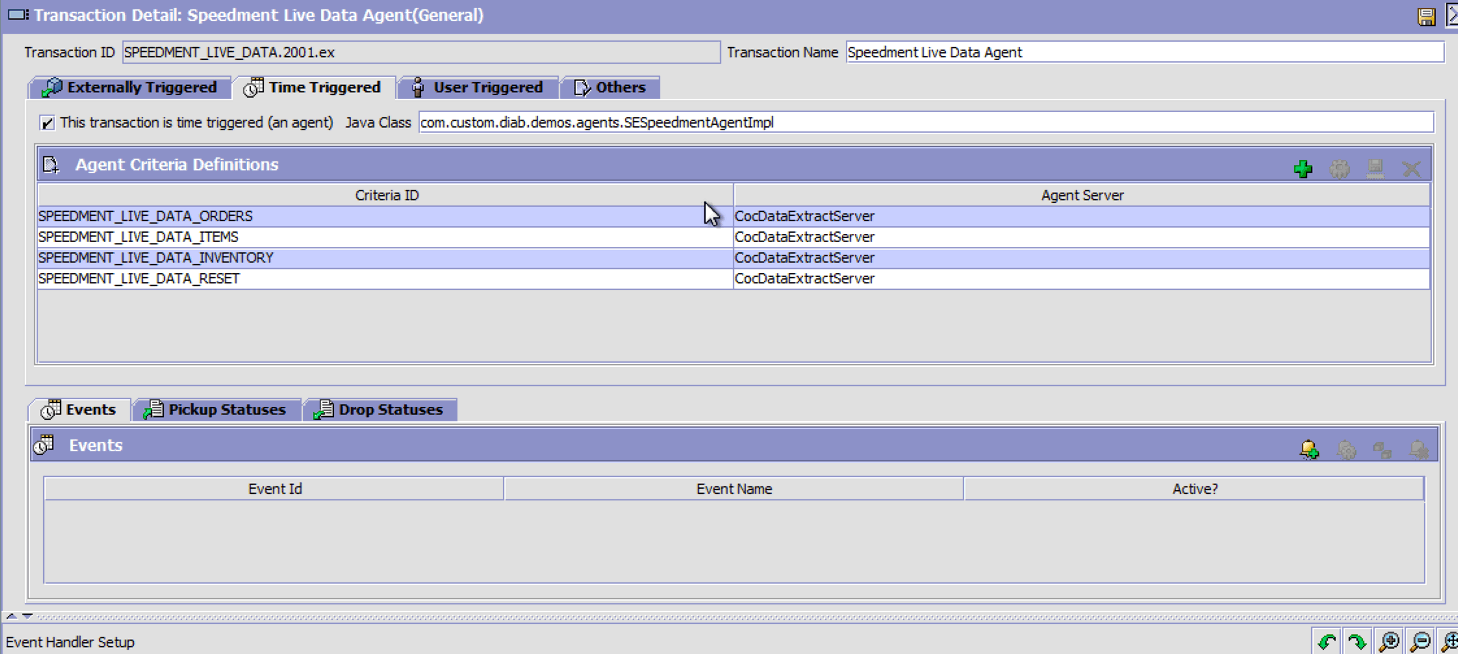


### Figure 7 - SPEEDMENT-INVENTORY Criteria



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 **cutomer\_overrides.properties** is available for legacy customers that still use the customer\_overrrides 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.

**MPORTANT: 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**

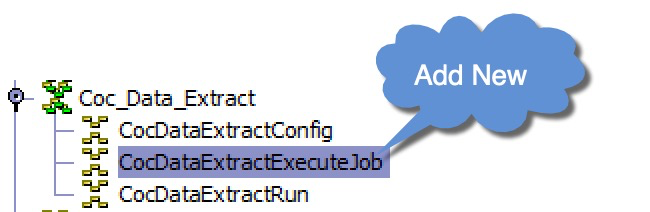
Using the **Database mode,** writes data directly to the target database from the Live Data Agent and eliminates the need for using the client-side server. While this mode is allowed, this mode can be prone to more extract failures if the client database is not available, and it can degrade the performance 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.

## 

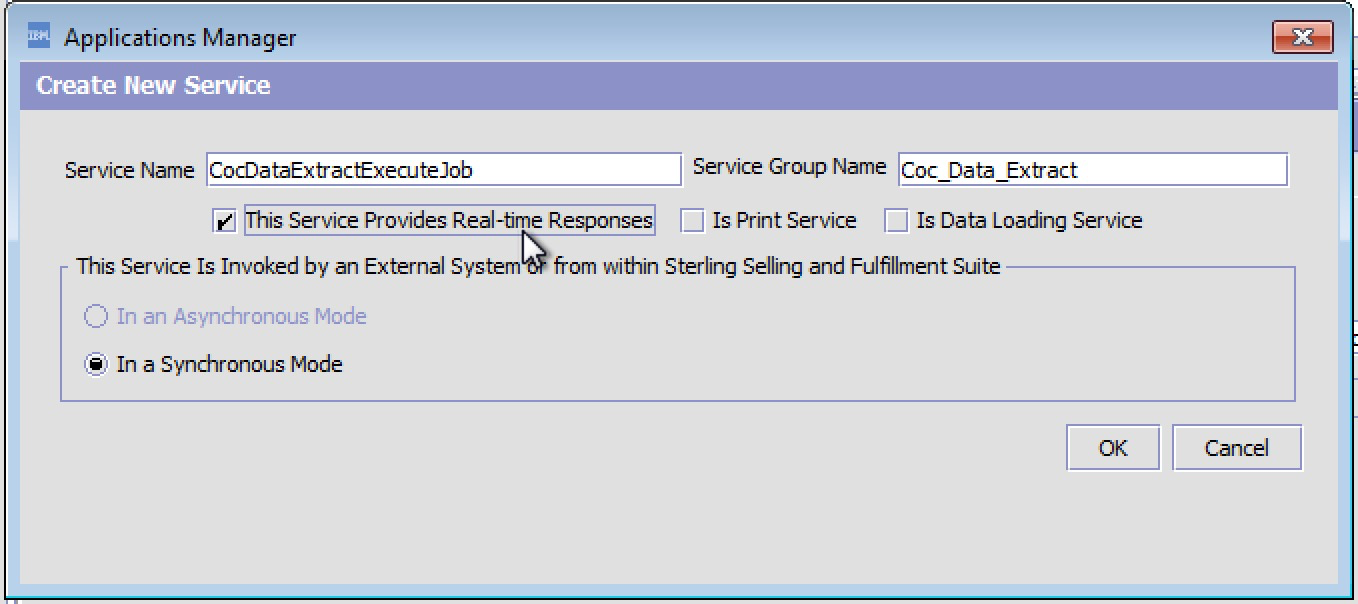
## 2.5 - Step 5 - Create a new Custom API to Call the Extract Worker API

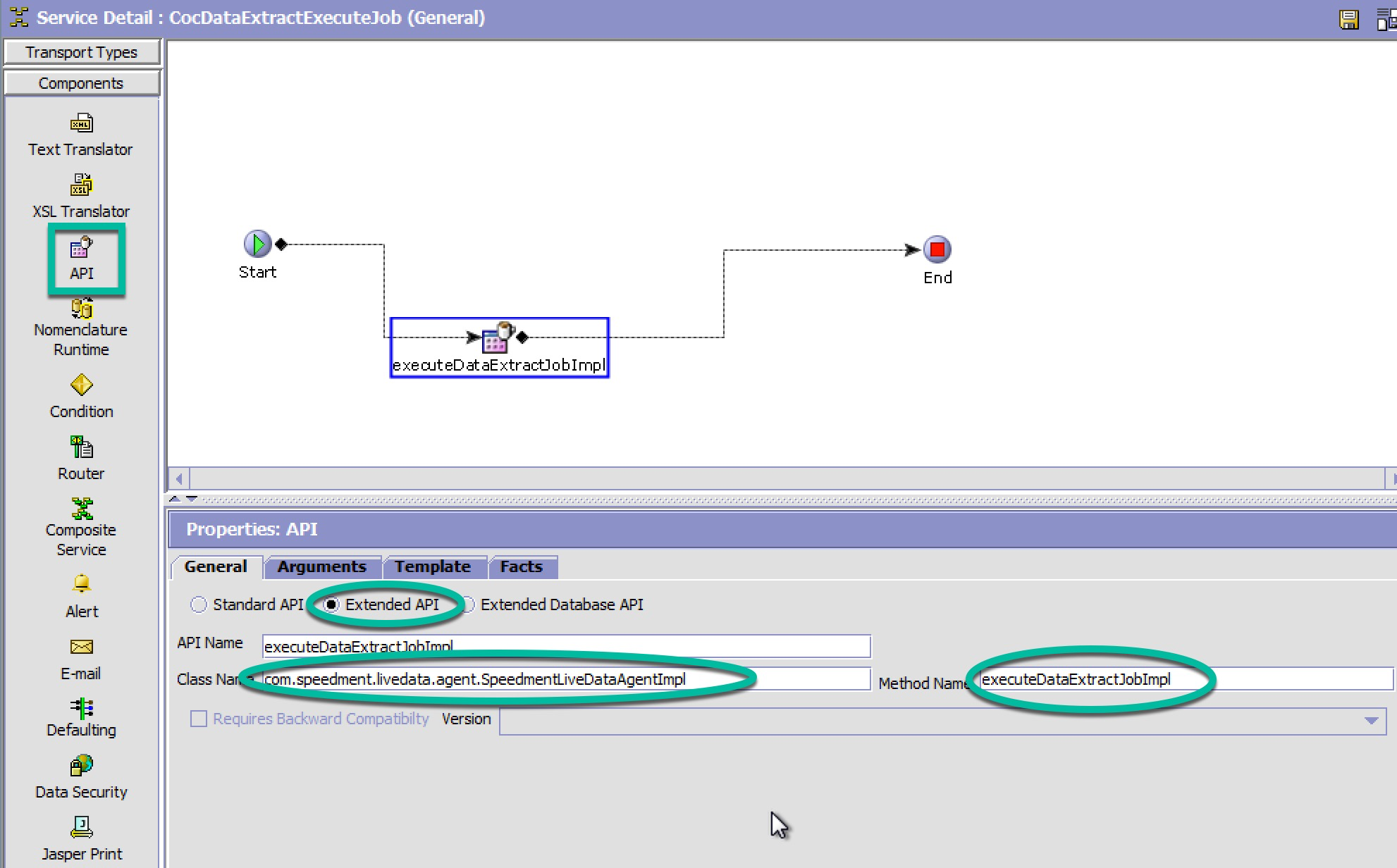
The Live Data Agent allows you to configure your own “**executeJob Worker**” API which will get called every time the Live Data Agent has a record to write to the target. The worker provided for you will write all the data records over a **Kafka** topic named “**speedment-topic**”, or write directly 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:



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