Smile CDR

Smile Guide: Channel Import Configuration Using Kafka

A guided journey on installing and using Smile CDR's Channel Import functionality



Table of Contents	
Table of Contents	1
What to Expect	2
Background	2
Objectives	2
Prerequisites	2
Installation Instructions for Windows (Docker) Zookeeper Installation Kafka Installation Changes to Your Smile CDR	3 3 4
Installation Instructions for MacOS Zookeeper and Kafka Download/Setup Changes to your Smile CDR	5 5 7
Channel Import Module Configuration on Smile CDR Web Admin Console	10
Kafka-Publisher	13
Glossary	17
Reference Links	18



What to Expect

Reading time = 1 hour

By the end of this document, you'll understand what Channel Import is, how to set up a channel import and how to use it to ingest data in real-time.

Background

Many health systems already have data stored in FHIR format that can be ingested into Smile CDR. While there are tools available, such as ETL Import and CSV Bulk Import, these rely on a static data source, which is often not how you want to have the data ingested. While various mechanisms exist to attempt to push data in real-time from one system to another, the Channel Import module aims to provide a channel-based method of ingesting FHIR data into Smile CDR. For more information please see this link on <u>Channel Import</u>.

Objectives

The intention of this document is to provide a step-by-step procedure of how to set up and use Smile CDR's Channel Import. You should be able to successfully complete the setup with little to no technical background on Channel Import.

Prerequisites:

- 1. See the following document on <u>installing and configuring Smile CDR</u>. This will need to be done prior to beginning this tutorial.
- NodeJs; if it's not installed, please consult the link here on how to install Node.js <u>on Windows</u> or <u>on</u> <u>MacOS</u>.
- 3. A text editor of your choice. We recommend Visual Studio Code, but Notepad (which comes preinstalled on Windows) and TextEdit (which comes preinstalled on Mac) work perfectly fine.
- 4. Knowledge of message brokers, Zookeeper and Kafka.
- 5. <u>Docker Installed</u>.
- 6. Kafka Publisher/Data Feeder to Kafka.



Installation Instructions for Windows (Docker)

Zookeeper Installation

- 1. To make use of channel import, we need to install Zookeeper on Docker. Please note that all commands are completed in the command prompt. To do this:
 - a. First, we have to create a network bridge (Confluent in this case). To do so, **copy and paste the following line into your command terminal** (continuing within your downloads folder), **then hit enter:**

docker network create confluent --driver bridge

b. Run the below command to load the Zookeeper image on Docker. To do so, **copy and paste the following line into your command terminal, then hit enter:**

```
docker run -d --net=confluent --name=zookeeper -e
ZOOKEEPER_CLIENT_PORT=2181 -p 2181:2181
confluentinc/cp-zookeeper:5.0.1
```

c. Next, in Docker, **click play** to the right of the zookeeper environment. You'll now see the container icon turn green; zookeeper is running. If it automatically appears green, this is also fine and there's no need to press play.

Kafka Installation

- 1. To make use of channel import, we need to install Kafka on Docker. Please note, step *a* is completed in the command prompt.
 - a. First we will load the Kafka Docker container. **To do so, copy and paste the following command into your command prompt**

```
docker run -d --net=confluent --name=kafka -e
KAFKA_ZOOKEEPER_CONNECT=zookeeper:2181 -e
KAFKA_ADVERTISED_LISTENERS=PLAINTEXT://<<Local IP address>>:9092
-e KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR=1 -p 9092:9092
confluentinc/cp-kafka:5.0.1
```

Note: The entire section highlighted in yellow must be replaced with your own local IP address. To find this address, **click on the** *Windows Start Icon* > *Settings* > *Network and Internet* > *Properties*. Scroll down and look for your IP address listed next to IPv4 address.



b. In Docker, **click play** to the right of the Kafka environment. You should see the container icon turn green, indicating that Kafka is running. If it automatically appears green, this is also fine and there is no need to press play.

Changes to Your Smile CDR

- 1. Stop your Smile CDR instance on Docker.
- 2. Copy the "classes" folder from the Smile CDR Docker container onto your local machine. To do so, **type the following line, then hit enter:**

docker cp smilecdr:/home/smile/smilecdr/classes ./

- 3. **Open File Explorer on your computer.** Within the Downloads folder, you should now notice a folder called "classes." **Open this folder.**
- 4. **Open the file called "cdr-config-Master.properties"** in any text editor (i.e. Notepad, VS Code, etc...).
- 5. Once this file is open, you'll notice that all sections of this file are separated by headings that begin with a "#." **Scroll down to the below section:**

Broker options are EMBEDDED_ACTIVEMQ, REMOTE_ACTIVEMQ, KAFKA, NONE

6. Remove the line currently present and replace it with the following (copy and paste):

module.clustermgr.config.messagebroker.type =KAFKA

module.clustermgr.config.kafka.bootstrap_address =kafka:9092

If you've successfully completed this step, the field should look like the image below:





- 7. Save this file by holding "ctrl" and "s" (don't change the directory to which it's saved).
- 8. Now we must copy back the classes folder from the Download's directory on your local machine to the Smile CDR Docker image. To do so, **copy and paste the following line into your command prompt, then hit enter:**

docker cp ./classes smilecdr:/home/smile/smilecdr/

9. Now put Smile CDR on the Confluent Docker network. **Copy and paste the following line into your terminal and click enter:**

docker network connect confluent smilecdr

Installation Instructions for MacOS

Zookeeper and Kafka Download/Setup

- 1. Make sure you've stopped running your Smile CDR instance.
- 2. Open this link https://kafka.apache.org/quickstart:
 - a. **Open the hyperlink** in Step 1 of the website:

STEP 1: GET KAFKA Download he latest Kafka release and extract it: 1 | \$ tar -xzf kafka_2.13-3.0.0.tgz 2 | \$ cd kafka_2.13-3.0.0



b. **Click on the suggested link** at the top of the following page in order to download a zip folder:

APACHE	СОМ	MUNITY-LED	DEVELOPMENT	"THE APACHE	WAY"	SPORT .
SOFTWARE FOUNDATION ESTABLISHED 1999	Projects -	People -	Community +	License 👻	Sponsors -	APACH
We suggest the following site for y	our download:	-				
https://dlcdn.apache.org/kafka/3	.0.0/kafka_2.13-3.0.0.tgz					
Alternate download locations are s	suggested below.					
It is essential that you verify the in	tegrity of the downloade	d file using the PG	P signature (.asc file) or	rahash(.md5 or .s	ha* file).	
HIIP https://dlcdn.apache.org/kafka/3 BACKUP SITE	.0.0/kafka_2.13-3.0.0.tgz					
https://dicdn.apache.org/kafka/3	.0.0/kafka_2.13-3.0.0.tgz					
VERIFY THE IN It is essential that you verify the in Software Foundation Releases for	TEGRITY O tegrity of the downloade more information on wh	F THE FI d file using the PG yyou should verify	LES P signature (.asc file) or our releases.	rahash(.md5 or .s	iha∗ file). Please read \	erifying Apache
The PGP signature can be verified	using PGP or GPG. First	download the KEY	s as well as the asc sign	hature file for the rel	evant distribution.	
% gpgimport KEYS % gpgverify downloaded_fil	le.asc downloaded_file					
or						

- c. **Extract the zip folder** into your Downloads folder.
- 3. Open a new terminal window and navigate to your Downloads folder, then the Kafka folder you extracted in Step 2.c. To do so, **run (copy and paste) the following commands:**



cd kafka_<mark>x.x-x.x.x</mark>

Note: the section highlighted yellow will be replaced with whichever version number you downloaded, in this case "2.13-3.0.0." If you ran these commands correctly, your terminal should look similar to the image below:



4. To start Zookeeper, **copy and paste the following command into your terminal, then press enter:**

bin/zookeeper-server-start.sh config/zookeeper.properties

- 5. Leave this terminal window open.
- 6. In a new terminal window, **navigate to your Downloads folder**, **then the Kafka folder you extracted in Step 2.c** (you can use the same commands as in Step 3):



7. To start the Kafka server, run the following command:

bin/kafka-server-start.sh config/server.properties

```
Daniels-MacBook-Pro-2:~ danielmoriana$ cd downloads
Daniels-MacBook-Pro-2:downloads danielmoriana$ cd kafka_2.13-3.0.0
Daniels-MacBook-Pro-2:kafka_2.13-3.0.0 danielmoriana$ bin/kafka-server-start.sh
config/server.properties
```

8. Leave this terminal window open.

Changes to your Smile CDR

 In a new terminal window, navigate to the directory of your Smile CDR. In this example, it's located in the computer's Downloads folder. Once complete, navigate to your Smile CDR's Classes folder:

> [Daniels-MacBook-Pro-2:~ danielmoriana\$ cd downloads [Daniels-MacBook-Pro-2:downloads danielmoriana\$ cd smilecdr [Daniels-MacBook-Pro-2:smilecdr danielmoriana\$ cd classes

2. Copy and paste the following command into your terminal, then hit enter:

vi cdr-config-Master.properties

[Daniels-MacBook-Pro-2:classes cdr-config-Master.properties	danielmoriana\$ ls config_seeding	s smilecdr-demo.jwks
cdr-messages.properties	fhir_gateway	
client_certificates	logback.xml	
Daniels-MacBook-Pro-2:classes	danielmoriana\$ vi	i cdr-config-Master.properties

- 3. Once the file is open, **hit the "i" key** to change to insert mode where you can now edit the file.
- 4. Locate the following property in the file:

module.clustermgr.config.messagebroker.type =EMBEDDED_ACTIVEMQ



Channel Import—Kafka

```
*****
# Node Configuration
node.id
                                                                       =Master
# Cluster Manager Configuration
****************
# Valid options include H2_EMBEDDED, DERBY_EMBEDDED, MYSQL_5_7, MARIADB_10_1, POSTGRES_9_4, ORAC
LE_12C, MSSQL_2012
module.clustermgr.config.db.driver
                                                                      =H2_EMBEDDED
module.clustermgr.config.db.url
                                                                      =jdbc:h2:file:./database/h2_clust
ermgr
module.clustermgr.config.db.username
                                                                       =SA
module.clustermgr.config.db.password
module.clustermgr.config.db.schema_update_mode
                                                                      =SA
                                                                       =UPDATE
module.clustermgr.config.stats.heartbeat_persist_frequency_ms
module.clustermgr.config.stats.stats_persist_frequency_ms
                                                                      =15000
                                                                      =60000
module.clustermgr.config.stats.stats_cleanup_frequency_ms
                                                                      =300000
module.clustermgr.config.messagebroker.type =EMBEDDED_ACTIVEMQ
# Request headers to store in the audit log
module.clustermgr.config.audit_log.request_headers_to_store=Content-Type,Host
******
# The following setting controls where module configuration is ultimately stored.
# When set to "DATABASE" (which is the default), the clustermgr configuration is
# always read but the other modules are stored in the database upon the first
# launch and their configuration is read from the database on subsequent
# launches. When set to "PROPERTIES", values in this file are always used.
# In other words, in DATABASE mode, the module definitions below this line are
# only used to seed the database upon the very first startup of the sytem, and
# will be ignored after that. In PROPERTIES mode, the module definitions below
# are read every time the system starts, and existing definitions and config are
# overwritten by what is in this file.
node.propertvsource
                                                                      =DATABASE
*****
# Database Configuration
**********************
module.persistence.type
module.persistence.config.db.driver
                                                                       =PERSISTENCE_R4
                                                                      =H2 EMBEDDED
module.persistence.config.db.url
                                                                      =jdbc:h2:file:./database/h2_fhir_
persistence
module.persistence.config.db.hibernate.showsql
                                                                      =false
module.persistence.config.db.username
module.persistence.config.db.password
module.persistence.config.db.hibernate_search.directory
                                                                      =SA
=SA
                                                                      =./database/lucene_fhir_persisten
се
module.persistence.config.db.schema_update_mode
                                                                      =UPDATE
module.persistence.config.dao_config.expire_search_results_after_minutes=60
module.persistence.config.dao_config.allow_multiple_delete.enabled=false
module.persistence.config.dao_config.allow_inline_match_url_references.enabled=false
# Subscription
=SUBSCRIPTION_MATCHER
module.subscription.type
module.subscription.requires.PERSISTENCE_ALL
                                                                      =persistence
-- INSERT --
```

5. Change this property value from "EMBEDDED_ACTIVEMQ" to "KAFKA" (see below):

module.clustermgr.config.messagebroker.type =KAFKA



Channel Import—Kafka

*****	****
# Node Configuration	
***************************************	****
node.ld	=Master
********	*****
# Cluster Manager Configuration	
***************************************	*****
# Valid options include H2_EMBEDDED, DERBY_EMBEDDED, MYSQL_5_7	, MARIADB_10_1, POSTGRES_9_4, ORAC
LE_12C, MSSQL_2012	
module.clustermgr.config.db.url	=H2_EMBEDDED =idbc:h2:file:/database/h2_clust
ermgr	_jabothizi i i i i i i i i i i i i i i i i i i
module.clustermgr.config.db.username	=SA
module.clustermgr.config.db.password	=SA
module.clustermgr.config.db.schema_update_mode	=UPDATE
module.clustermgr.config.stats.neartbeat_persist_frequency_ms	=12000
module.clustermgr.config.stats.stats_persist_frequency_ms	=300000
modulofoldstelmg1foonflgfoldtofoldto_oldendp_floquency_mo	
# Broker options are EMBEDDED_ACIIVEMQ, REMULE_ACIIVEMQ, KAFKA	, NOT E
module.clustermgr.config.messagebroker.type =KAF	KA
H Desurat bandana ta atana in tha audit ina	
# Request headers to store in the audit log	ntent-Type Host
module.clustermgi.comig.addit_log.request_neadels_to_store=co	ntent-Type, nost
***************************************	*****
# Other Modules are Configured Below	
***************************************	******
# The following setting controls where module configuration is	ultimately stored.
# When set to "DATABASE" (which is the default), the clusterma	r configuration is
# always read but the other modules are stored in the database	upon the first
# launch and their configuration is read from the database on	subsequent
# launches. When set to "PROPERTIES", values in this file are	always used.
#	
# in other words, in DATABASE mode, the module definitions bei	ow this line are f the sytem and
# will be ignored after that. In PROPERTIES mode, the module d	efinitions below
# are read every time the system starts, and existing definiti	ons and config are
# overwritten by what is in this file.	
#	
node.propertysource	=DATABASE
*****	****
# Database Configuration	***************
*********	*****
module.persistence.type	=PERSISTENCE_R4
module.persistence.config.db.driver	=H2_EMBEDDED
module.persistence.config.db.url	=jdbc:h2:file:./database/h2_fhir_
persistence	-6-1
module.persistence.config.db.nibernate.snowsqi	=======================================
module.persistence.config.db.password	=SA
module.persistence.config.db.hibernate_search.directory	=./database/lucene_fhir_persisten
ce	
module.persistence.config.db.schema_update_mode	=UPDATE
module.persistence.config.dao_config.expire_search_results_aft	er_minutes=60
module.persistence.config.dao_config.allow_multiple_delete.ena	bled=false
module.persistence.config.dao_config.allow_inline_match_uri_re	enabled-false
module.persistence.conitg.dao_conitg.allow_external_references	
*******	****
# Subscription	
***************************************	*****
module.subscription.type	=SUBSCRIPTION_MATCHER
module.subscription.requires.PERSISTENCE_ALL	=persistence
INSERT	

6. Once the change is done, save the file by **pressing the "esc" key, then type ":wq" and hit enter**.



Channel Import Module Configuration on Smile CDR Web Admin Console

1. Click the following link to open the Smile CDR web admin console:

http://localhost:9100/signin

Click the green "Sign In" button and fill out the prompt page with the following information:

- i. Username: *admin*
- ii. Password: *password*
- b. On the home page, click on the "Add Module of Type" drop down menu, then select "Channel Import" and click on the green "add" button.

s smile cdr	合 Home 🏼 ✿ Config ◄						
OC Configuration							
Cluster				Configuration			
Overview				-			
Node: Master				This section may be used deployment	to configure storage modules, endpo	ints, security, and other features. Smile CDR is divided into modules	that are individually configured according to the needs of the
clustermgr	Cluster Administrative Modules		0				
admin_json		9000	0	Manage Node Modules			/
fhir_endpoint fhirweb_endpoint package_registry persistence subscription	FHIR Modules	8000 8001 8002	0 0 0 0 0	A node is a single running Add Module of Type C	g process, and it hosts any number of r Channel Import I ES	nodules. These modules perform individual functions, and they are o	onfigured and wired together via their individual configurations.
local_security	Security Modules		ø	Node ID	Module ID	Description	Status
smart_auth		9200	Ø	Master	clustermgr	Cluster Manager	⊘ Running
				Master	admin_json	JSON Admin API	⊘ Running
				Master	admin_web	Web Admin Console	⊘ Running
				Master	fhir_endpoint	FHIR REST Endpoint (R4)	⊘ Running
				Master	fhirweb_endpoint	FHIRWeb Console	⊘ Running
				Master	package_registry	Package Registry Endpoint	Ø Running
				Master	persistence	FHIR Storage (R4 Relational)	⊘ Running
				Master	subscription	Subscription Matcher (All FHIR Versions)	⊘ Running
				Master	local_security	Local Inbound Security	⊘ Running
				Master	smart_auth	SMART Outbound Security	⊘ Running



2. The next step is to configure the channel import module. To do this, **set the configurations to the specifications below:**

Configuration	Sample Value	Description
Channel Name	import	The name of Kafka topic or ActiveMQ queue where inbound resources will be arriving.
Concurrent Consumers	3	The number of the concurrent consumers.
Concurrent Retry Consumers	3	The number of concurrent consumers on the retry channel.
Default mediaType	application/fhir+json	If set, applies the mediaType to incoming messages that are missing the mediaType attribute. Legal values are text/plain, text/csv, application/fhir+json, application/json. Defaults to application/fhir+json.
Retry Channel Name	retry	The name of Kafka topic or ActiveMQ queue where inbound resources are sent when a failure occurs during processing of an incoming resource. Non-null value required for retry to be enabled.
Retry Delay(ms)	1000	The minimum amount of time to wait (milliseconds) between retry attempts.
Failure Channel Name	failed	The name of Kafka topic or ActiveMQ queue where resources are sent after they have exceeded the maximum number of retry attempts, and have still not been successfully processed.



Maximum Delay(ms) between attempts.	1000	The maximum amount of time to wait (milliseconds) between retry attempts. This provides an upper limit for exponential backoff.
Maximum amount of retry attempts.	1	The maximum amount of times to attempt import before considering a message failed. Non-zero value required for retry to be enabled. If set to zero, failed messages will skip the retry channel completely and go directly to the failure channel.
FHIR Storage Module (any FHIR version)	persistence (FHIR Storage (R4 Relational))	The FHIR Storage module to associate with this module.

3. At the top of the page, **click "Save," then "Start"** to begin the module (See Figure 3).

Calce State S	Smile CDR #Home + Cordg + #Runtime + Documentation	n Øsapport	(Healthy :)	Admin 🛓 🔻
Claster Create Modele Node: Master Catter Administrative Models Create Modele D. creaned import Administrative Models Model D. creaned import Administrative Models Dimonstrative Models Model D. creaned import Models D. creaned import Models D. creaned import Import Model D. creaned import	OC Configuration / OC Create Module			
persistion: Curve import isdo-rgino/ isdo-rgino/ channel/layort 200 Channel Marcel The material fails or Athandba gave after isdowed measurem will be arring: Channel Aleryor Channel Aleryor Despendencies 200 The material fails or Athandba gave after isdowed measurem will be arring: Show All Concurrent Comments Import Channel Morry Concurrent Comments Concurrent Comments Concurrent Comments Concurrent Comments Concurrent Comments Concurrent Comments Concurrent Comments Concurrent Comments Concurrent Comment The material attack or Athandbage and the induction attack or Athandbage	ে বিদয়সৈয়া ' বি' Area Hotalia Torretaria National Sector Sect	Create Model Create new model Medic D_connet_import Dismo Channel Import Concurrent Consumer + During Marcine During Marcine During Marcine Concurrent Consumer + Build Import During Marcine Import Import Important transment to the interpretant of the second to the interpretan		



Configuration / Configuration	s channel_import				
Cluster Overview				Node Master > Module channel_import	Type Channel Impor
Node: Master				Process Module Status Timestamp / Description	
clustermgr admin_json admin_web	Cluster Administrative Modules	9000 9100	0 0	Presquite @ Bunning © Stopped 11.39:40	
 channel_import Channel Import Channel Retry Dependencies 			۲	Jung to: Module suntune satus Channel Import	
Show All				Channel Name 🕢 The name of Kafka topic or ActiveMQ gusue where inbound resources will be arriving.	
thir_endpoint thirved_endpoint persistence subscription local_security smart_auth	Security Modules	8000 8001 8002 9200	0 0 0 0 0	Concurrent Consumers Concurrent Active of the concurrent consumers as the retry channel. Concurrent Retry: The number of accounter consumers as the retry channel. Concurrent Retry: Concurrent Retry: Channel Import ETL.Script If the supplies an ETL pressure used for ortifun interlogues in the Channel Inport Strates Channel Import ETL.Script If the supplies an ETL pressure add for ortifun interlogues in the Channel Inport Strates Channel Import ETL.Script If the supplies an ETL pressure add for ortifun interlogues in the Channel Inport Strates Channel Import ETL.Script If the supplies an ETL pressure add for ortifun interlogues in the Channel Inport Strates Channel Import ETL.Script If the supplies an ETL pressure add for ortifun interlogues in the Channel Inport Strates Channel Import ETL.Script If the supplies and ETL pressure add for ortifun interlogues in the Channel Inport Strates Channel Import ETL.Script If the supplies and ETL pressure add for ortifun interlogues in the Channel Import Strates Channel Import ETL.Script If the supplies and ETL pressure add for ortifun interlogues in the Channel Import Strates Channel Import ETL Script If the supplies and ETL pressure add for ortifun interlogues in the Channel Import Strates Channel Import ETL Script If the supplies and ETL pressure add for ortifun interlogues add (Strates) a	

- 4. We must now grant the Anonymous user with Super User Permission from your Smile CDR Web Admin GUI. To do so:
 - a. Click on the "Config" drop-down menu, then select "User Manager."

\rightarrow C ()	localhost:9100/config/usermg	ır/					
≸ smi	ecdr & Home	✿Config ▼ ◀Runtime ▼ ◎I	Documentation 🛛 😨 Supp	ort			
ser Manager		🏟 Module Config					
lsers		😩 User Manager Q Search Parameters					
Q Search	(Username or Family Name)	 d OpenID Connect Clients d OpenID Connect Servers ■ Config Diagnostics 	local_security (Master)	✓ Add User			
	Module	Username	Family Name	Given Name	Last Active Date	2FA	Roles and Permissions
Modify	Node: Master Module: local_security	ADMIN	GenericUser	Admin	2021-10-08	No	Superuser
Modify	Node: Master Module: local_security	ANONYMOUS	Anonymous	Anonymous	2021-10-08	No	

- b. From the list of users, search for "ANONYMOUS" and click on "Modify."
- c. From the "Roles and Permissions" section, **scroll down to "ROLE_SUPERUSER" and click on "YES"** to enable the permission.



Superuser User has all permissions to do anything. Use this permission with caution since it gives the user access to almost all administrative functions.

d. Scroll to the top of the page and click "save."



Kafka-Publisher

- 1. We need to publish data to the Kafka message broker. To do this, we've created a basic Kafka publisher simulator with GUI. **Download the following <u>zipped folder</u>**.
- 2. Unzip the "kafka-publisher.zip" to your Downloads folder.
- 3. Navigate to the Downloads folder by running the following command:

cd Downloads

4. Navigate to the Kafka-publisher zip folder by running the following command, then click enter:

cd kafka-publisher

5. Next we will install the required npm/JavaScript Packages. To do so, **copy and paste the following into your command prompt, then click enter:**

npm install

6. Now we must start the Kafka publisher simulator. To do so, **copy and paste the following command into your command prompt, then click enter:**

npm run start

Once it starts, your command line should look like this:



7. Open the following link: <u>http://localhost:4001</u>



8. You should see a new page that allows you to create, update and delete records using Kafka.

Kafka publisher Simulator	
Make sure you have configured the kafka endpoint and topic name correctly in the default, son file	
Operation Type	
CREATE	~
For 'CREATE' operation if from resource will be ignored auto generated For 'UPDATE' and 'DELETE' Operations resource must contain id	
Resource FHIR-JSON	
E = = I ▼ 🖋 つ C Code -	powered by ace
10 "system": "http://terminology.hl7.org/CodeSystem/v2-0203", 11 "code": "HR" 12), 13] 14),	•
<pre>15 "system": "unridd:0.1.2.3.4.5.6.7", 16 "value: "122467" 17) 18], 19 "active": true, 20 " "name": [21 - (21 - (22 - "sem: [ficial", 23 - "family": "hotswell", 24 - "simen": [25 - ["simen" 25 - ["simen" 26 - [] 27 -] 27 -] 28 -] 29 - ["simen" 29 - ["simen" 20 - ["simen" 20 - ["simen" 21 - ["simen" 22 - ["simen" 23 - ["simen" 24 - ["simen" 25 - [] 26 - [] 27 -] 27 -] 28 -] 29 -] 20 -] 20 -] 20 -] 20 -] 21 - [[[[[[[[[[[[[[[[[[</pre>	

9. The GUI page comes with a sample FHIR-JSON resource. Select the Operation Type as "CREATE" and click on "Submit." This shows that the message was successfully published to Kafka and should display the same message as the screenshot below:

→ C O localhost4001	C \$
Kafka publisher Simulator Make sure you have configured the kafka endpoint and topic name correctly in the default joon file	
Message published to Kafka Successfully (Operation :CREATE)	
Operation Type	
CREATE	
for URDER the add CLLET Contrainer Head contain at the second se	powers
10 "system": "http://terninolog/.h17.org/CodeSystem/v2-0203", 11 "code": "Wit" 12 } 13] 14 }.	
5 - ⁵ ₂ yster*, ¹ worodd(0.1.2.3.4.5.6.7°, 5 - ⁵ ₂ yster*, ¹ 23457 3 1 10 - 1 30 *active: true, 28 *mee*: [20 C 21 C	
<pre>23 "family": "Notsovell", 24 "given"; [</pre>	
26 1	

Note: If this property in the Smile persistence module isn't enabled, it'll throw an error message. To resolve either, set the property to "true" or remove the following code from the Kafka-publisher and submit:



Code (line 32-34):

"managingOrganization": {

```
"reference": "Organization/1"
```

- }
- 10. To validate if the Smile CDR has consumed that resource from the Kafka broker, hit the FHIR endpoint and check to see if the total number of resources increased to 7 from 6.
 - a. To do this, **open the following link and check how many patients our Smile CDR FHIR endpoint is returning:** <u>http://localhost:8000/Patient</u>
 - b. Since a new resource was added in Step 9, you'll notice that the total number of resources is 1. If a resource was not added successfully, the total number of resources will display 0.



11. This confirms that our Channel Import module set up works fine end-to-end with Smile CDR, Zookeeper and Kafka running on a Docker.





Glossary

Channel: a medium through which you can send a message to a destination. When two applications wish to exchange data, they do so by sending the data through a channel that connects the two. The application sending the data may not know which application will receive the data, but by selecting a particular channel to send the data on, the sender knows that the receiver will be one that's looking for that sort of data by looking for it on that channel. In this way, the applications that produce shared data have a way to communicate with those that wish to consume it.

Kafka: perhaps the most popular modern message broker. Kafka is open source and used at almost every web scale company. At Smile CDR we use both Kafka and Active MQ, but for the purposes of simplicity, will be using Kafka for this Smile Guide. For more information on Kafka see this link on <u>Apache Kafka</u>.

Message Broker: also known as an integration broker or interface engine. It's an intermediary computer program module that translates a message from the formal messaging protocol of the sender to the formal messaging protocol of the receiver. Essentially, it enables applications, systems, and services to communicate with each other and exchange information. Message brokers are elements in telecommunication or computer networks where software applications communicate by exchanging formally-defined messages. Message brokers are a building block of message-oriented middleware (MOM) but are typically not a replacement for traditional middleware like MOM and Remote Procedure Call (RPC).

Messaging Queue: a form of asynchronous service-to-service communication used in serverless and microservices architectures. Messages are stored on the queue until they're processed and deleted. Each message is processed only once, by a single consumer.

Zookeeper: primarily used to track the status of nodes in the Kafka cluster and maintain a list of Kafka topics and messages. For more information, see this link on <u>ZooKeeper</u>.



Reference Links

1. Smile CDR & Docker Installation Guide

https://docs.google.com/document/d/1rlg0jf6E8WFphGvbro8GUwEvKJZ1geV1/edit#

2. Node JS - Windows Installation

https://www.youtube.com/watch?v=AuCuHvgOeBY&t=53s

3. Demo on Channel Import- By Gary Graham

https://vimeo.com/510491999

(In the video if Gary refers to any file for sample code or installation instruction that should be inside below repo.)

https://gitlab.com/smilecdr-public/feature-walkthroughs/-/tree/master/channel_import

4. Scripts/Commands to Run Zookeeper, Kafka and Create Network Bridge on Docker <u>https://gitlab.com/smilecdr-public/feature-walkthroughs/-/tree/master/channel_import/setup</u>

5. Sample Patient Resource

https://www.hl7.org/fhir/patient-examples.html

6. Apache-Kafka Download

https://kafka.apache.org/quickstart

Smile CDR

Smile CDR Inc.

622 College Street, Suite 401 Toronto, Ontario M6G 1B4, Canada info@smilecdr.com 1 (800) 683-1318

www.smilecdr.com

Copyright @2021 Smile CDR Inc

All product names, logos, and brands are the property of their respective owners. All company, product and service names used are for identification purposes only. The use of these names, logos, and brands does not imply endorsement.

Version: 1.0 Last Updated: November 15, 2021

Principle Author: Kishore Kulkarni