Modernizing e-Government: Four Principles for Choosing the Ideal API Platform

State and local governments today strive to transform their operations to break down silos, improve interagency collaboration and communication, and delight their citizens with access to streamlined and automated services. Unfortunately, transforming government operations can be like trying to turn an aircraft carrier—a slow, laborious process. When it comes to technology, public agencies and institutions are often committed to information systems that have been running for decades. Such legacy systems present daunting challenges to the goals of automation and digital transformation.

As a result, citizens typically must access multiple systems or government agencies simply to file an application or submit a service request; private institutions face the same hurdles. Siloed legacy systems also make it challenging for separate government agencies to obtain necessary information from one another.

While governments have been trying to solve this silo problem for many years, the available solutions—including point-to-point integration techniques and data warehousing technologies—have come with problems of their own. Point-to-point connections can prove brittle, for example, and centralizing all data in a single location is a herculean task bound to create friction between organizational units.

Fortunately, the advent of Application Programming Interfaces (APIs), microservices, and the ability to flexibly aggregate and orchestrate them are finally making truly interoperable systems a reality. APIs connect different systems and make collaboration possible. Modern API platforms enable APIs to be flexibly accessed, orchestrated, monitored, throttled, secured, and so on, to meet the wide-ranging needs of government agencies and their constituents.

API platforms can eliminate the need for data to be stored in a central repository; government agencies can continue to operate and control their own systems and databases while sharing information as needed— with the level of access and privacy they desire. Since each agency remains in full control of its own processes, data, legacy applications, and APIs, this new approach aligns perfectly with each agency's organizational structure, governance, and culture.

InterSystems®
Creative data technology

This approach to digital transformation holds out the promise of seamless, user-friendly service both for citizens and for public servants working to enhance intergovernmental communication. But not all API platforms are created equal.

An ideal API platform must ensure:

1. Fast and flexible development
2. Reliability, scalability, and performance
3. Trust and security
4. Easy management and maintenance

Fast and Flexible Development

API development can be complex, particularly given the government's need to adhere to standards and regulatory policies. The ideal API platform provides capabilities and tooling to simplify and speed API development. These include:

- Connectors
- OpenAPI Specification (OAS) support
- Developer portal
- Data transformation

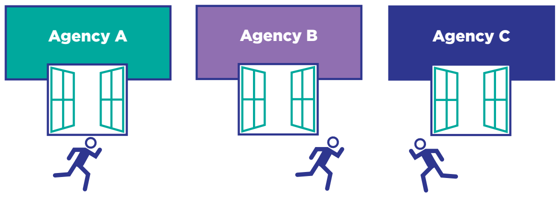
THE BOTTOM LINE:

A good API platform should provide "RESTful"[1] services as well as connectors for numerous legacy systems that do not support modern interoperability technologies. The most common connectors are: File/FTP, SOAP, Language Bindings for Java, .NET, Python and C, TCP/IP, HTTP/HTTPS, support for processing XML, Fixed Length, CSV, and JSON payloads.

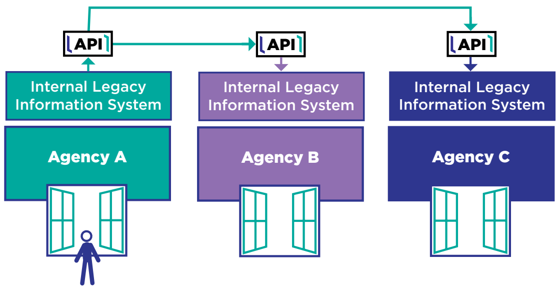
Connectors

The ideal API platform must help public agencies work together seamlessly—even when those agencies use different systems for organizing and storing information.

Consider the case of a citizen applying for a government program that is being run by Agency A. If the program requires information held by Agency B, that citizen typically has to gather the information from Agency B himself. If an application requires information from several agencies, a citizen can find herself buried in paperwork. The same difficulties arise for private institutions interacting with the government, and even for government agencies working with other agencies.



An API can be used to connect legacy systems that don't currently communicate. APIs expose data and functionality from internal systems in a way that is easy for others to use—without requiring any changes to the legacy systems. Essentially, they serve to connect systems that weren't designed to work together—a bit like the universal power adaptors people use on vacation abroad.


API
API
API
Internal Legacy Information System
Internal Legacy Information System
Internal Legacy Information System
Agency A
Agency B
Agency C

[1] RESTful: REST stands for "representational state transfer," a trusted industry standard for creating web services. A RESTful service is one that goes beyond simple URLs, applying full HTTP semantics (GET, PUT, DELETE, etc.) to provide service that is fast and reliable.

The difference is that while people can only visit one country at a time, a single government task can require information from several agencies. Agency A might need to talk not only to Agency B, but also to Agencies C, D, and E. That is why it is critical to employ an API platform that provides connectors to support a wide range of legacy systems. APIs should also be deployed close to each legacy system so that all conversations can run through them.

OpenAPI Specification (OAS) Support

Since legacy systems come in many different forms, essentially speaking their own different languages, it's important to begin the digital transformation by setting up the interfaces (the APIs) that will connect these systems to the outside world via the API platform. Ideally, the design of APIs should be coordinated so that the whole system works together to automate and streamline business processes as desired. This provides a high-level, "schema-first" approach to development, giving individual developers a plan to follow as they flesh out functionality.

Schemas should employ OpenAPI Specification (OAS), the broadly adopted industry standard. OAS is essentially a lingua franca designed to enable both humans and computers to understand the services the API will provide without the need for access to source code, extra documentation, or network traffic inspection.

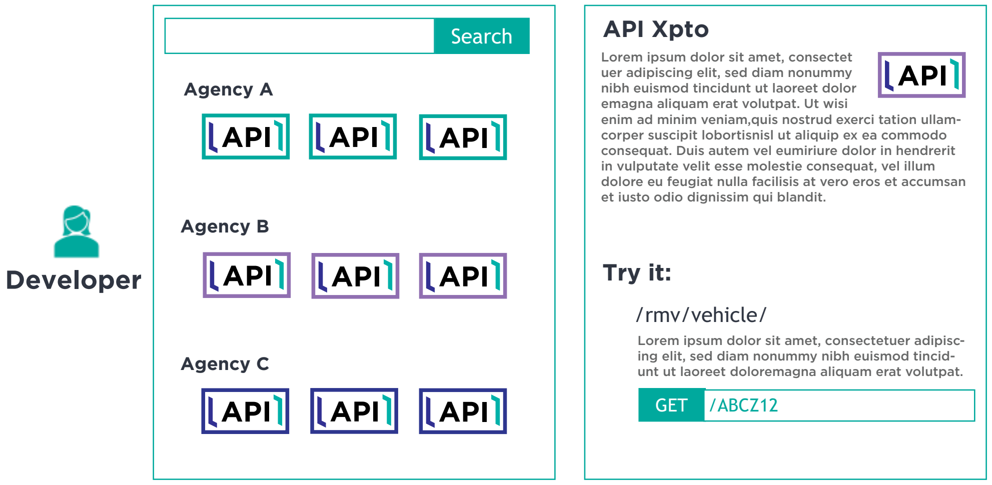
API platforms that support OAS provide streamlined development, coding, and testing.

THE BOTTOM LINE:

Processes should be built using a schema-first approach, and schemas should be OAS-compliant.

Developer Portal

Let's say that Agency A has developed an API that other agencies want to use. They can't use it if they can't find it. And if they find it, they will need documentation that explains what it does and how to use it, ideally with code samples that developers can use.



Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolor emagna aliquam erat volutpat. Ut wisi enim ad minim veniam,quis nostrud exerci tation ullamcorper suscipit lobortisnisl ut aliquip ex ea commodo consequat. Duis autem vel eumiriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet doloremagna aliquam erat volutpat.

That is why the API platform must provide a developer portal that all agencies can use to publicize their APIs to other agencies. This is the single go-to place to search for useful APIs.

Within the portal, a developer must be able to:

- Search for APIs
- Read API documentation
- Try APIs with sample data
- Call sample code with different programming languages
- Request access to an API

THE BOTTOM LINE:

API platforms should provide a developer portal where developers can see API documentation and examples of usage. The portal should make it easy for agencies to find, access, and use APIs for their applications.

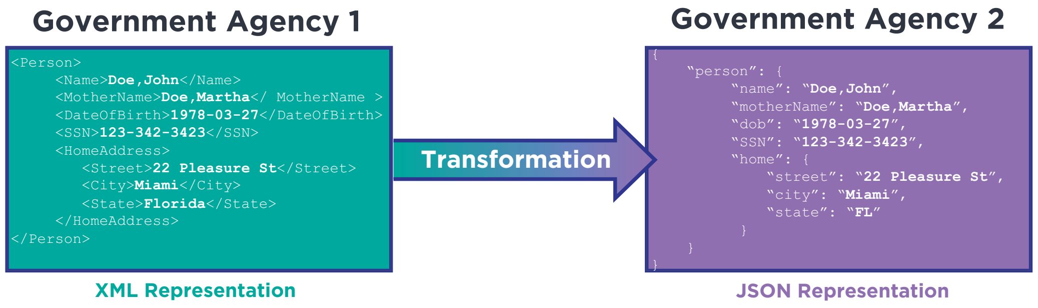
THE BOTTOM LINE:

The API platform must include mechanisms to visually build transformations between different data representations and data structures and to apply lookup tables to map custom codes to the reference data defined by the regulatory framework.

Data Transformation

The information systems used throughout the government were designed by different vendors and developed over decades to meet various distinct goals. Naturally, they have different ways of representing the data most useful to them. A good API platform therefore needs to work a bit like a translator helping people from different agencies and governments to communicate. It has to provide tools developers can use to transform data to ensure effective communication between systems that represent data differently.

This illustration shows how two agencies might represent data about a person. The legacy system for Agency A provides the data as an XML document (in green) and uses full names for the state (e.g., Florida). Agency B needs the data in JSON, a modern standard, and states must be represented by their two-character codes (e.g.: FL). That means that the data provided by the legacy system from Agency A needs to be transformed before it can be shared.



```
<Person>
     <Name>Doe,John</Name>
     <MotherName>Doe,Martha</ MotherName >
     <DateOfBirth>1978-03-27</DateOfBirth>
     <SSN>123-342-3423</SSN>
     <HomeAddress>
        <Street>22 Pleasure St</Street>
        <City>Miami</City>
        <State>Florida</State>
     </HomeAddress>
</Person>
```

```
{
    "person": {
         "name": "Doe,John",
         "motherName": "Doe,Martha",
         "dob": "1978-03-27",
         "SSN": "123-342-3423",
         "home": {
             "street": "22 Pleasure St",
             "city": "Miami",
             "state": "FL"
         }
    }
}
```

To transform the data from the legacy system, the API platform must provide standards for interoperability that ensure that any required transformations and lookups to the data make all the information appear the same.

Reliability, Scalability, and Performance

API platforms need to ensure that all API requests are delivered and processed, while also throttling and scaling requests when necessary to ensure legacy systems are not overwhelmed. Capabilities for doing this include:

- Data caching, replication, queueing, and horizontal scalability of pure services
- High availability through the use of service replicas

THE BOTTOM LINE:

API platforms should speed processing by caching and/or replicating the data from legacy systems and serving outside requests from other agencies. They should also allow queueing of incoming data and throttling or scheduling requests. They must allow for the horizontal scalability of pure services.

Data Caching, Replication, Queueing, and Horizontal Scalability of Pure Services

Let's say you are responsible for one agency and it has a legacy system that works just fine for your needs. Moreover, it holds data that other agencies could use, and you are willing to share that data with them under the right circumstances.

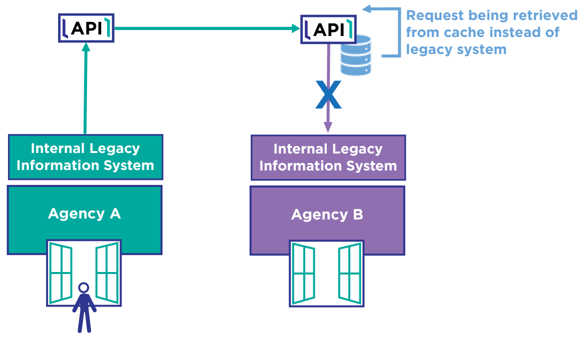
Here is the dilemma: If you let other agencies, or even authorized private institutions, access data from your legacy application, that means extra work for your application. This could affect performance and disturb the day-to-day work of your agency. That is why the proper configuration of your API is so important.

An ideal API platform should allow you to configure throttling of your new API to slow down requests to a pace your system can handle. If requests are coming in from other agencies at a faster pace than your legacy system can accommodate them, you can also use the API platform to queue them up for later consumption by your legacy application. You can even schedule the consumption of the data for specific times of the day that will not affect normal agency activities.

Throttling and queueing help developers work around the restrictions of the legacy systems. But, slowing down other agencies can violate service level agreements (SLAs) with those organizations. So, the question is: Do we really need to talk to the legacy system to fulfill all requests?

The answer is that some requests can be entirely fulfilled by the API platform.

The API platform can create a data cache or even maintain a permanent copy of the data from the legacy system. Let's say that the new API platform is keeping a copy of the data. Now, every time an external system requests data, instead of retrieving the data from your legacy system, the API platform simply serves it from its copy, enabling your legacy system to continue working as usual. The API platform is doing the hard work of answering all those requests for data from other agencies and private institutions.

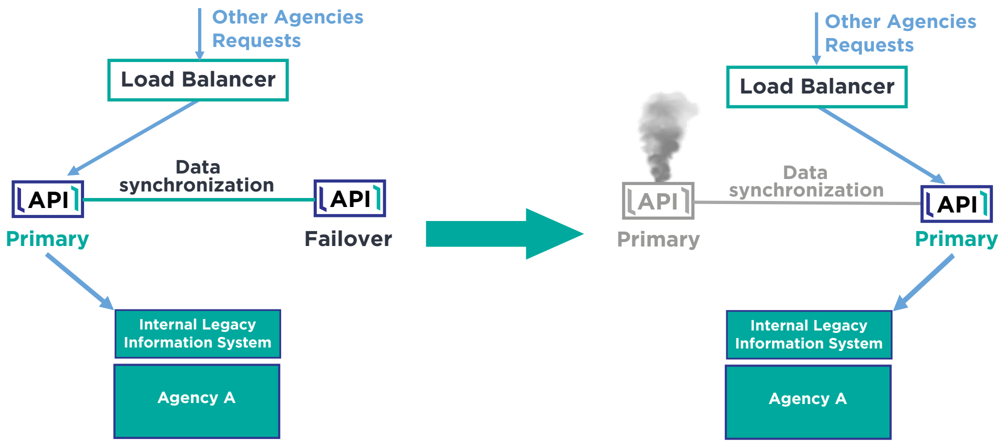

API
API
Request being retrieved from cache instead of legacy system
X
Internal Legacy Information System
Agency A
Internal Legacy Information System
Agency B

That is how we transform "dirty services" into "pure services"[2] — and pure services can always scale.

High Availability Through the Use of Service Replicas

As agencies gain interoperability, the API platform will become a critical component of day-to-day operations. In such a situation, failure is not an option. If the computer server running the API platform should become unavailable for any reason, there must be a replica server(s) available to take over immediately and keep all processes running smoothly.

In an ideal API platform, this is accomplished through load balancing between replicas without any shared architecture, a practice known as "shared nothing" synchronous data replication. This practice provides the government with the freedom to deploy the solution on premises and/or on any cloud vendor and to employ a combination of different deployment environments.



THE BOTTOM LINE:

The replica makes sure that all the code and data is always available on another server or cluster. This replica must be maintained by the API platform software without relying on shared infrastructure or a single point of failure (sharing the same storage area network). Such architecture is also known as "shared nothing" and allows the service to be deployed on commodity hardware on premises or in the cloud.

[2] Pure Services – Services that do not fully rely on legacy systems to work. A Pure Service may be serving data that is cached or replicated in the API platform. Data can be pulled from the legacy system for caching or replication in many different ways. Replication, for instance, can happen during hours of the day when the legacy system is not under the pressure of day to day work. Pure Services may also be 100% independent of legacy systems. That means that the service is 100% built and run by the API platform.

THE BOTTOM LINE:

An ideal API platform provides strong, consistent, and high-performance security infrastructure for applications and for data both in-transit and at rest.

Trust and Security

Since APIs are often transmitting sensitive information between agency systems, everyone participating must trust that the APIs are keeping information secure using the latest in security technologies. Naturally, much of the data inside the API platform will also be sensitive and should not be visible or accessible by any unauthorized personnel. Protecting this information requires strong data security and encryption capabilities at multiple levels.

The most fundamental security capabilities necessary are authentication, authorization, auditing, and managed key encryption:

- Authentication verifies the identity of all users.
- Authorization ensures that only specified users can access the resources that they need.
- Auditing involves logging predefined system and application-specific events and storing all requests and responses for the purposes of investigation (forensics).
- Managed key encryption protects information from unauthorized viewing.

Strong data encryption must be applied to all the logs, tables, and data structures that hold sensitive information. The encryption key must only be required when the API platform is started, and it cannot be stored on the file system of the server; that would be equivalent to leaving the key to the safe hanging next to the safe on the wall.

The API platform security architecture must also provide:

- Compliance with certification standards
- Ease of development for building security features into applications
- Low impact on performance and operations
- Effective and smooth operation of both the APIs and the platform itself within a secure environment
- Infrastructure for policy management and enforcement

The API platform should also provide something called “row-level security,” which restricts access to data based on who is authorized to view the information. With row-level security, each row in the database holds a list of authorized viewers (either users or roles), and safe access to information is supported by the use of SSL/TLS, the standard protocol for encrypting information in transit. Row-level security also provides tools for establishing a public key infrastructure (PKI), the protocol used to establish the identity of querying entities (such as another government agency).

Easy Management and Maintenance

While some government agencies centralize IT management, many do not. An API platform must allow agencies to work in either a centralized or decentralized manner so that each agency can maintain its existing structure and independence as needed. The platform also needs to support metering the consumption of services so that usage costs can be paid by consumers, and so that all SLAs are met. Key features include:

- Business monitoring and distributed system monitoring
- Business process orchestration and business rules
- Dealing with exceptions and providing forensic information

Business Monitoring and Distributed System Monitoring

As agencies start to rely on services from each other, it becomes critical to ensure proper governance. This includes:

- Monitoring and ensuring APIs can only be used by the authorized agencies
- Metering of services consumption so that the costs of these services can be paid by whomever is using them
- Monitoring of SLAs to ensure they are all met

It is up to the organization to decide if this monitoring should be done by each agency (decentralized monitoring) or if a central entity will take ownership of this task and monitor the health of the API platform deployed with all agencies from a central perspective (centralized monitoring).

Since most government agencies will want autonomy, however, a centralized monitoring system is likely to prove impractical. An ideal API platform should therefore provide tools for distributed (or decentralized) monitoring.

Key health metrics to monitor include:

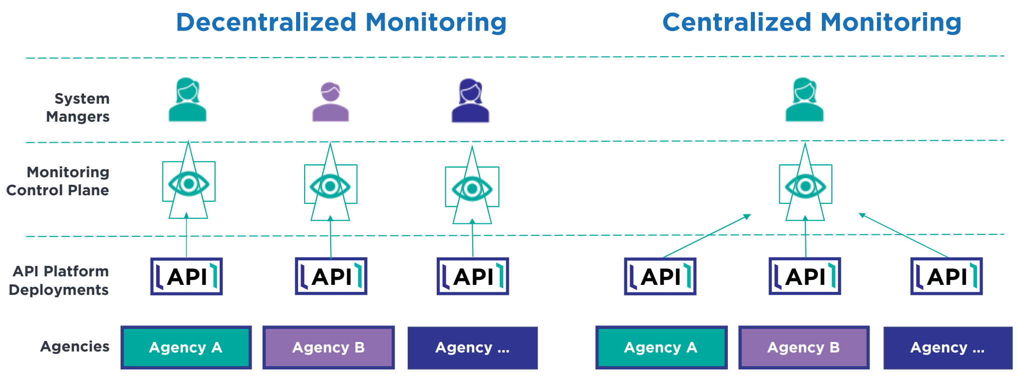
- CPU usage over time
- Memory consumption over time
- Disk usage and free space over time
- Database usage and free space over time (for cached and replicated data)
- Queue size over time

Each of these metrics needs to be collected for each server supporting the API platform deployment and communicated to the administrator(s) monitoring the system. Alerts must be sent to administrators when anomalies are detected, such as:

- Disk space running out faster than usual
- CPU usage spiking for a long time
- Memory running out and causing swapping
- Queue sizes suddenly increasing

THE BOTTOM LINE:

API platforms must include mechanisms for monitoring the APIs deployed by each agency. Business metrics such as metering of service usage and monitoring of SLAs are essential. Preventive monitoring of the individual API platform system deployments with each agency in terms of disk space, memory, CPU usage, health of the API platform, and system alerts must also be considered.

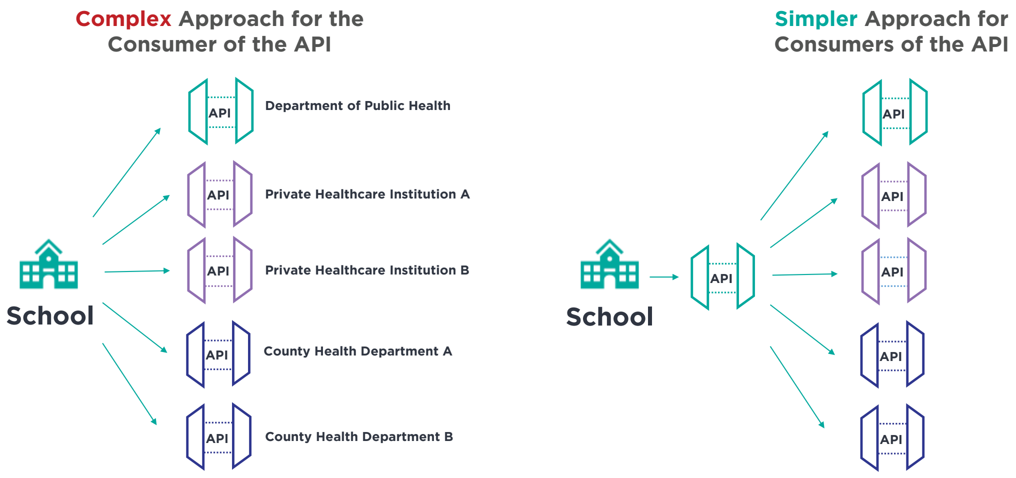


THE BOTTOM LINE:

API platforms must allow for the design and execution of business processes that are essential for orchestrating work that may involve many API calls, thus triggering data transformations, complex decision-making conditions and business rules, human workflow, and alerts.

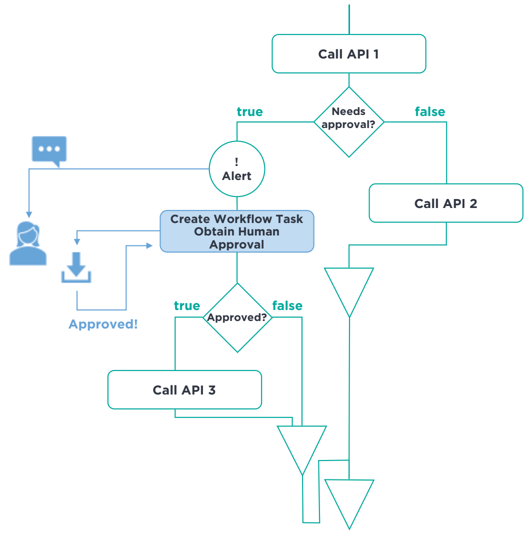
Business Process Orchestration and Business Rules

Ultimately, the ideal API platform aggregates and orchestrates the underlying services offered by various agencies to meet the needs of the government and citizens. Consider the following scenario: A citizen needs vaccination records to register his child for public school, but those records currently reside in five different agencies. What if the school information system could use APIs, provided by the agencies and institutions that are custodians of vaccination data, to fetch this information? Below is an example of how an API platform can improve the creation of an application that uses APIs to retrieve the data.


Complex Approach for the Consumer of the API
School
API
Department of Public Health
Private Healthcare Institution A
Private Healthcare Institution B
County Health Department A
County Health Department B
Simpler Approach for Consumers of the API

The illustration above reveals how APIs might serve this need without a platform. Each school system would have to call the individual APIs provided by custodians of information and wrangle with differences in how the information is presented. (This assumes the school is capable of calling an API, which may not be possible if it only uses the legacy system itself.) The challenges are compounded if all the schools in a county or in the state need to call all these APIs individually and attempt to transform the information returned from each.

An API platform addresses these challenges by allowing for the creation of aggregation services, also known as orchestration services or business processes.



In this scenario, there is only one API for obtaining vaccination information. When a school calls it, a process starts that calls the APIs at each agency and applies any required transformations and lookups to the data so that all the vaccination information appears the same. As a bonus, if a new custodian of vaccination information should appear and offer a new API for its data, the platform can simply add it so that all schools immediately benefit.

To speed development, an ideal API platform should enable developers to employ a graphical user interface (GUI) to visually lay out the steps of a process as a flow chart such as the one shown here.

The process might include such steps as:

- Making synchronous calls to external systems and published APIs
- Making asynchronous calls to external systems and published APIs
- Applying transformations to data
- Applying control structures such as IF conditions and loops
- Designing and applying business rules
- Triggering human workflows and alerts

When a developer saves these graphical processes, the system should automatically generate computer code, such as Business Process Execution Language (BPEL) that can be swiftly encapsulated and published as a single new API.

Dealing with Exceptions and Providing Forensic Information

No system is immune from error, and there are always exceptions to every rule. Thus, every API platform needs to independently send alerts to human administrators. There are many reasons for such alerts, including:

- A process has failed
- No new data has arrived in a period of time (signaling the unavailability of an external system)
- Too many messages are queued (signaling slowness of an external system)

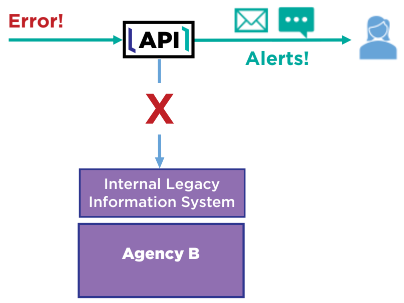
One of the ways of dealing with such issues is to start a human workflow. Messages and alerts can be sent via email, text, or other mechanism to inform specific individuals that pending tasks need human resolution.

The API platform must also provide full traceability. Every API call that retrieves or provides legacy system data must be logged. An administrator must be able to find the original exchange by searching for the contents of the call (e.g., searching for a SSN, VIN, vehicle plate, etc.) and by data ranges. This is very important to supporting the investigation of incidents such as:

- Data is not appearing correctly in the legacy system, and it is not clear if the fields were missing when the data arrived, the API platform transformed the data incorrectly, or the legacy system is displaying the information incorrectly.
- The wrong data is appearing in the legacy system. That may cause a serious problem for the agency consuming the data. If the agency can trace the data back to its source API and see that the data came in wrong from the other agency, it can defend itself in case of litigation.

THE BOTTOM LINE:

The API platform must provide mechanisms for sending alerts to humans and support processes that need human intervention. All data exchanged for a defined period of time must be stored for forensic analysis. This data must be easily searchable by data range, API, and contents.



This is a critical feature of the API platform since it can be used to pinpoint the cause of data corruption. The data logs must be able to be kept for long periods of time, as defined by the local administration. It is also important to be able to offload these logs to an external message bank or log sink, in case the data is required in the future for forensic analyses.

Conclusion

API platforms provide state and local governments with the ability to speed and streamline interactions with citizens, with other government agencies, and with the private sector. The best API platforms provide a range of powerful capabilities that enable fast and flexible development; high reliability, scalability, and performance; security for applications and for data in-transit and at rest, and streamlined management and maintenance, while enabling government organizations to retain complete control of their systems and their data.

For more information please visit: InterSystems.com/Government

About InterSystems

Established in 1978, InterSystems is the leading provider of data technology for extremely critical data in healthcare, finance, supply chain and other industries and throughout the public sector. Its cloud-first data platforms solve scalability, interoperability, and speed problems for large organizations around the globe. InterSystems is committed to excellence through its award-winning, 24×7 support for customers and partners in more than 80 countries. Privately held and headquartered in Cambridge, Massachusetts, InterSystems has 25 offices worldwide.

For more information please visit: InterSystems.com

Read the Companion Brochure: The 10 Key Features of an Ideal API Platform

InterSystems®
Creative data technology