Abstract

A key aspect of digital transformation is building effective systems of engagement across social, mobile and web that can help businesses sell, serve and engage their customers anytime, anywhere and on any device. While end user app ecosystems are growing steadily with the emergence of innovative solutions, mainframes continue to be a relevant and vital part of business operations for more than 70% of Fortune 500 enterprises. With the bulk of systems of record applications running on mainframes, there is an ever increasing need to integrate mainframe data with social, mobile, web, and cloud-based SaaS applications. This point of view examines the methodologies, solutions and benefits of such initiatives along with potential tools and accelerators to streamline implementation. It also outlines the Infosys approach to developing innovative end user apps and solutions that engage customers and users.
Introduction

APIs are key components that connect the systems of engagement with the systems of record. Before getting into API enablement solutions in detail, let us take a quick look at system classifications, the need for integrating data between systems, and data access strategies. As the need for speed varies across system classifications, it is vital to establish a standardized, flexible and adaptable integration solution to bridge the speed gap.

System classifications and a speed layer view of the technology landscape

**Systems of record** are ERP-type systems used to run various aspects of the business like financials, manufacturing, CRM, and HR. These always contain the source-of-truth data or the master copy of the data.

**Systems of engagement** are systems that are used directly by end users and customers for ‘sticky uses’ like mobile apps and collaboration systems. Their purpose is to engage and serve the users anytime, anywhere.

Not all applications need to have the same speed in terms of time to market, i.e., the same cycle time to deploy changes or the number/frequency of changes. This depends on the nature of applications.

**Fig 1: Speed layer view to segregate different workloads based on their system classifications**

- **System of records**
  - Core business functions
  - Critical business functions
  - Fundamental and standard functions like accounts, posting, etc.
  - Mainframe

- **System of integration**
  - API management and orchestration of different platforms
  - Non-core and peripheral functions
  - Functions that deliver a competitive edge like payments, mortgages, etc.
  - API gateway, ESB
  - Microservices based architecture (x86 / Cloud)

- **System of differentiation**
  - Customer-facing channels
  - Analytics and ML
  - Innovative functions and solutions like mobile banking origination, offers for customers as well as AI-led customer channels like integration with Alexa
  - Modern UI (e.g. Angular) on x86/Cloud
  - Hadoop, NoSQL, MF
  - zAnalytics*
  - Innovative solutions on x86/Cloud

- **System of engagement**
  - Systems of engagement are experience focused applications used to engage large user bases like customers and employees. Hence, there is a need to update these continuously, particularly since user experience is the key to user/customer stickiness.

- **System of insight**
  - Systems of insight are analytics and ML systems used to engage end users and customers in providing insights and collaboration systems. Their purpose is to engage and serve the users anytime, anywhere.
While it is important to build state-of-the-art systems of engagement (as explained in Fig 1), such systems need access to the system of record data to provide meaningful functionalities to end users.

Providing data access to end user applications can be done in two ways:

1. **Data integration** – Consume data directly from mainframe data sources (DB2, VSAM and IMS) through web services

2. **Data offloading** – Create a copy of the mainframe data source outside the mainframe through Extract, Transform, Load (ETL) or Change Data Capture (CDC) processes

This paper presents two solution patterns to expose mainframe business services (code) and data to end user applications. While there are numerous ways to do this, the ones presented here are standardized and widely used. These solutions are based on data integration with mainframe, not data offloading. The use cases that require data offloading are already mentioned above. However, a detailed examination of these are beyond the scope of this paper.

**Data integration using APIs**

API enablement of mainframes is critical to seamlessly integrate mainframes with external applications. This is a standard modernization pattern. While there has been significant discussion on APIs and Representational State Transfer (REST) based interfaces over the last decade as well as the latest integration technologies are already replacing them, there is still considerable opportunity to expose mainframe-based services as APIs. Before delving into the technical solutions, let us take a quick look at the characteristics of REST-based APIs and the challenges of implementing these on mainframes.

**Fig 2: Use cases and benefits of data integration versus data offloadings**

**Fig 3: Characteristics and challenges of APIs**
Pattern 1: Expose legacy applications as REST APIs through z/OS Connect

Through this solution, existing z/OS business services or data from disparate legacy data sources can be exposed as APIs. These services may contain complex business logic and may run into thousands of lines of code. Thus, utilizing these services as-is saves time and money. The data from various legacy data sources like DB2, VSAM and IMS can also be exposed directly as APIs.

Pattern 2: Expose legacy data as REST APIs through Java Web Services and DB2 Native SQL PL Stored Procedures (NSP)

Through this solution, z/OS data can be fetched through native stored procedures and any additional business logic can be implemented on Java Web Services, which can in turn be consumed by end-user applications.

Technical overview

z/OS Connect is a product from IBM that provides a simple and secure way to discover and invoke applications and data on z/OS from mobile, cloud and web applications. Built on IBM WebSphere Application Server Liberty profile, the product runs on z/OS and uses standardized interfaces and data, including REST APIs and JSON. It also provides a tooling facility that helps create and modify REST APIs easily.

Currently, there is an increasing need for mobile, web and cloud-based applications to instantly integrate and consume business services from mainframe. This requires intuitive, standards-based and self-describing APIs that are exposed from the mainframe. Here, the underlying technology stack and data model is irrelevant as long as the functional and non-functional requirements are met. z/OS Connect provides this abstraction to business services and data that reside on mainframes.

The key steps involved in exposing an API are:

- Create a service definition to map and define the API to a z/OS asset. The service definition or SAR file contains a bind file and a JSON schema document. The bind file provides z/OS with knowledge of how the JSON maps to the target data structure (copybook) and a JSON schema, which are used to map the requests and responses
- Import the SAR file into the z/OS Connect API editor and create the API definition
- Configure the service provider in the server.xml file of the liberty profile. A service provider is what provides services to a specific backend resource like CICS or DB2. z/OS Connect can support multiple service providers for various z/OS backend systems
  - Deploy the bind files and JSON schema in z/fs and configure the service providers in server.xml
  - Deploy the API in z/OS Connect

Fig 4: API enablement using z/OS Connect

Z/OS Connect features

- Provides a common and consistent entry point to access Z assets
- Leverages specialty processors
- Provides a point of authorization for all back-end services
- Captures usage information by writing to SMF logs

API enablement using z/OS Connect: The mainframe platform is connected to mobile, web and cloud applications through APIs. The APIs are defined and managed using the z/OS Connect API editor, and the service provider is configured in the server.xml file of the liberty profile.
Pattern 2: Expose legacy data as REST APIs through Java Web Services and DB2 Native SQL PL Stored Procedures (NSP)

**Technical overview**

RESTful web services have become a de-facto standard for application integration. NSPs are a proven way to extract data through SQL while implementing simple processing logic like control flow, loops, validations, and transformations. These services deliver increased performance and scalability at reduced network traffic and cost. Today, many organizations have made it a standard practice to access DB2 data through stored procedures rather than direct JDBC calls as this method provides an additional layer of security in terms of what external apps can read or manipulate in the database.

DB2 native stored procedures are invoked by Java Web Service through JDBC calls and exposed as RESTful endpoints to external applications. In the above reference architecture (Fig 5), Java Web Service makes multiple parallel calls to the DB2 stored procedures, maps result sets to data objects (through OR mapping), applies business rules, and aggregates the results into a response object. The DB2 NSP is mainly used for data access with some basic validations and transformations that can be accommodated in the NSP queries. The web service may aggregate multiple parallel calls to DB2 NSPs into a logical response. It also handles security checks. Once the response object is created, it is returned to the user.

Java Web Service runs on a tomcat server deployed on a Docker container. It can scale horizontally based on volume. If the API is exposed for the public, then an API management layer may be needed to handle other non-functional requirements like security, logging and caching along with monitoring requests and managing APIs.

When stored procedures are exposed to external web services, performance and scalability become critical, particularly since the volume of requests can go up to hundreds of transactions per second. In such cases, the stored procedures should be tested for optimal performance based on some considerations like:

- Ensuring the access type is index-based, with only matching columns and without tablespace scans. This is critical when joining tables with millions of rows
- Determining the overall CPU time and cost required by using the execution time of the stored procedures together with the call volumes
  - Total CPU time per day = Execution (in milliseconds) for a transaction * number of transactions per day
- CPU and memory utilization can be used to determine NSP performance
  - Scale the Java layer horizontally by adding more active nodes
  - Conducting thorough integrated performance testing that simulates the actual production volumes (transactions per second) to identify performance issues and tune the applications accordingly

![Fig 5: API enablement using Java Web Services and DB2 NSPs](image-url)
Applications exist and are built to serve data in order to enable efficient execution of business functions and extract rich business insights. NSPs are an important component of this architecture. Thus, it helps to have a better understanding about stored procedures (SPs) and NSPs and their specific advantages.

In DB2, there are two types of stored procedures:

- External SPs run in z/OS WLM environment and are typically coded in COBOL/PL1
- Native SPs (from DB29) run in DBM1 address space and are coded in SQL-PL language

Native stored procedures can be used in cases where:

- Processing is data/SQL intensive and all data is available in the DB2 database
- There is minimal business logic to be coded in the NSP
- It is critical to maintain a low cost per transaction, i.e., where the NSP execution is zIIP (IBM® z Integrated Information Processor) off-loadable

Other alternatives like application programs (CICS/COBOL) should be considered in cases where:

- The processing logic is heavy with constructs like IF/WILE/CASE/REPEAT
- There is a need to invoke calls to external applications
- There is need to utilize specialized library functions
- The application requires access to files/tables other than the DB2 database

Beyond API enablement, there are other use cases for implementing NSPs. Cost savings can be realized by converting general processor workloads to zIIP eligible workloads. Some of these use cases are:

- Converting external SPs to NSPs as NSPs are faster and easier to migrate to DB2 LUW at a later stage
- Converting high-volume CICS transactions — those used primarily for data access (DB2 only) and have minimal business logic — to NSPs
- Identifying highly repeated data access patterns or any logic that involves complex computations/derivations and used in many batch and online programs. These data access patterns can be implemented as NSPs to reduce MIPS usage depending on volume

Key benefits of API enablement

- Faster implementation — Consume systems of record data with minimal coding for faster implementation of end user apps
- Accelerates SMAC adoption — Easily and quickly adopt any social, mobile, analytics and cloud (SMAC) technology
- Improves performance — Eliminates redundant code and componentizes business logic to achieve high reusability and better maintainability
- Builds future-ready applications — The highly scalable and agile architecture can address evolving needs
Enabling API across the enterprise can be daunting, particularly since large enterprises may run numerous applications on the mainframe across different business units. Thus, a planned and methodical approach to API enablement is vital to unlock maximum benefit.

Infosys has successfully executed several API enablement projects for various clients. We help companies discover, assess and prioritize a minimum viable product (MVP) and finally deploy APIs on their mainframes.

<table>
<thead>
<tr>
<th>Discover</th>
<th>Assess</th>
<th>Prioritize</th>
<th>Minimum viable product</th>
<th>Deploy</th>
</tr>
</thead>
<tbody>
<tr>
<td>• List all existing capabilities that are exposed</td>
<td>• Standardize and rationalize services discovered in the previous step (using a bottom-up approach)</td>
<td>• Prioritize APIs in the new catalog for migration</td>
<td>• Deploy capabilities for API management (API Connect) and REST-based services on mainframe (z/OS Connect)</td>
<td>• Migrate APIs based on priority, clone backend services and use wrapper to format messages</td>
</tr>
<tr>
<td>• Identify other capabilities that can be exposed</td>
<td>• Review industry standard APIs for the given domain (using a top-down approach)</td>
<td>• Prioritize any remediation required for backend services like modularization</td>
<td>• Remediate cloned backend services in a phased manner (not linked to when its exposed)</td>
<td>• Onboard consumers in a phased manner</td>
</tr>
<tr>
<td></td>
<td>• Build a consolidated full stack API catalog for the portfolio</td>
<td>• Assess capabilities that can be migrated to the cloud or to an on-premises distributed platform</td>
<td>• Clone a few backend services and expose these as REST-based services</td>
<td>• Retire the old services once all consumers are on-boarded</td>
</tr>
</tbody>
</table>

**Fig 6: Infosys approach to API enablement**
Leveraging Infosys and industry tools

Infosys has invested in tools and accelerators to accelerate and de-risk modernization projects. Two of these tools are described below:

1. Mainframe knowledge curation
The Infosys Knowledge Curation Platform (Ki) delivers an interactive environment for stakeholders to curate and enrich knowledge and form a single source of truth with summary and detailed views. The Ki portal can be used by modernization SMEs to gain insights into the application and understand which parts are used and which are not. These insights allow Infosys to work with the organization to develop a relevant mainframe migration strategy.

Infosys recommends a meet-in-the-middle approach to curate knowledge from the existing mainframe portfolio in order to reduce dependency on the customer mainframe SMEs. Fig 7 illustrates this approach and outlines semi-automated, mostly automated, baseline knowledge, and extracted knowledge inventory stages. For the semi-automated stage, Infosys has developed L1, L2 and L3 business process maps for different industries like financial services, healthcare and retail. These capabilities are used to map existing mainframe application components. The maps are reviewed and used to identify critical business process flows and functionalities to be migrated.

For the mostly automated stage, Infosys has developed a number of parsers and scripts that can be used along with partner tools from IBM and Micro Focus. These tools are used to analyze the operational workloads, interfaces, databases, components, and code. It will then curate the knowledge in a software readable format.
2. API enablement toolset

Fig 8 lists the challenges in a typical API enablement project. It also explains the various industry-standard and Infosys toolsets that can be used to address these challenges.

<table>
<thead>
<tr>
<th>Challenges</th>
<th>Infosys solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct mobile integration with z/OS</td>
<td>z/OS Connect</td>
</tr>
<tr>
<td>Simplification of integration patterns for REST</td>
<td></td>
</tr>
<tr>
<td>Rapid identification and definition of services</td>
<td>Infosys Ki</td>
</tr>
<tr>
<td>Extracting rules from spaghetti applications</td>
<td>Infosys Anti Pattern</td>
</tr>
<tr>
<td>Exposing services as microservices</td>
<td>Cornerstone</td>
</tr>
<tr>
<td>Enabling caching of services for static reads</td>
<td></td>
</tr>
<tr>
<td>API governance and gateway</td>
<td>Enterprise API gateway/governance tools (Apigee, API Connect, MuleSoft, etc.)</td>
</tr>
<tr>
<td>Enabling agility and faster time to market</td>
<td>IBM DevOps tools</td>
</tr>
<tr>
<td>Optimized workload management</td>
<td>Open source DevOps tools</td>
</tr>
</tbody>
</table>

*Fig 8: API enablement toolset*
Infosys understands that mainframe applications run the most critical business processes and transactions for many organizations, making them vital for supporting day-to-day business operations. The Infosys approach to mainframe modernization is unique because it addresses key challenges and risks associated with migration. It helps clients maximize value from mainframe modernization by:

- Leveraging a knowledge-based engineering approach to discover and analyze the existing mainframe footprint, accelerate as-is understanding and use this information to plan the scope of specific workload migration
- Analyzing and deconstructing the mainframe portfolio so workloads can be migrated with minimal impact to other dependent applications
- Focusing on people, process and technology to accelerate modernization and business outcomes
Exposing mainframe business services/data as RESTful APIs enables web, mobile, distributed, and cloud-based applications to consume such services/data anytime, anywhere and from any device, platform or technology stack. The power of data is key to enabling faster time to market and accelerating development of innovative end-user solutions/apps. While Z systems support many of the sophisticated business applications and data today, its value increases exponentially when these business applications and data can be exposed through an enterprise-wide catalogue of APIs.

Conclusion
Maran Gothandaraman is a Principal Consultant with the Infosys Mainframe Modernization Practice. He leads the mainframe modernization solutions program in the US. Maran has over 17 years of experience delivering complex IT engagements across industry verticals with a focus on mainframe modernization, application development and project/program management.