WEBINAR

Adoption of BIAN APIs in Day-to-Day development by USA Based Bank, PNC

July 28 | Q&A

Question	Answer
[Slide 14] Question/comment: you make a difference between "Inner" and "Outer" APIs, which sounds/looks strange: (1) what will be the difference? (2) Will it not be an unnecessary network hop?	1) **Inner –** These are our System of Record APIs. They expose the data from our source systems and are BIAN Inspired. This layer is meant to encourage reuse across channels **Outer-** These are the orchestration and filtering layer. These APIs integrate with the Inner APIs to complete an experience. Outer APIs are tightly coupled with an experience. Said differently, these are the API of the experience. 2) This logical separation of inner and outer APIs is an extra hop, but not an unnecessary one. If the logic for both exposing a System of Record and filtering the data for an experience were combined into 1 API, this would greatly reduce the opportunities for reuse. The reason for this is that most experiences have different sets of requirements for data they want to display, so those integrations so System of Records would need to be reimplemented for each of these experiences. This also helps keep channel display logic out of the inner API and System of Record.
Do any of your Outer APIs use other Outer APIs, or do they all only use Inner APIs ?	Other than a few exceptions, it is not a common pattern that we see. The current example that we see a pattern like this is in data aggregation for APIs on our Open Banking Platform. Since there are many integrations to pull data from, there is an outer to pull together the data, and another Outer to handle filtering of the data.
Is the outer API also developed using BIAN standards?	No – outer APIs are driven by the experience. The experience will dictate what data needs to be returned and in what formats. We aim to have the outer API be as simple as possible and usually has a Swagger that closely mimics the way the data is presented in the user interface.
Did you need to customize any of the inner APIs? As in did you need to add/remove fields which were not on the BIAN API.	Yes, this is something that we commonly see as we leverage BIAN for building our APIs. In these situations, we are only implementing fields that the source system actually returns, regardless of the BIAN model. If our system of record returns a field that BIAN has not defined, we

	leverage our API Model and Data Dictionary to choose the correct entity name.
First name, last name is US specific (the global approach is family name, given name) - is this only for the US?	The data dictionary has been developed with the terms that our internal teams are most familiar/comfortable with. The model could definitely be adapted to support a more global approach in the future if PNC expanded into other markets.
This also sounds very much like the historic service domain model of IBM which is completely based on SOA architecture. So, are you building Microservices? or are the "inner API" apps monoliths?	The Inner APIs are built as micro services. Each of the microservices has a single responsibility for the interaction with the system of record without doing any additional business logic at the inner api layer. We also do not leverage other SOA constructs such as Enterprise Service Bus technology. The architecture we presented where the experience communicates with the outer API who in turn communicates to the Inner API and lastly the Inner API to the system of record --- there are no components in between those hops. Compared to legacy SOA implementations, this is significantly simplified implementation. Our Inner APIs are based on BIAN SD and System Of Record, so we are tying the CRUD Operations together based on SD & SOR – {BIAN SD} – {SOR}, giving an opportunity for further segregate the endpoints based on Behavior Qualifier- {BIAN SD} – {SOR} – {BQ}.
Do you host your APIs in cloud or on-prem? If your current mode is on-prem, what actions are required for proper cloud migration(API definitions, pipelines, etc.)	We host our APIs on-prem in a cloud environment. All our tools are built in a very dynamic model meaning we can plug and play with different tools rather easy. To move to a pure cloud environment, we would not need to alter most of our tooling at all and the effort would be rather small.
Who creates the API's in the organization...the API COE, or are they done in the experience team?	The API CoE assists in creating the definition and structure of the APIs. The Delivery or experience teams are responsible for actually coding, deploying, and managing these APIs.
Slide 16 You always say Swagger, don't you use OAS 3.0 ?	Yes – Open API Spec 3.0 is our standard.
How to handle the naming of SDs which are implemented at Enterprise level and also at LoB level?	Our API CoE team is specifically focused on building out APIs in the Retail LOB. Those APIs are hosted on Innerpedia where other LOBs can leverage the definitions for integrations to those source systems.
Is Innerpedia a proprietary tool or available is open-source?	Currently Innerpedia is a proprietary tool to PNC

What is your approach to create a business scenario using the existing service domains by integrating them? Have you developed any tools for that?	We leverage the service domain only as an inspiration to inform our API model in terms of which areas of the Bank will need to be API modelled. We do not leverage or model the business scenarios by way of BIAN. Instead, we focus on delivering speed and agility in the creation of new experiences by way of this API framework.
Can an orchestration update more than one SOR? If so, how is the transaction managed?	Yes, an orchestration can touch multiple systems of record. Regarding how those transactions are managed, it would depend on the requirements. Those calls can be done either synchronously or asynchronously
Does the contribution back to BIAN has been resulting in publishing of new endpoint designs?	Yes – we have seen new service domain definitions created within BIAN due to some gaps that have been identified during different System of Record integrations.
Was the innerpeadia and the rest of the API toolkit a pure in-house build or a customised 3rd party software? Are other BIAN members able to gain more detailed understanding of the toolkit?	All of the tools that were showcased during the demo are pure in-house builds and are not available externally at this time.
What level of customization of the BIAN models was necessary in the API adoption process? (Level 1, little customization – Level 10, a lot of customization) / 2) What items mainly needed to be customized?	I would say somewhere in the middle of the scale (4-6). The mapping of BIAN domains to our implementations require very little customization overall. The endpoint definitions have some customization to support our implementation of exposing systems of record. We leverage Service Domain, Control Record, Behavior Qualifiers, Action terms from BIAN. In order to make endpoints very specific to the use case/functionality we add custom sub-qualifiers at the end of the endpoint The greatest customization that happens is in regard to the request/response payloads. These are almost 100% custom with only a little BIAN inspiration.
The slide talked about 'From' and 'To', earlier account API dealt with ACH Fullfillment/Position keeping etc now being mapped to Account(Inner) API which directly deals with one SOR. How these different service domains(ACH 3esigned3t/position keeping) are being mapped inside the Account Inner API? Does that means each Inner API does	We focus on building an Inner API that sits on top of the legacy systems that implement the ACH payment rails. This is necessary since these systems have existed for many years and we must continue to leverage them. However, if we were to consider a greenfield implementation and build out new payment rails we would expect a pure microservices approach much like the “coreless bank” approach BIAN presented at SIBOS in 2019.

API catalogue (inner APIs) mostly applicable on PNC managed platform. Where do these outer APIs reside? Can we align outer APIs too to BIAN ?	Outer APIs need not be BIAN compliant because they are directly tied to the experience. While Inner APIs are BIAN compliant because they are domain driven, Outer APIs can be performing many functions, so BIAN compliance may not be advantageous. We catalogue the Inner APIs in Innerpedia and we catalogue the outer APIs in our experience hub.
How has Party Reference Data Management SD deployed. Is it at Enterprise level OR at LoB level OR at both	Enterprise level.
how a bank could have ur corebanking capabilities which is designed by BIAN?	All assets demonstrated in the webinar are proprietary to PNC.
An ecosystem is most strategic useful if one is a platform business player. Will this new value create more open competition for banks then?	We believe that in the future, Banks will take many platform "flavours". At present, we are focused on building a simple and easy to build interface and abstraction layer to our core business data.
To BIAN: How does the quantity of BIAN input from BNC compare to the quantity of BIAN input from other banks. Does most of your BIAN input/feedback come from PNC?	
Does BIAN have recommendations on the implementation details (Infrastructure) or do they only give the API layer?	
API catalog (inner APIs) mostly applicable on PNC managed platform. Where do these outer APIs reside? Can we align outer APIs too to BIAN ?	Outer APIs can feed many different experiences – both internal and external to the bank. APIs that are exposed to external Third Parties are made available through our Developer Portal through our Open Banking Platform. Outer APIs that feed internal experiences are made available through a full-stack hub that aims to depict the contracts between the experience, outer and inner APIs.
Technologies used for APIs – Only REST or Graphql/gRPC used	Technologies discussed during the presentation as it relates to BIAN are only REST at this time.
What is your approach to create a business scenario using the existing service domains by integrating them? Have you developed any tools for that?	Not at this time.
Is one microservice created per SD?	There are many scenarios for what makes up a single microservice, SD only act as a guidance.

BIAN
Banking Industry
Architecture Network

	The API CoE has published guidance for teams when determining what should constitute a microservice. The high-level guidance is that a microservice should be made up of a unique SD and System of Record combination. This can vary for different architecture patterns if this combination is not practical.
What standard/best practice did you use to define the rules for the Swagger Validator?	There were many points of inspiration for the rules engine that makes up the swagger validator tool. The first would be lessons learned through working without the tool for over a year. This helped us address pain points in the current implementation and how we could address these issues. The next point of inspiration would REST standards. These include standard status codes, how data should be passed, etc. Finally we looked at how we could future proof the swaggers for tools and technologies we wanted to leverage in the future. We want to use the swagger to automate as much testing as possible, so requiring fields like examples and descriptions aid in that area.
What tool do you use to manage your ER model?	We are still working to finalize the tool that we will use for ER modelling. For now, we are leveraging basic spreadsheets to start the modelling.
How long did it take to develop the API toolkit for the COE?	To stand up the API toolkit in an operable fashion (ie create an initial endpoint list, starting to create the model), it took roughly three months. However, it is an on-going effort to expand coverage of the BOM and refine the API model.
The API CoE Hub is a nice piece of work. What platform(s) is that running on? Thanks!	The tech stack for the API CoE Hub is developed with an Angular Front end with Java Spring boot backend.
The compliance dashboard specifically references microservices. Is the compliance score for APIs, microservices, or both?	The compliance dashboard has multiple functionalities. It gives an overall score that takes many things into account (documentation, coding standards, architecture standards, configurations, etc.). In addition, it also brings scores from other tools (e.g. SonarQube) to give teams a holistic view of their microservice and how it is performing over time.
What is the average timeline of this re-publishing to reach the bank completing the loop?	Each swagger is published to innerpedia as soon as that code is available in its respective environment. In terms of getting consumers to upgrade to new versions of an API after being republished – we usually ask teams to abide by

	a 6 month deprecation period to allow consumers time to upgrade.
If you would use the Serverless model of implementation where you define your APIs in a declarative manner how would this change your approach?	A declarative manner of API definition where you define the APIs through code annotations would change our full approach would need to change, but the tooling would remain as is. The contract first approach we take is done to ensure teams have thought out their APIs and have a clean starting point. We also are able to leverage many forms of automation via the swagger such as virtualized endpoints, code and test generation, and pre-validation. In a severless model we would need to rework our tools to look at the API definition that is generated by the declarations within the code and then produce a pass/fail based off the output.
a. How PNC is leveraging BIAN API Catalog for API Development. b. For Entity / Attribute are we following BIAN API Models as is?	a. We have leveraged the BIAN catalogue rather heavily in building out our API toolkit. As discussed, the API toolkit is meant to create a comprehensive list of APIs that PNC needs to build to expose our Systems of Record as API endpoints. We used the BIAN catalog to build this all out. b. We are not following the BIAN models "as is". Everything that we are building is "BIAN Inspired". We are using BIAN to assist in mapping and creating the structure of our APIs, but have additional extensions and standards that are PNC specific.
If the teams use your swagger generator as a starting point, then why do need a validator? Are they able to customize the swagger?	Ideally, teams will always use the swagger generator to create their swaggers. However, sometimes teams will manually create their swaggers and benefit from having a validation tool available. Also, it could be the case that a team uses the swagger generator to create their swagger, but a new validation rule is created after their swagger has been created. From a pipeline perspective, the API CoE calls out to the swagger validator as a step within the Jenkins builds to ensure all swaggers are properly created. Teams can customize the swagger as long as they are following the rules set forth from the CoE. As an example, teams can customize field types, examples, and extend anything else

	about the API they wish. They cannot remove required fields like description, examples, etc.
In the BIAN Service Landscape I could see the APIs. How to derive the ER model from this?	We are building ER Model based on the current services that we have in PNC and leveraging/making use of BIAN BOM Model.
Do you use control records and behaviour qualifiers concepts on the URIs of your inner APIs?	Yes, Control Records and Behaviour Qualifiers are mandatory to have in the BIAN endpoints. We do not create custom CR & BQs. If there is a need to create a one, we get in touch with BIAN for their inputs.
What is the difference between inner API vs Endpoint catalogue?	Inner APIs are our System of Record APIs. They expose the data from our source systems and are BIAN Inspired. This layer is meant to encourage reuse across channels. Our Inner APIs are inspired from the BIAN API Model and leverages the BIAN Object Model (BOM) to model its inputs and outputs. An inner API is designed to only perform one function, and abstracts information away from the System of Record. The endpoint catalogue is is an ongoing list of endpoints that can be defined leveraging BIAN standards. Prior to creating a swagger for a service, development teams will first reference the endpoint catalogue to ensure that the endpoints that they wish to create have not already been built. If they have not, then the development teams will ensure that the endpoints that they want can be built adhering to the standards that the API CoE have set. This step is meant to be a precursor of sorts prior to beginning development. Once the endpoints have been approved by the CoE, then the development teams can proceed in making their swagger. This is essentially a predevelopment cycle to be performed by teams to ensure that no duplicate endpoints are being created and that the endpoints that do get created are in line with our standards.
Answered in session	
[slide 13] What do you mean by "inner" ? Do you foresee a difference between APIs used by PNC applications and APIs used by 3rd parties in the context of an "Open Banking" ecosystem?	
How complete is the set of inner APIs? Does it now cover all of the retail banking and commercial banking businesses?	We have started with Retail and will gradually expand to other LoB’s

Are your toolkits self-developed?	Yes
Do you see BIAN adoption beyond Banking line of business? Investment Banking? Wealth Management? Asset Management?	We have started with Retail and will gradually expand to other LoB's
Can you talk about challenges in implementing BIAN model at PNC. What was your major challenge in this entire journey of BIAN adoption? What challenges did you have in adoption & rollout?	Finding the right BIAN SD/BQ- PNC use cases for Inner APIs are very specific to what system of record is exposing, so the BIAN BQs may not be directly related. Endpoints have to be customized by adding sub-qualifiers beyond BQs to align them towards the API functionality. Rq/Rs Payload – BIAN do provide references to the payload data but does not provide the exact attribute level of information.