iQMS Audit Trail - feeding the data into Audit Trail

Status	IN PROGRESS
Owner	@Sławomir Pawluk
Contributors	
Approved	
Due date	
Decision	
On this page	

❓ Problem statement

The changes to Records have to be audited. Each change (single or bulk) needs to be recorded in order to determine who and when updated the data

💡 Research insights

📊 Solution hypothesis

- Ideally the Audit Trail data feeding should be asynchronous
- Cost of the solution is important
- Audit trail of changes happened before this feature is deployed can be ignored
- Audit Trail should not increase API response times

Requirements:

- Each **Audit Trail Entry** must be related to **Record** - a tracked business entity
- Each **Audit Trail Entry** must contain a **User** who made an entry
- Each **Audit Trail Entry** must contain a **Date** when an entry was made
- Actions that create **Audit Trail Entry** are:
  - Create a record
  - Update a record
  - Delete a record
  - View a record
- Each **Audit Trail Entry** for a **Created** entry must contain list of properties and its values, when entry was created
- Each **Audit Trail Entry** for a **Updated** entry must contain list of properties, its values and values that were replaced by update
- Each **Audit Trail Entry** for a **Deleted** and **Viewed** entry does not contain properties of the **Record**

🌈 Design options

There are 3 classes of solutions that can be applied:

- Outside of process data gathering like CDC, temporal tables (history tables), scheduled jobs, direct access to service database, Debezium
- Async messaging using varying transport options (Azure Service Bus, Storage Queues, Kafka, MSMQ, RabbitMQ, SQL Database, etc.)
- RPC calls (HTTP, gRPC, GQL, etc.)

		Option 1	**Option 2**	**Option 3**
Overview		CDC	NServiceBus + Azure Service Bus	API (REST, GQL, gRPC)
Link		Change data capture (CDC) with Azure SQL Database - Azure SQL Database	NServiceBus	
Benefits and risks		⊕ Does not require changes in iQMS code ⊖ Change of database schema can break Audit Trail (depends on configuration and implementation) ⊖ Azure SQL limitation - minimum tear S3 ($150+/instance/month) or vCores Provisioned ($450+/instance/month - compute + license) ⊖ Cannot handle “Record viewed” use-case ⊖ Requires tedious configuration per table ⊖ Low level data that requires parsing and merging	⊕ Developers have full control over what is saved into Audit Trail ⊕ Asynchronous operations ⊕ Can be reused to split services into separate deployment units ⊖ Each change have to be explicitly recorded by sending a message to Audit Trail component ⊖ NServiceBus license required ⊖ Azure Service Bus - additional cost ($0.0121/h - $9/month - 13M operations/month included) ⊖ Azure Service Bus does not have local emulator. Change of transport (use RabitMQ locally) or shared Azure Service Bus instance required (additional cost)	⊕ Well known by team members ⊖ Not scalable ⊖ Distributed transaction - prone to crashes ⊖ Requires synchronous operations
Criteria	**Cost**	Database cost increases to $150-450/month minimum/environment	NServiceBus library - $250/10 endpoints/100k messages/month. We can start from FREE Community $0/month Azure Service Bus Starting from $10/month/environment - 13M operations/month included - 13-100M - $0.80/1M - 100-2,500M $0.45/1M	No additional external cost Cost of development unknown
	Code changes	No changes required	Every time record is changed, a message must be sent	Every time record is changed, a request must be sent
	Maintainability	• Low level SQL feature that requires good understanding of documentation • Errors may be hard to track due to limited monitoring capabilities	• New pattern to be learned by team • Can be reused to split services • Can be used to implement Edit Mode using Sagas	• Not scalable • Will reduce overall system performance

Option 1 - CDC

CDC - Change Data Capture - a SQL Server feature that enables the identification and tracking of changes in database tables. It captures and records insert, update, and delete operations, providing a reliable source of change information. Leveraging CDC allows us to identify modified data efficiently and transmit it in real-time to downstream systems.

To enable CDC:

1. Enable CDC in database

```sql
EXEC sys.sp_cdc_enable_db;
GO
```

2. Enable CDC per table:

```sql
EXEC sys.sp_cdc_enable_table
    @source_schema = N'SchemaName',
    @source_name = N'TableName',
    @role_name = NULL;
GO
```

Parameter `captured_column_list` can be used to limit columns that are captured.

3. Invoke `cdc.fn_cdc_get_all_changes_<capture_instance>` (docs) or `cdc.fn_cdc_get_net_changes_<capture_instance>` (docs) to get a table representing recorded changes
4. The result must be parsed and can be used as a data source.

Performance considerations

- Number of tables with CDC enabled
- Frequency of changes in those tables
- Additional space required
- Single database / elastic pool database

Debezium can be used to generate event streams from SQL Database. Using Debezium requires to add specified components to application infrastructure

- Kafka
- Zookeeper
- Debezium

On top of those additional services SQL Database tear must be upgraded per database instance increasing the cost of the solution and increasing cost when splitting up to separate services.

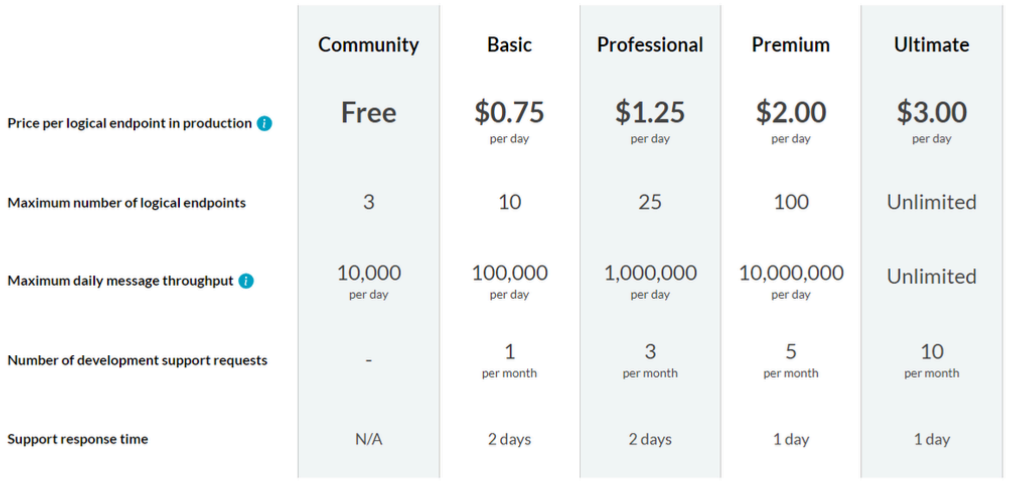
Option 2/3 - NServiceBus varying transport

NServiceBus is an industry standard library that helps to implement messaging in distributed system. It supports multiple transport mechanisms, persistence of messages, SAGAs, fault-tolerance and more.

Since NServiceBus publishes great documentation I link it here:

- NServiceBus Introduction
- NServiceBus SAGAs Introduction

Pricing



	Community	Basic	Professional	Premium	Ultimate
Price per logical endpoint in production	Free	$0.75 per day	$1.25 per day	$2.00 per day	$3.00 per day
Maximum number of logical endpoints	3	10	25	100	Unlimited
Maximum daily message throughput	10,000 per day	100,000 per day	1,000,000 per day	10,000,000 per day	Unlimited
Number of development support requests	-	1 per month	3 per month	5 per month	10 per month
Support response time	N/A	2 days	2 days	1 day	1 day

Option 3 - REST API/GQL/gRPC

Well known pattern, different protocols to transfer the data.

This method of data feeding is synchronous. It would require to implement connectivity between services, fault-tolerance, transactionability.

After user clicks the button on interface, not only they had to wait for data changes to be saved, but also Audit Trail to be send over network and saved. This hooping over to different services is bad design practice that in long run causes performance problems.

✅ Follow up

Decision	Status	Next steps
Review possibility of migrating EditMode implementation to SAGA	PROPOSED	☐
Automate CDC configuration - enable CDC and configure tables on deployment	PROPOSED	

💎 Source files

Type /link to add links to design files.