

Expose Existing z Systems Assets as APIs to extend your Customer Reach

Unlocking mainframe assets for mobile and cloud applications

Asit Dan

z Services API Management, Chief Architect

asit@us.ibm.com



#SHAREorg



SHARE is an independent volunteer-run information technology association that provides **education, professional networking and industry influence.**



Discussion points

- Business drivers/opportunities for leveraging z Assets as APIs
 - API Economy, Mobile, Quick Wins and Improving consumability and governance
- End-to-end Architecture and Roles
 - z/OS Connect and REST enablement
- Discovery of z based Services
- Collaboration across teams & crossing organizational boundaries
- Incremental adoption

Exposing z Assets as APIs

- *API Economy*
- *Unlocking z assets for mobile and cloud applications*
- *Consumability and governance*



#SHAREorg

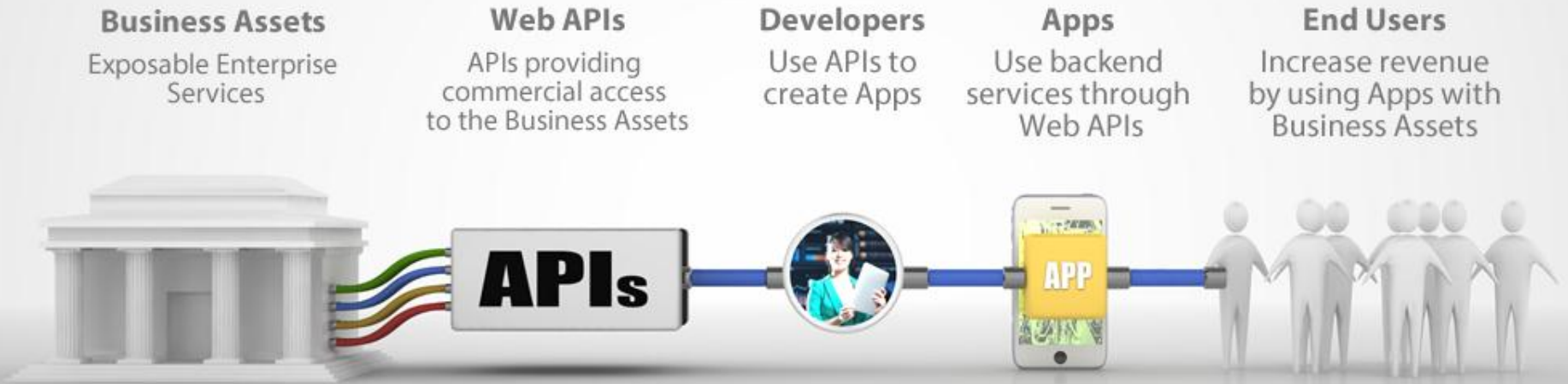


SHARE is an independent volunteer-run information technology association that provides **education, professional networking and industry influence.**



API Economy

Lifecycle



The API Economy

Where companies [providers] expose their (internal) digital business assets or services in the form of (Web) APIs to third parties [consumers] with the goal of unlocking additional business value through the creation of new assets

API Economy

Provider Perspective

Creating new opportunities by extending customer "reach" by exposing z based core business functions as APIs

Mobile App Assemblers
Developers & Partners

- IBM APIs**
 - Watson
 - Cloud Provisioning
 - Xtify
- Insurance APIs**
 - Life
 - Home
 - Auto
 - Claims
- Bank APIs**
 - Mortgage
 - Online Payment
 - Loans
 - Account Query
- Auto Dealer APIs**
 - Price
 - Availability
 - Location
 - Configuration
- Map Provider APIs**
 - Address
 - Locator
 - Weather
 - Traffic

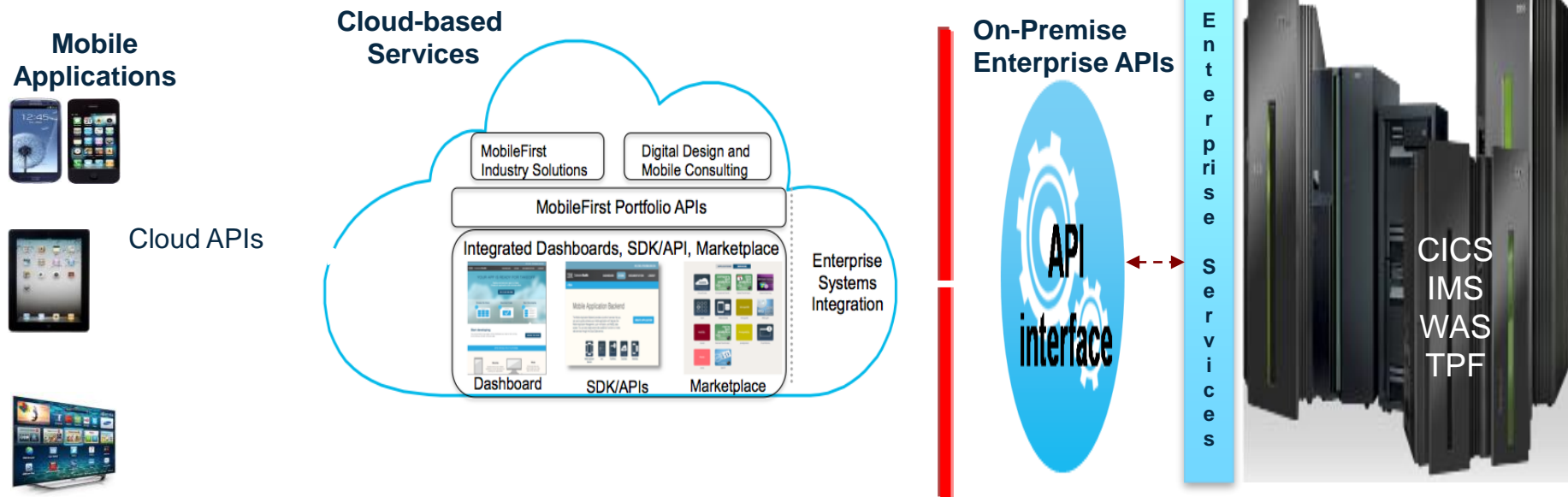


New Customers



Why API Management?

Business challenges addressed with APIM in exposing z based services/assets



Mobile, Cloud and Third-party Applications invoking z Services using APIs

1. Consumability of the APIs is Key:

- Easier creation and assembly of API from existing assets/services
- Visibility of APIs to internal and external developers
- Easier registration (by consuming applications) and set up including managing entitlement

2. Retaining business control (e.g., enforcing entitlement, accountability/chargeback) and gaining business insight in API invocation

- Securing APIs using a secure GW from unwanted external invocations (mapping to application level security) and enforcing workload entitlement
- Business Monitoring of API access in gaining business insight on the use of APIs by external applications, and for accountability/chargeback

Complete your session evaluations online at www.SHARE.org/Seattle-Eval

Making APIs consumable requires publishing not just what business functions they perform, but various additional details on their use, and making it easy for app developers to sign up

- Listing and categorizing APIs for easy to find
- Describing details on how to invoke an API



APIs	
REST basics	<
REST APIs	<
Release notes	<
Air	<
SOAP basics	<
SOAP APIs	<

REST APIs

Power your applications and infrastructure with Sabre's REST APIs. Now you can incorporate Sabre's intelligence services into your travel booking system.

Help your travelers decide

- When are the best dates to travel to my destination?
- Want to go somewhere but not sure where?
- Want to know where you can fly and when within your budget?
- Is this a good price relative to what others have purchased lately?

Name	Category	Function	API type	Docs
▶ Lead Price Calendar	✈ Air	🔍 Search	REST	➔
▶ InstaFlights Search	✈ Air	🔍 Search	REST	➔
▶ Destination Finder	✈ Air	🔍 Search	REST	➔
▶ Low Fare Forecast (beta)	✈ Air	📊 Intelligence	REST	➔

Destination Finder

The Destination Finder API retrieves a then current nonstop lead fare and an overall lead fare available to destinations from a specific origin on round-trip travel dates from the Sabre® cache. (A lead fare is the then current lowest published fare available via the Sabre cache for an origin, destination, and round-trip date combination.)

Calling this API

We designed several sets of request parameters for calling the Destination Finder API. You must pass a single origin airport in all cases:

- Provide a single set of round-trip travel dates (in `departuredate` and `returndate`). You can pass any single departure date from the current date to the current date + 192. You can pass any single return date that is equal to or later than the departure date, that does not exceed the maximum length of stay. (Length of stay is 0-16.)
- Provide a single length of stay (in `lengthofstay`) without any dates. The API retrieves lead fares for 30 consecutive dates that begin on the current date. To calculate return dates, the API adds `lengthofstay` to the departure date, where `lengthofstay` is 0-16.
- Provide a length of stay (of 0-16 days) with earliest and latest departure dates (in `earliestdeparturedate` and `latestdeparturedate`). For `earliestdeparturedate`, the date can begin with today's date. The `latestdeparturedate` can begin on today's date + 192, but the span of departure dates cannot exceed 30 days. The length of stay is from 0-16. The API retrieves lead fares for each departure date in your range, and calculates the return date by adding the `lengthofstay` to the departure date.

You can use these optional parameters in any request format, in any combination:

- `theme`. Filters the response for destinations that are based on a theme. A theme is similar to a travel category, and is based on geography or points of interest, such as beaches or national parks.
- `location`. Filters the response for destinations that are based on a country code.
- `minfare` and `maxfare`. Filters the response for lead fares based on these amounts. You can use these parameters together or separately.

An example public site listing its published APIs (similar to many others Twitter, Amazon, etc.)

Exposing Enterprise Services as APIs

SOA efforts have been driven by achieving developers' productivity gain and enabling reuse of functions, i.e., focused on development of services.

API Management, on the other hand, is driven by consumption of these services, i.e., improving consumability of these services for both external and internal developers of applications accessing these APIs, while also retaining control by the providers of these APIs

- Improving consumability includes listing APIs in a browsable/searchable catalog, and making it easy to register applications with the right entitlement level
- Retaining control includes not only enforcing entitlements and managing workloads, but also providing insight based on access history and accountability for chargeback.

APIs for z Assets Adoption Scenarios

1. APIM for Mobile

- Client is looking to start/expand a **mobile deployment**
 - Mobile app requires **access to backend services**
 - Client requires the ability to **monitor and manage** usage of backend services by new mobile applications

2. APIM for Cloud

- As for mobile, client is looking to develop new applications but in this case **in the cloud**.
 - As above, these applications require access to backend services and they need to monitor and manage usage of these existing services.
 - Additionally, the client need the ability to publish these services so the cloud developers (internal or external) can find and access services

3. APIM for Existing Services

- Client is seeking improved **consumability and governance** of existing services
 - Looking for the ability to **charge back** usage of existing services
 - May be working with **partners** to provide access to existing services
- Client is seeking **quick wins** spurring business innovation

End-to-end Architecture for Discovering and Accessing z Assets

- *Architecture and Roles for API creation and consumption*
- *Discovery protocol*



#SHAREorg



SHARE is an independent volunteer-run information technology association that provides **education, professional networking and industry influence.**



Roles in Three-tier Architecture for API Creation and Consumption

1. Development of enterprise services from existing z assets, making it easy to invoke these applications (*Bob*)

- z application environments (CICS, IMS, WAS) provides tools and runtimes to develop and invoke Web and REST based services from applications based on COBOL and PL/I
- IT role (Bob) is knowledgeable about this SW stack

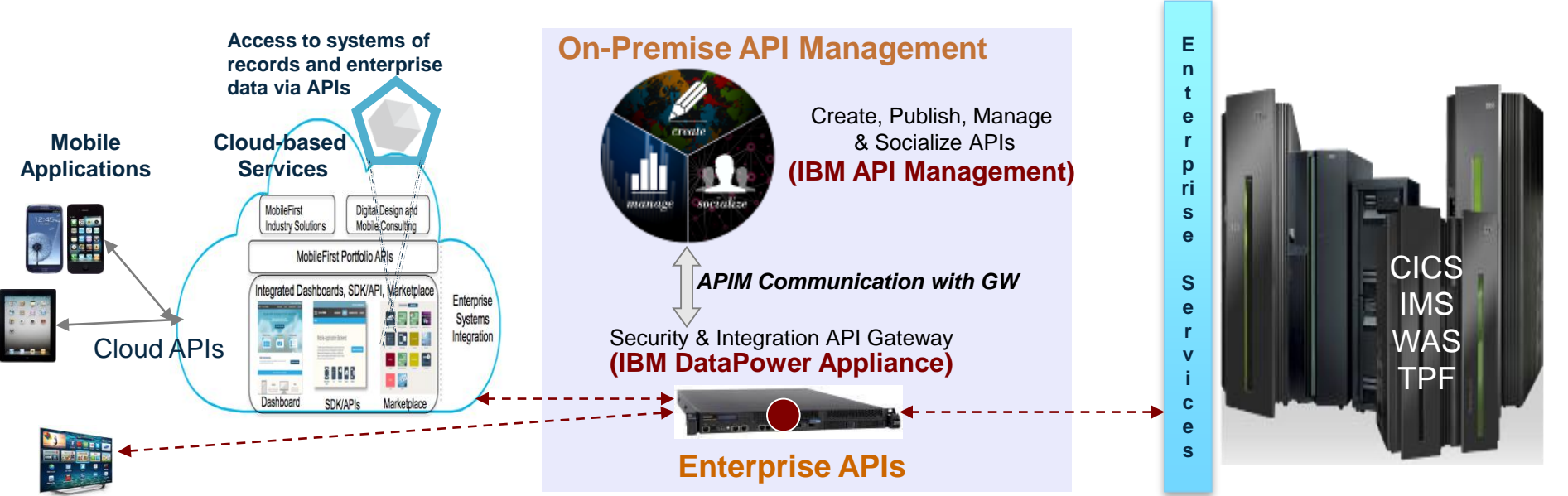
2. Development and management of APIs exposing existing enterprise services (*Shavon*)

- Addressing consumption and governance of APIs
- Create an API , discover a suitable existing service, and assemble this API from this service

3. Development of Mobile, Cloud or Third-party applications consuming APIs (*Jane*)

- Browse catalog and identify APIs to consume
- Register an application that will consume a specific API at a specified entitlement level
- Use of developer portal to test APIs

End-to-End Architecture for Mobile, Cloud and Third-party Applications accessing z Assets/Services using APIs



3. Mobile/Cloud App Enablement

2. Develop Enterprise APIs

1. Develop Services (Web or REST) → Enterprise Transaction Processing

Mobile/Cloud/Third-party Application Development

- Invokes APIs for accessing SOR

API Management

- Consumability by internal and external developers (creation and look up)
- Entitlement Management (securing, workload enforcement)
- Usage monitoring & Analytics

Service Enablement

- Enables invocation of z applications by remote applications using standard protocols (WSDL, REST)
- Converts SOAP or JSON into application specific (e.g., COBOL, PL/I) data and invokes applications

Jane - Mobile app developer
Uses APIs to access Back-End services

Shavon - API developer
Develops APIs from z based services

Bob - developer of z based Services
Develops services from CICS, IMS and other z applications

Complete your session evaluations online at www.SHARE.org/Seattle-Eval



Businesses are transforming themselves to participate in the API economy

How do you rapidly and securely expose your business to this developer ecosystem?



IBM API Management

Expose business services securely as APIs to developer communities, and analyze API usage

Provide self-service API portals to external/internal app developers

Manage & monitor the entire API platform



Complete your session evaluations online at www.SHARE.org/Seattle-Eval



Off-premise SaaS



On-premise private



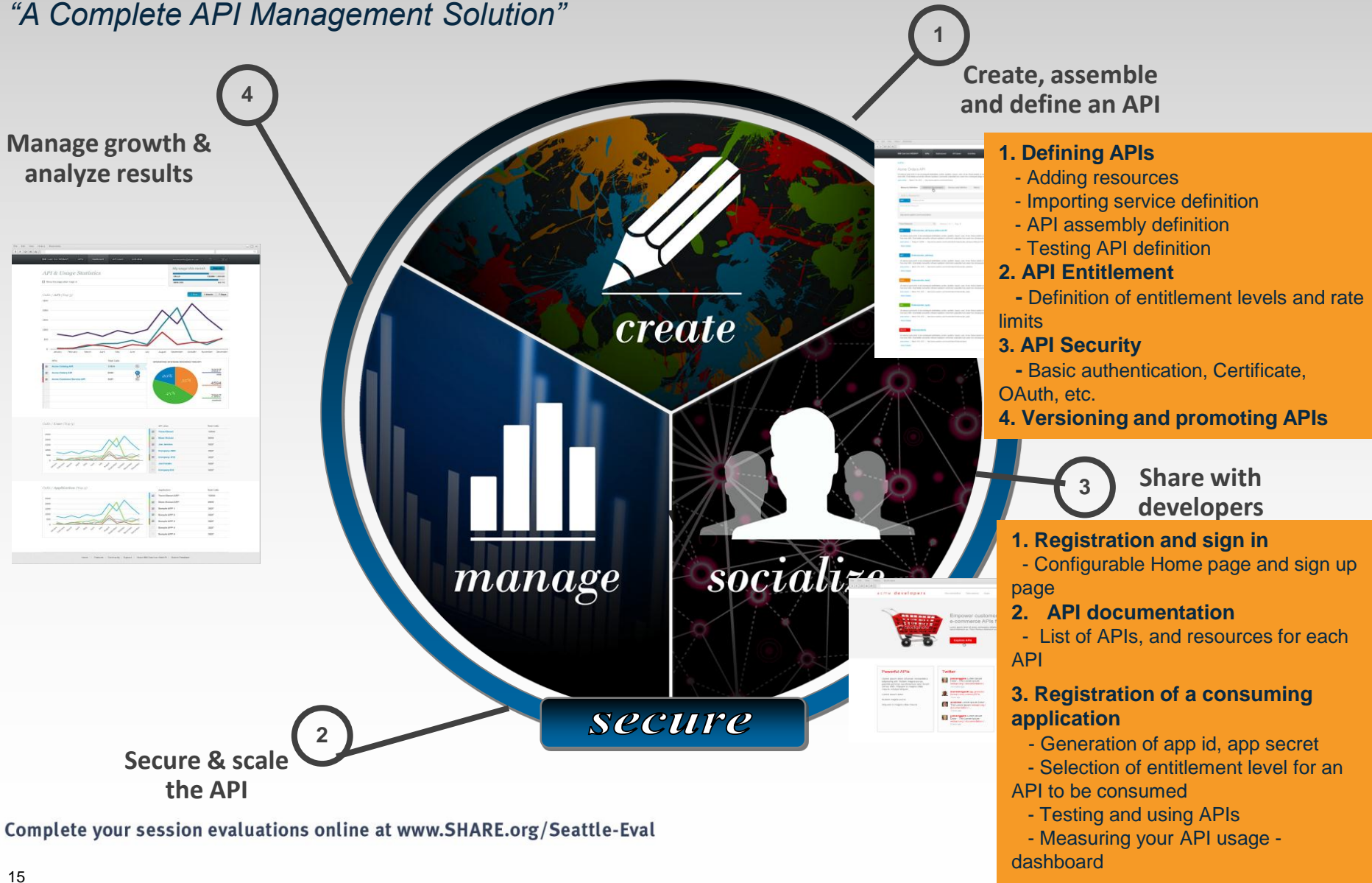
Off-premise private



Hybrid



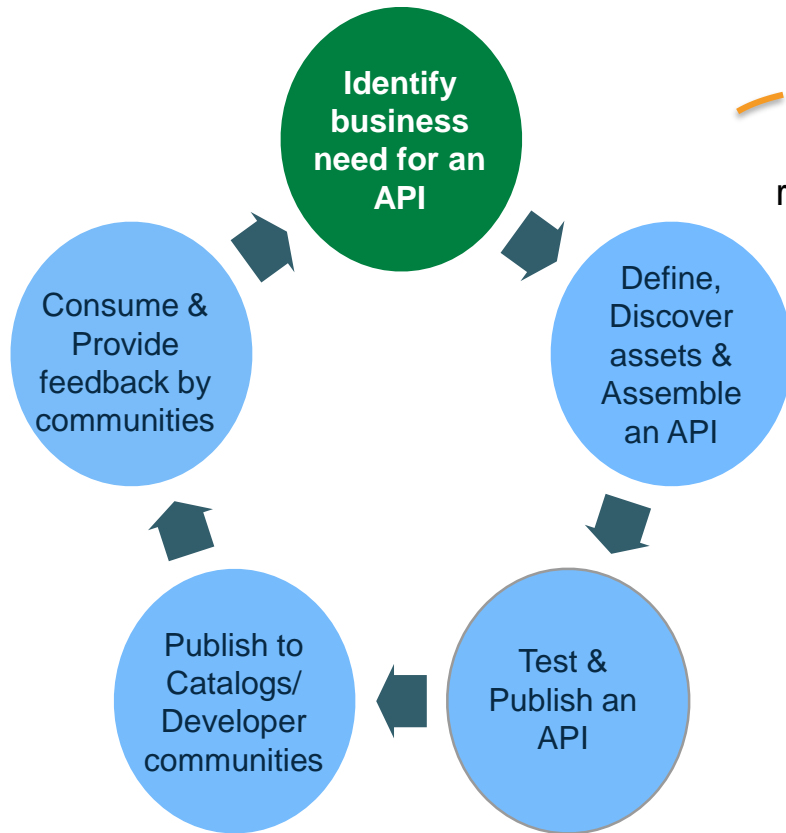
IBM API Management: “A Complete API Management Solution”



Complete your session evaluations online at www.SHARE.org/Seattle-Eval

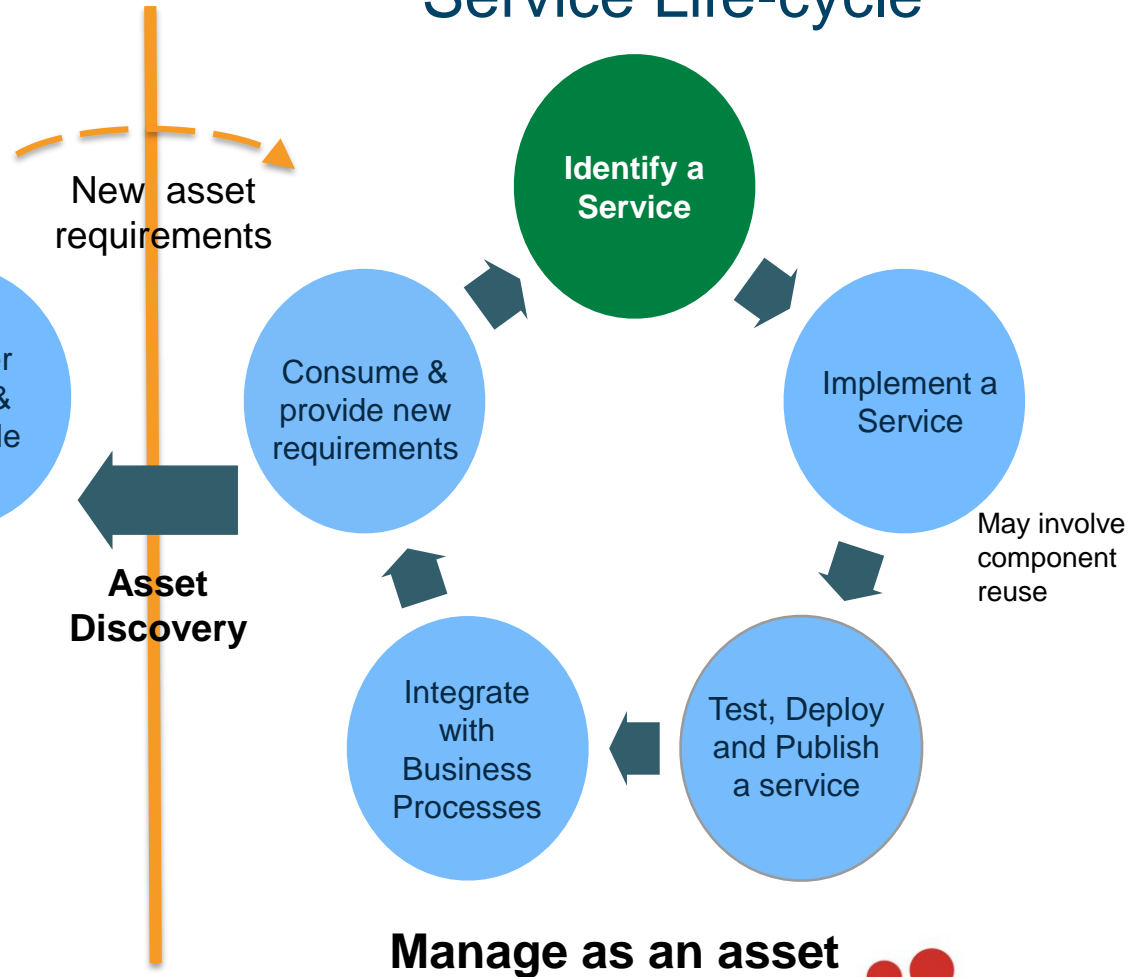
Life-cycle of APIs and Services

API Life-cycle



Manage like a product

Service Life-cycle



Manage as an asset

Roles and Development Tasks for enabling Mobile and Third-party Applications accessing Enterprise Assets

Bob

Creates **enterprises services** from existing *z* assets (CICS, IMS, WAS...)

- Uses an asset specific tool to generate service deployment artifacts (e.g., **bindfile, WSDL**) starting with an existing asset, and deploys the generated artifacts in an appropriate runtime environment (e.g., z/OS Connect, CICS TS/TG, IMS SOAP GW)
- Additionally, using an appropriate tool (e.g., CICS or IMS Explorer), explores asset details, and captures key **service metadata** for later understanding of its business function (such as, **description of business function, business classification of this function, association of keywords**, etc.)
- May also publish this service to an external registry (e.g., WSRR or other existing registries in customer environments)

Shavon

Creates an **API** from an existing service, as well as various entitlement policies

- **Discovers** deployed services from various back-end application environments in determining an appropriate service to expose
 - Queries existing services based on matching business functions (i.e., service metadata)
 - Navigates service list and views **service description, other metadata and interface definition** in understanding and selecting an appropriate service
- Defines an API starting with the selected service interface
 - Defines API name, description and resource
 - Defines an appropriate transformations in mapping an API resource to an existing service interface definition
 - Defines various entitlement policies
- May syndicate this API to multiple marketplaces (e.g., BlueMix)

Jane

Creates a **mobile app** invoking **APIs**

- Navigates and views APIs using the developer portal
- Develops app code invoking selected APIs

What is Service Discovery?

Identifying an existing service for performing a specific intended business function, and getting detailed definition of the service

- 1. Identifying:** Searching, browsing, understanding and eventually, selecting a service from a set of services
 - Querying to retrieve a list of matching services
 - Browsing information on retrieved services
 - Getting additional details as necessary in understanding a service
 - Selecting a service when a service is deemed a good match
- 2. Getting detailed service definition:** Retrieving various details of a selected service
 - Service schema and description (e.g., WSDL or JSON) for API definition
 - Getting additional technical details for API assembly including details on runtime invocation and security protocols

Why is Discovering a Matching Service Challenging?

Identifying a service or code for reuse

- Need to ensure that the code performs the intended functions for the consuming application
- A good service documentation needs to include
 - not just the technical details on the interface,
 - but also the semantics of the terms, constraints on use,
 - and description of business functions it performs
- Even for code reuse by the same development organization, needs to look back at the associated business requirements, unless key information is codified as associated business metadata, e.g., business classification of transactions, such as Payment, credit, etc.

Why is Discovering a Matching Service Challenging?

Identifying a service or code for reuse, i.e., ensuring that it performs the intended function is always challenging

- **Searching on technical information regarding a service, e.g., service interface definition (WSDL), is not sufficient, as it doesn't provide a lot of understanding on the business function a service implements**
 - Names of z services may be auto-generated from cryptic names of CICS and IMS transactions
 - However, names of services and parameters, and even any presence of certain parameters in the definition (e.g., "interest rate") can narrow the searches to likely reusable services (e.g., for "loan payment" business function)
 - Based on a prior knowledge, searching for a specific technical attribute can be useful in identifying a service
- **Detailed textual description of a service helps to distinguish business functions amongst a set of closely related services**
 - Even with SOA governance, various trade-offs in the reuse of code may result in multiple closely related service definitions
 - Evolution in supporting new requirements, ownership and impact on the deployed services, performance considerations, etc.
 - Business glossary can be helpful in understanding meaning of parameters
- **Business classification on services can help narrow the search to a small list of services**
 - Business may define a prior categorization of business functions (e.g., "Payment", "Loan application")
 - Annotating services with business tags can help identify services that may be relevant
- **Key word search in any aspects of service - definition, description and other metadata and even code - can help at the initial stage of discovery**
 - Customers already make use of such functions for reuse for code development

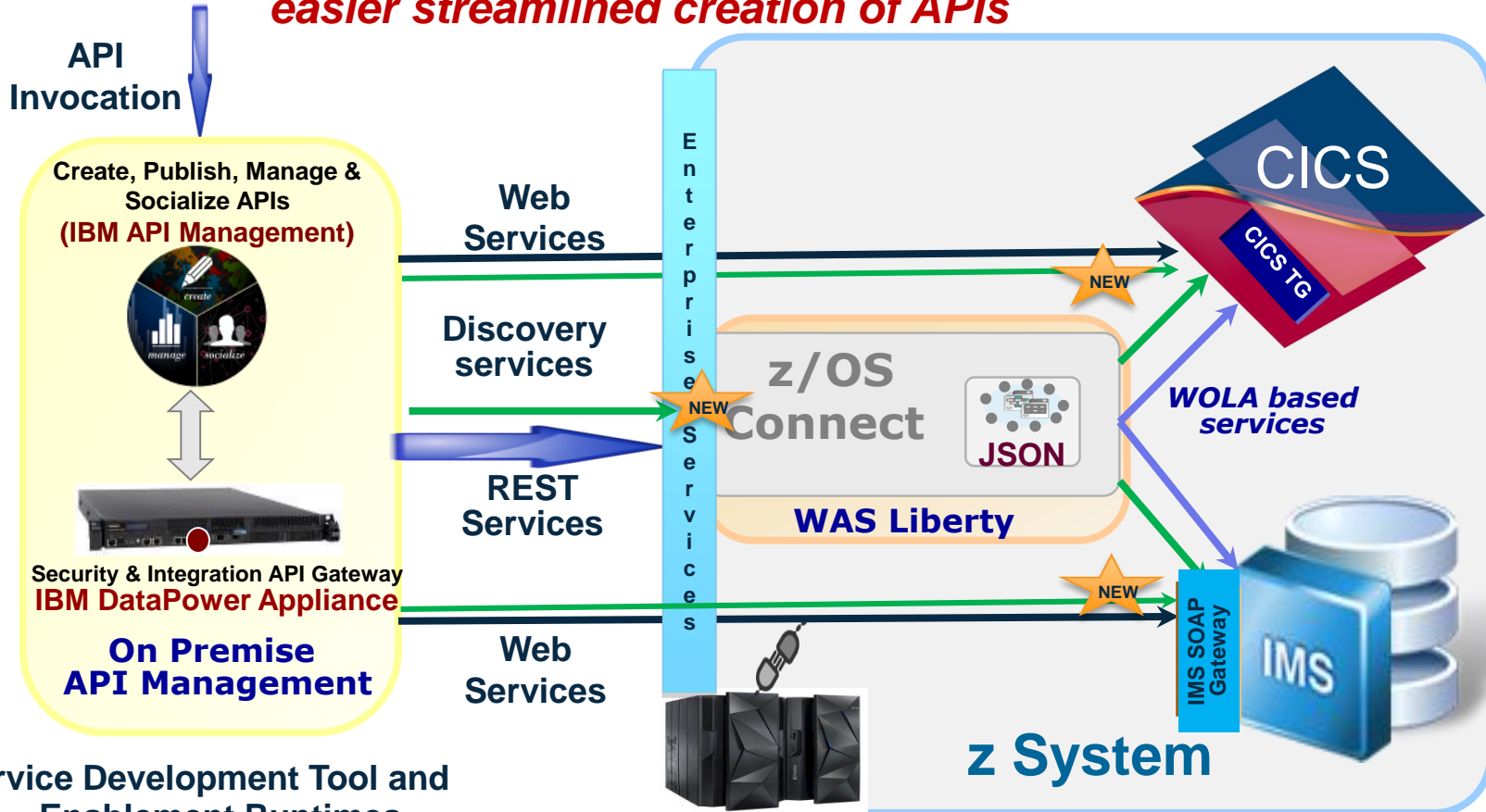
How to Search for a Matching Business Service?

Searching on various service attributes

- **Keyword search**
 - Search on keywords on any field on service definition, especially the textual description of service
 - Example: **serviceKeyword=** *interest rate or loan application*
- **Technical attributes:**
 - Search on well-known attributes of a service: *serviceName, operationName, messageFieldName, serviceSchemaType (WSDL or JSON)*, etc.
 - Example: **serviceName=***loanPayment* & **serviceSchemaType =** *WSDL*
- **Business service classification (not supported)**
 - Search on a list of queryable business tags
Example: **serviceClassification=***loanApplication or addressValidation or creditCheck*
 - Search and navigate across pre-defined service classification hierarchy: *z customers may support a pre-defined grouping of services (with **business service classification**), and a specific application environment may also provide the ability to navigate/query sub-groups*
 - Example: **ServiceClassificationGroup=***loanApplication*

Discovery and Invocation of z Systems based Services

New integrated capabilities in APIM and in z System for easier streamlined creation of APIs



Service Development Tool and Enablement Runtimes

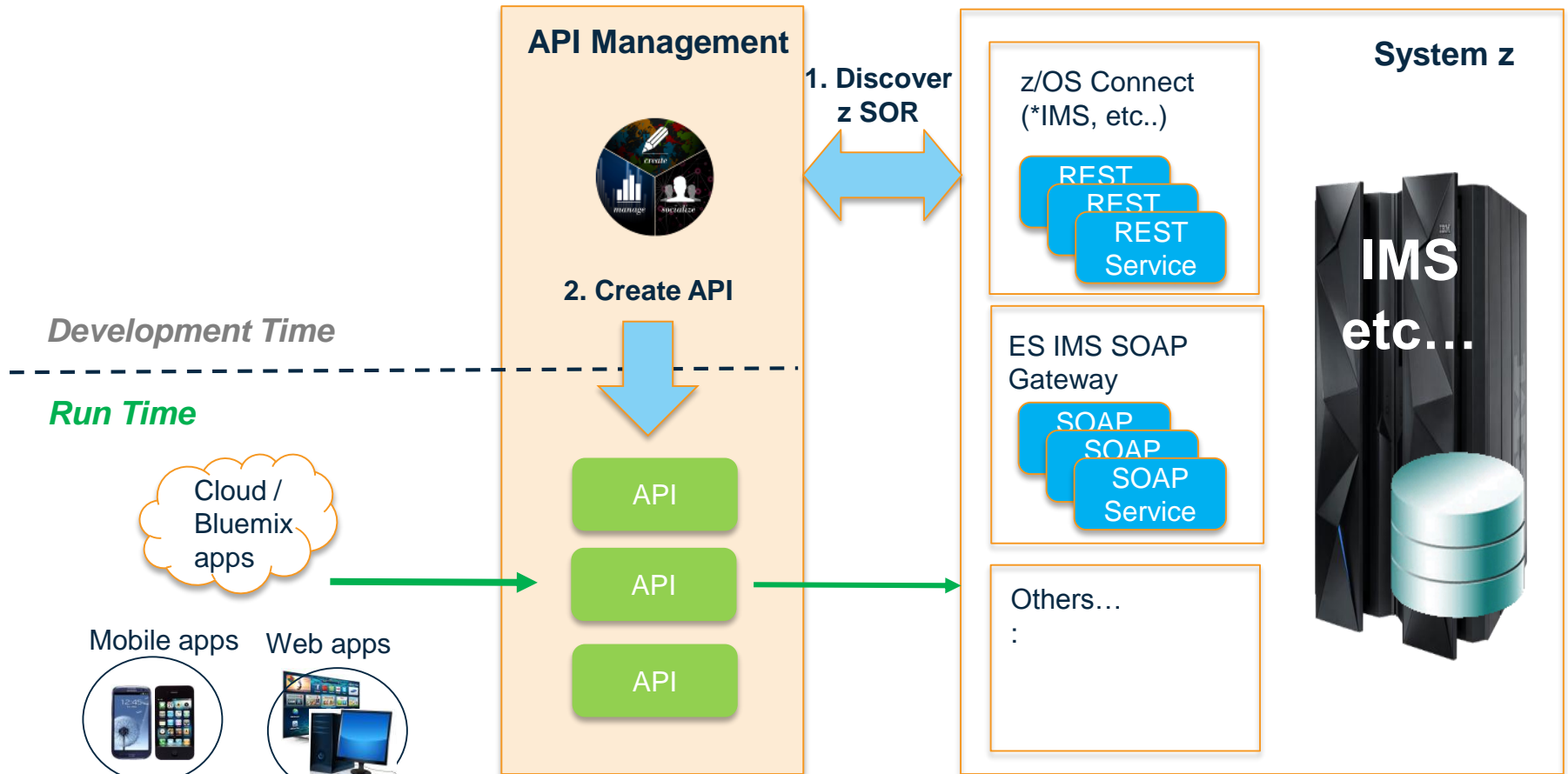
- ❑ **Web Services**
CICS and IMS provides separate tools and runtimes; TPF provides runtime libraries
- ❑ **REST/JSON**
CICS and IMS use common z/OS Connect runtime

Discovery of z Services for API Development

1. Get a list of deployed services (Service Identification)
 - Filter based on technical and business service attributes
2. Get schema for a specific service (API Definition)
3. Get additional deployment details for a service (API Assembly)
 - E.g., security protocol support, invocation uri

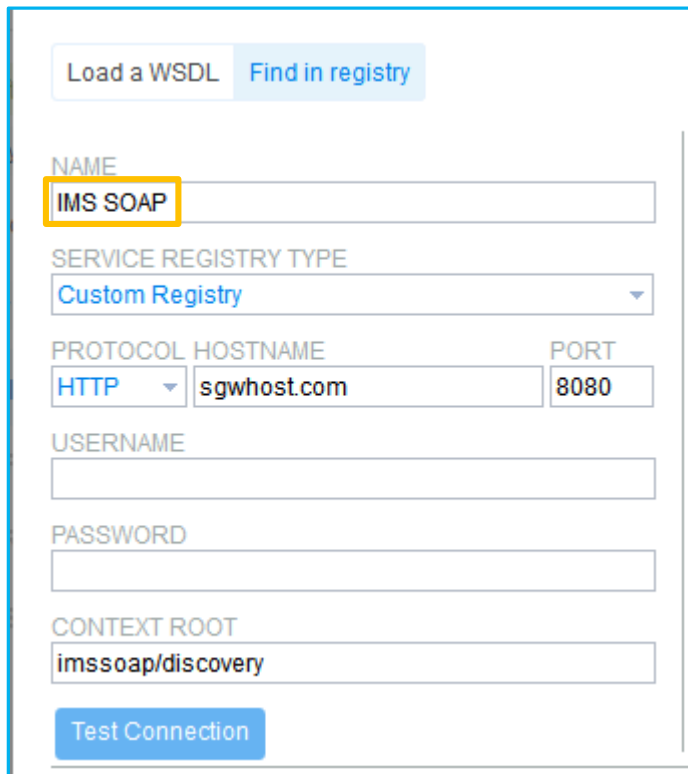
Complete your session evaluations online at www.SHARE.org/Seattle-Eval

APIM and z Systems service discovery



* = Intended support

1. Define Service Registry in APIM



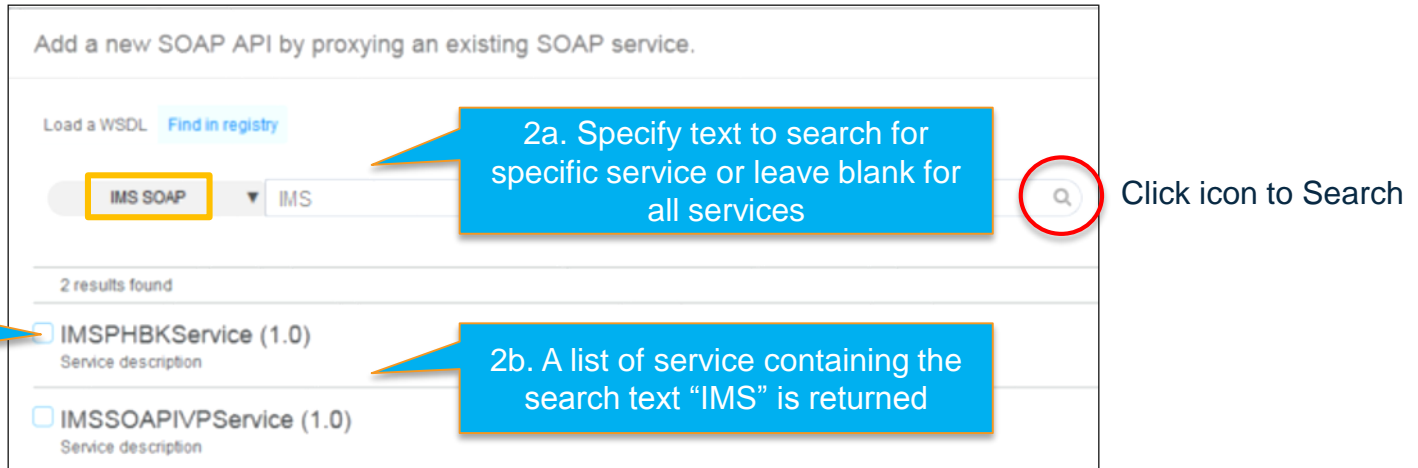
The screenshot shows a web form for defining a service registry. At the top, there are two buttons: "Load a WSDL" and "Find in registry". Below these are several input fields and a dropdown menu:

- NAME:** A text input field containing "IMS SOAP", which is highlighted with a yellow border.
- SERVICE REGISTRY TYPE:** A dropdown menu with "Custom Registry" selected.
- PROTOCOL:** A dropdown menu with "HTTP" selected.
- HOSTNAME:** A text input field containing "sgwhost.com".
- PORT:** A text input field containing "8080".
- USERNAME:** An empty text input field.
- PASSWORD:** An empty text input field.
- CONTEXT ROOT:** A text input field containing "imsssoap/discovery".

At the bottom of the form is a blue button labeled "Test Connection".

- Add SOAP Gateway as a custom registry in APIM
- Specify host, port and the context root of the SOAP Gateway service registry
- Test and save connection

2. Discover, search and add API for z System services



The screenshot shows a web interface for adding a new SOAP API. At the top, it says "Add a new SOAP API by proxying an existing SOAP service." Below this, there are two buttons: "Load a WSDL" and "Find in registry". A search input field contains "IMS SOAP" and "IMS". A magnifying glass icon is circled in red. Below the search field, it says "2 results found". Two services are listed: "IMSPHBKService (1.0)" and "IMSSOAPIVPSERVICE (1.0)".

2a. Specify text to search for specific service or leave blank for all services

2b. A list of service containing the search text "IMS" is returned

2c. Select the desired service to create API

Click icon to Search

- With the SOAP Gateway registry selected, discover services by clicking the search icon
- You can specify a text string to search for specific service. Or leave the search field blank to retrieve all services.
- Once the list of matching services returned, select check box for the desired service to create API.

3. API for z System service is created

APIs

Draft APIs
+ API
Find





Deployed to	Name ▲	Path	Last Modified	Actions
Sandbox	<p style="margin: 0;">ContactService (1 version)</p> <p style="margin: 0; font-size: 12px;">Operations related to Bank A contacts exposed by the Mainframe</p> <div style="display: flex; align-items: center; margin-top: 5px;"> REST + Tag </div>	/contact	8 days ago	★ 🗑️
Show	<p style="margin: 0;">IMSPHBKService (1 version)</p> <div style="display: flex; align-items: center; margin-top: 5px;"> SOAP + Tag </div>	/IMSPHBKService	20 days ago	★ 🗑️

- A new API is created for the IMS SOAP service. It is added to the list of APIs managed by APIM
- APIs can be published and make visible publicly for internal or external users

APIM and Bluemix Integration


- APIM integrates with Bluemix to enable Bluemix developer to discover APIs in APIM

Integration
Extend existing investments and infrastructure

 API Management IBM BETA	 Cloud Integration IBM	 Secure Gateway IBM	 Containers Experimental
-----------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------

- APIs in APIM can be published and accessible by Bluemix applications as Custom APIs

Custom APIs
APIs published in your org or shared from APIM

 GoodHealth v1 : Sand... Community

Collaboration across Teams and Incremental Adoption

- *Lessons learned through POC*
- *Incremental adoption*



#SHAREorg



SHARE is an independent volunteer-run information technology association that provides **education, professional networking and industry influence.**



Key Lessons Learned so far #1

Collaboration across key stakeholders, i.e., z Asset owners, z Architects and Enterprise Architects are essential in exposing z assets as APIs

- **Need buy-in from the asset owners to champion and agree to expose APIs**
 - Asset owners must see the new business opportunity and/or benefits of exposing as APIs in improving consumability and governance
 - Asset owners also must overcome their apprehensions in exposing business critical assets, and be assured of APIM capabilities in retaining control for access to these assets
 - Develop roadmap with an incremental approach for developing and consuming APIs (by Mobile, Cloud and other third-party or internal app developers)
- **Need collaboration across IT roles for developing z assets as APIs**
 - z Architects must participate in developing enterprise services from the existing applications
 - Enterprise Architects must participate in designing, assembling and managing APIs using APIM

Source: 3456A Leveraging z Systems Assets with API Management at **Humana**, InterConnect Conference 2015, Craig Whitaker, Asit Dan

Key Lessons Learned so far #2

Two ends of the opportunity spectrum – Quick win spurring innovation and improving consumability and governance of existing assets

- Quick wins spurring Innovation
 - Build APIs to expose newly developed simpler extensions to existing apps
 - Mostly for simpler look up or retrieving real-time data
 - Alternatively, incrementally creating new channels for updates by end-customers
 - Simple APIs, however, with huge potential
 - Simple REST based APIs are ideal
 - Consumption via mobile apps
- Improving consumability and governance of existing assets
 - Start with existing services (most likely Web services) if SOA approach is already in place
 - Alternatively, identify/refine the components to be exposed as services/APIs (using the service identification methodology)
 - Define a roadmap for incremental adoption of APIs
 - Co-exist with other application integration/invoke approaches until the newly defined APIs are fully adopted by all consumers

Source: 3456A Leveraging z Systems Assets with API Management at **Humana**, InterConnect Conference 2015, Craig Whitaker, Asit Dan

Key Lessons Learned so far #3

Considerations of both REST/JSON and Web Services are appropriate

- REST/JSON based services
 - For new APIs for simple data look ups or updates by end-consumers
 - For consumption by Mobile apps and/or exposing to third-party apps
 - Little constraints imposed by existing services, or adoption by existing consuming applications
 - When required, REST based APIs can be created easily from existing Web services using IBM APIM
 - Simpler datasets and/or validation schema requirements
- Web Services collaboration across IT roles for developing z assets as APIs
 - Continue with APIs based on existing Web Services when
 - Significant changes are needed to adopt any new REST based APIs
 - Complex dataset or complex Schema based validation is required
 - No new consuming apps that demands simple REST based APIs

Source: 3456A Leveraging z Systems Assets with API Management at **Humana**, InterConnect Conference 2015, Craig Whitaker, Asit Dan

Call to Action

✓ Leverage existing z assets by exposing as APIs

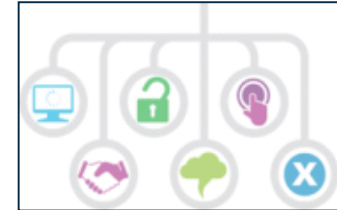
- Reach out to **z asset owners** in exploring scenarios around the three entry points
 1. Reach **new customers** and **markets** with new applications and solutions accessing core business functions, through business partners
 2. Improve experience of existing customers and/or deliver new services with **Mobile applications**
 3. Gain more **business control** and **insight** over access to the enterprise services, while improving consumability and simplifying access to z assets
- Team up with **enterprise architects and z architects** - both from the customer side and IBM – for designing an end-to-end solution architecture

✓ Work with IBM through POC

- Use IBM APIM and z product capabilities in developing API based solution
- Identify/define incremental business scenarios, and try out through POCs

API Management Resources

- Product Page
 - ibm.com/apimanagement
- API developer community
 - developer.ibm.com/api
- Follow us on Twitter
 - @ibmapimgt
- YouTube Channel
 - youtube.com/ibmapimanagement





MOBILE

Thank You

