Application Programming Interface-Digital Strategy for Core Banking

Arindam Roy

B. E (Jadavpur University, Mechanical), MBA (Systems & Finance, MDI-Gurgaon)
Email: *arindamr34[at]gmail.com*
Contact no: 9831483728

Abstract: ***Once regarded as a technical interface, the humble API is now praised as a strategic business asset that must be taken seriously. Around the world, banks are awakening to the transformational potential of APIs as the core building block of open banking. Here we consider APIs in the context of channel development and facilitating an "opti-channel" customer experience. By developing and selling access to new API products, banks are able to create additional direct revenue streams. These premium APIs can also be used as up-sells or cross-sells for other banking products (such as certain corporate accounts). APIs empower banks to deliver products and services in context when customers need them. As well as integrating bank products into third-party apps and services, APIs allow banks to disaggregate the banking value chain. This work provides a brief window into how API driven architecture is driving the digital transformation for banks.***

Keywords: APIs, digital transformation, banking, open banking, customer experience

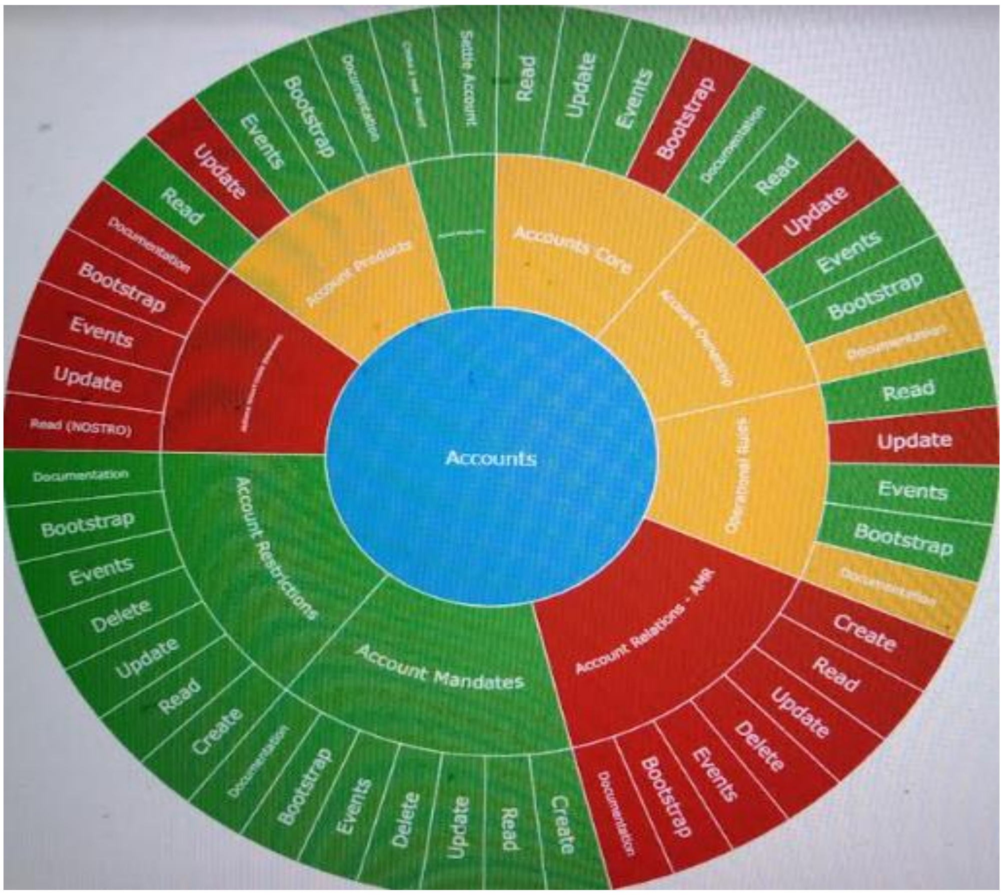
Research

Data in the domains such as Account, Customers and Product is the key for any bank and hence exposure of the data through APIs and Events facilitates a bank to function better.

Account Core API contains basic information of an account including Account Id's, related Product Id, account currency, account holding branch, etc. It allows retrieving and updating basic account information by calling Get Account Core API and Update Account Core API

There is a close relationship between the Account Domain and the Account product. A product defines a configuration of various conditions (also called product rules) for an account based on that product. In other words, product conditions are inherited to any account instance of that particular product.

Account domain offers APIs on Account Core, Account Ownership, Account mandates, Account Restrictions, Operational Rules, Open Account, Account Closure, Product Information an Account Relations



API Endpoints

Operation	Endpoint method	Endpoint Path	Description
Get Core account information	GET	/accounts/ [account Id]	Retrieves basic account details based on the provided account Id (anonymous account id or UUID). Basic information consists of different account keys, country and currency code, branches and product relations
Get customers accounts	GET	/customers/ [anonymous Customer Key]/accounts	Retrieves basic account details for all accounts owned by the customer identified by the proved anonymous Customer Key e. g different account keys, country and currency code, branches and product relations
Update branch relation for account	PUT	/account/ [account Id]/account-holding-branch	Update the branch relation of the account (account holding branch) to another branch
Change the product for account	PUT	/accounts/ [account Id]/product-id	Update the product relation on the account (change the product on which the account is based
Update the account name for the account	PUT	/account/ [account id]/account-name	Updates the account name based on the account Id
Update statement text for an account	PUT	/accounts/ [account Id]/statement-text	Update statement text based on the Account Id
Delete Statement Text for an account	DELETE	/accounts/ [account Id]/statement-text	Delete Statement Text based on the account Id
Delete account holding branch updated value for an account	DELETE	/accounts/ [account Id]account-holding-branch-update	Delete account holding branch updated value for an account based on the account t id

Account Core Bulk request is an asynchronous process to retrieve basic account details for the provided list of account ids. The bulk request contains list of accounts across customers for which the user wants to retrieve account details

Attribute /Parameter	Description	Mapping to mainframe attribute	Required
Account Id	ID of type UUID to uniquely identify an account (anonymous account id)		Yes
Internal Id	Internal unique identifier of an account	IDKT	Yes
National Id	External account identifier (customer known). in many cases in DK, the national Id is the same as the internal Id	E_IDKT	Yes
Country Code	The country code of the country the account is being operated in	KONTOLAND_KD	Yes
iban	IBAN (International Banking Account Number) of the account	IBAN	Yes (not required for internal accounts)
Domestic Id	The “preferred” external identification of the account in the country of operation	SEKTOR_IDENT	Yes
Domestic ID Type	The type of the domestic ID can be either IBAN or National	NA	Yes
type		KTTP	Yes
Opening Date	Creation date of the account in the bank	ETDT	Yes
Closure Date	Settlement /closure of the account depending on the account status	DTOPUD	No
Status	Account status • Active • Settled. – The account is about to be closed. When balance reaches 0.0, the status will change to closed • Closed-No further bookings can be made • Unclaimed-If an account is untouched for a given period of time, the balance is considered unclaimed. Account is closed and the balance is moved to an internal account. Funds can be reclaimed by the customer • Settled pension	KTSTKD • Active-0 • Settled.-1 • Closed-2 • Unclaimed-3 • Settled pension-4	Yes
Currency	Currency code for the currency the account funds are held in e. g DKK, GBP, SEK	VAKD	Yes

Account Mandate:
Account mandates API are used to add or remove mandates on a specific account to change the expiration date of the mandate. The input used to fetch this data is the account id (anonymous account id)

Account mandates are used for granting a third party (grantee) access to operate or view an account. One customer (access grantee) can have only one type of mandate for each account.

Method	End point Path	Description	Request Body	Response Body
POST	/accounts/ [anonymous Account Id/mandates/ [anonymous Customer Key]]	Add a new mandate to grant 3rd Party access to view or operate the account	[“type’: ”A-TWO JOINTLY”, ”valid From Date”: ”YYYYMMDD”, >Optional “valid To Date”: ”YYYYMMDD”→Optional	[“anonymous Customer Key”: ”XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX”, “type”: “A-TWO JOINTLY”, “valid From Date”: ”20210712”, “valid To Date”: ”20211231”
PUT	/accounts/ [anonymous Account Id]/ mandates/ [anonymous Customer Key]	Change expiration date of an existing account mandate identified y the customer ID	[valid To Date”: ”YYYYMMDD”]	[“anonymous Customer Key”: ”XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX”, “type”: ”A-TWO JOINTLY”, “valid From Date”: ”20210712”, ”valid To Date”: ”20221231”
DELETE	/accounts/ [anonymous Account Id]/mandates/ [anonymous Customer Key]	Remove an existing mandate to revoke the granted access	[account Id], [anonymous Customer Key]	[“anonymous Customer Key”: ”XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX”, “type”: ”A-TWO JOINTLY”, “valid From Date”: ”20210712”, ”valid To Date”: ”20221231”
GET	/accounts/ [anonymous Account Id]/mandates	Get mandates available on an account	[account Id]	[“anonymous Customer Key”: ”XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX”, “type”: ”A-TWO JOINTLY”, “valid From Date”: ”20210712”, ”valid To Date”: ”20221231”
GET	/customers/ [anonymous Customer Key]/account-mandates	Get all the mandates where the customer is a mandate holder	[anonymous Customer Key]	[“account internal Id”: XXXXXXXXXXX, “type”: ”SEPARATE”, ”valid From Date”: ”01-01-2019”, “valid To Date”: ”31-12-2019”]
POST	/accounts/ [anonymous Account Id/mandates/ [anonymous Customer Key]/validate-input	Validating input of add a new mandate to grant a 3rd Party access to view or operate the account	[“type’: ”A-TWO JOINTLY”, ”valid From Date”: ”YYYYMMDD”, >Optional “valid to Date”: ”YYYYMMDD”→Optional	
PUT	/accounts/ [anonymous Account Id/mandates/ [anonymous Customer Key]/validate-input	Validating input of change epiration date of an existing account mandate identified by the customer Id (anonymous Customer Key)		
DELETE	/accounts/ [anonymous Account Id/mandates/ [anonymous Customer Key]/validate-input	Validating input to remove an existing mandate to revoke the granted access	[account Id], [anonymous Customer Key]	

Input parameters

Parameter Name	In	Parameter Description	Required (Yes/No)	Datatype
Authorization	Header	JWT Token	Yes	JSON
Anonymous Customer Key	Request Param	Anonymized customer Id because of GDPR	Yes	String (UUID)
Account Id	Request Param	Anonymized account Id	Yes	String (UUID)
Type	Body	Type of mandate	Yes	Sting -Separate -Separate-Not Self -Channel Debiting -Channel Inquiry A-Two Jointly Two Jointly –Not Self -Several Persons-Jointly -All Owners Together -Individual Agreement B-Two Jointly C-Two Jointly Separate E-Bus. Self A-Two Jointly
Valid From Date	Body	The date from where the mandate is valid	No	String
Valid To Date	Body	The date when the mandate expires	No	String

Output parameters

Parameter name	In	Parameter description	Required (Yes/No)	Data type
Anonymous customer key	Body	Anonymized customer id because of GDPR	Yes	String (UUID)
Application Error Message	Error response			String
hhtp Status Code	Error response			String
Type	Body	Type of mandate	Yes	String
Valid From Date	Body	The date from where the mandate is valid	Yes	String
Valid To Date	Body	The date when the mandate expires	Yes	String

Account Ownership

The details provided by the API are the number of owners associated to that account, designated owner which is the primary owner of an account. Anonymous Customer key for each owner and other information like capital and interest percentage. The list will not contain account details. Filters can be used to isolate ownership data for single customer in the response

Attribute/Parameter	Description	Mapping to mainframe attribute
Number of Owners	The number of owners for one account	n/a
Association Owned Account	Yes/No	MKFLEJ
Valid From	Ownership valid from	GAEFRDT
Designated Owner	Primary owner of an account	PRIMEJER
Anonymous Customer Key	Customer Id	n/a (conversion of KNID or KundelId is customer id
Capital Percentage	e. g 50, 00	Kappc
Interest Percentage	e. g 100, 00	renpc

Request

Operation	Method	Endpoint Path	Request payload	Response payload
Get information on the owner of an account	GET	/ [account ID]/ownership		[“ownership”: [“number Of Owners”: 2, ”association Owned Account”: ”No”, ”validFrom”: ”2008-03-21”, “designated Owner”: ”xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx”, “owners”: [“anonymous Customer Numbe”: ”XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX”, “capital Percentage”: 50.00, “interest Percentage”: 50.00]

Error codes

Successful Read	200
Unauthorized	401
Not found	404
Internal Server Error	500
Bad Request	400

In Digital Core, we communicate changes over Account Events published using the bank’s enterprise publish/subscribe solution

Events are published to specific topics:

Consumers can create queues that subscribe to topics and receive all messages published there. All account event payloads are based on existing Account APIs.

Publish/subscribe enterprise platform (also known as PubSub) is an implementation of publish-subscribe pattern. Implementation is built on top of the messaging system IBM MQ

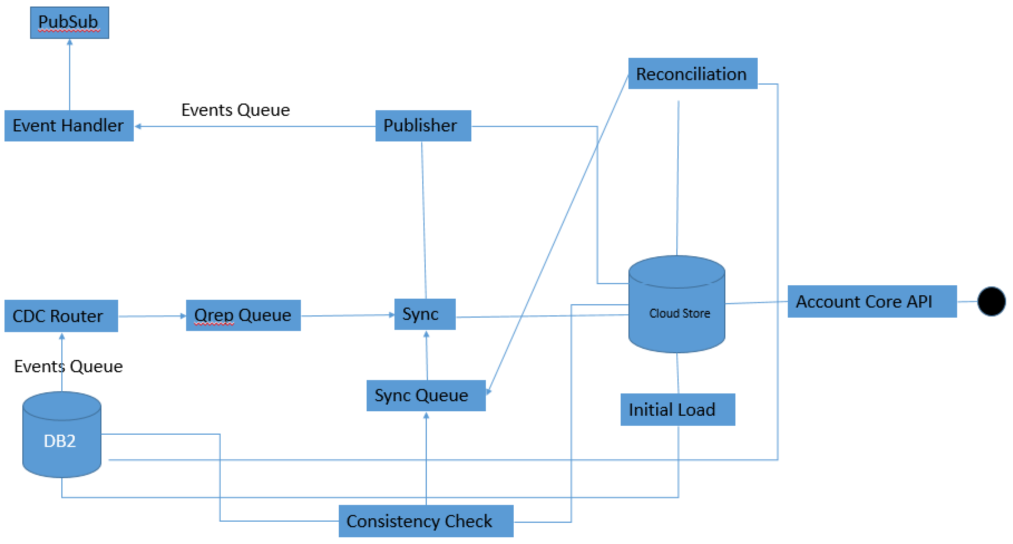
This platform enables developers to create event driven systems. Key benefits are the following

- Space decoupling-Interacting parties do not need to know each other
- Time decoupling – Publishers ad subscribers do not need to be up at the same time
- Synchrnization decoupling – Sending and receiving events does not block participants

The **PubSub** solution enables mainly 2 usecases scenarios:

1) Master Data Management – Enables consumers to keep locally persisted entities consistent (local replica of data)
2) Other domains to be notified and react according to changes in the domain (triggering automated processes/notifications and similar)

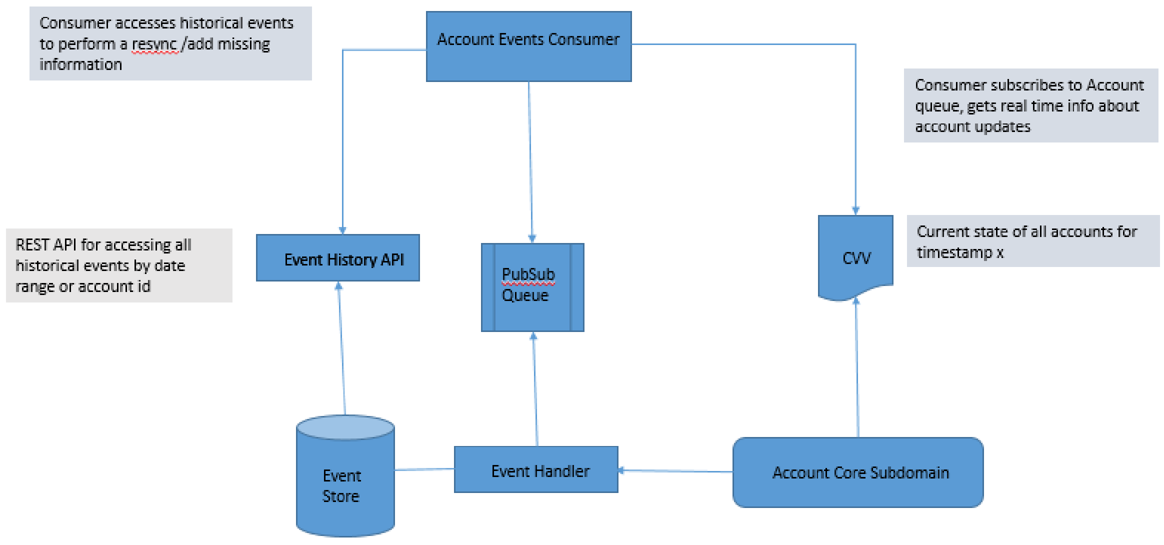
API Event Architecture



Logical Components

CDC Router	Component responsible for: • Listening to DB2 QREP changes for specific tables • Inserting Account related Qrep changes to the Qrep Queue Component is required because some tables will be shared among multiple subdomains
Cloud Store	A database used for: • Storing current account information for the KT Account Core API • Storing temporary "account changes" information for the Publisher component
Consistency Check	A process for checking periodically if the data between DB2 and Cloud store is consistent
DB2	DBLAN database – contains master data for Account Core
Event Handler	Component responsible for • Persisting events in Event store • Inserting events in pubsub queue
Event History API	API for accessing historical events from Event store ("replay" functionality)
Event Queue	Queue used by different Account subdomains for sending events to the PubSub queue
Event store	A database used for storing all historical events send by subdomains
KT Account Core API	A REST api for accessing account information (its V2 – on cloud)
Publisher	Component responsible for publishing Account related events to the Events queue. Evens are constructed based on Cloud store content
PubSub Queue	A Bank topic based Queue that can be subscribed by other systems for receiving updates on Account related information (e. g Account Stats change, etc)
Qrep Queue	A queue containing DB2 messages about changes in Account related tables
Reconciliation Service	An API that allows to initiate manual resynchronization of data for: • Given account ids • Given date range
Sync	Component responsible for: • Propagating account related changes from DB2 to cloud store • Preparing events for the Publisher
Sync Queue	A queue for handling resynchronization grpc calls that are in the CUSY reference architecture and allows better scalability

Event Consumer Journey


Consumer accesses historical events to perform a resync /add missing information
Account Events Consumer
Consumer subscribes to Account queue, gets real time info about account updates
REST API for accessing all historical events by date range or account id
Event History API
PubSub Queue
CVV
Current state of all accounts for timestamp x
Event Store
Event Handler
Account Core Subdomain