**IBM Cloud App ID**

# Engage your Users and Secure Applications with IBM Cloud's Solution for Application Identity

Gautam Zalpuri
Martin Smolny
John McAuley
Vladimir Atansaov

IBM | February 2023

# Contents

# Introducing IBM Cloud App ID

Great apps are focused on making the customer experience better. Or making life easier for employees. In either case, no matter how great or transformative your idea is for an app, the success of your app depends on your ability to build trust with your users. This trust is built, in part by protecting their data and controlling access to resources and transactions, and by delighting users with a tailored experience that simplifies, personalizes, or improves their efficiency. Knowing who is using your app is central to both.

When you know who is using your app, you can apply the right security controls and build a customized experience for your users. To achieve this, you need to add authentication (sign-in) and authorization (access permissions) to your app, as well as manage digital identities. For example, you might remember information that a user shared with you, such as their preferred checkout method. But, when it comes to security, designing it into an app is often frustrating for developers as it is risky, complex, and filled with edge cases. To fill the need for a simpler way to introduce authentication and authorization into your apps, we've delivered a solution - IBM Cloud App ID.

As a developer, you can use App ID to easily add authorization and authentication to your web, mobile, API, and back-end applications. You can also use App ID to host user data, such as application preferences, that can be leveraged to build custom application experiences.

The service gives you many ways to make the sign-in experience easy:

- Give your users the option of signing up directly from your application with their username or email address and thereafter use their username/email and password to sign-in.
- Leverage the SAML 2.0 Identity Federation capabilities that App ID provides and allow your users to sign-in with their existing enterprise credentials.
- Let your users use their existing social accounts such as Facebook or Google.
- Connect to any existing custom identity system, even if it uses a fully proprietary protocol.

Regardless of the way your users sign-in, App ID allows you to enforce access policy requirements for your back-end applications and APIs, so only properly authenticated and authorized users will have access.

App ID also helps you to deliver a personalized experience for your users based on a variety of factors. With App ID you can store information about your users and let your developers leverage this information to make their apps better.

## The Rising Need for Application Identity

As organizations look to better serve their customers and employees through digital apps and services, one key theme that emerges is a laser focus on the customer experience. Since superior digital experience starts with knowing your users and building trust, being able to manage digital identities is critical. To drive higher lifetime customer value, organizations are seeking to continuously refine and personalize the conversation with their target users. This is not possible without collecting and analyzing user preferences as well as application usage data.

Moreover, organizations are trying to move quickly to stay competitive, working to deliver leading apps and APIs that bring to life innovative business models before they are outpaced by traditional competitors or industry disrupters. In pursuit of a greater agility and speed, more and more customer-facing interfaces are incarnated as collaborating networks of micro-services.

As such, we've observed several trends transforming the world of corporate IT that challenge existing enterprise identity and security infrastructure and practices across multiple dimensions:

**Experience is Key** – Security needs to blend seamlessly into the user experience. Generic username and password screens no longer suffice. At the same time, you must instill confidence in your users that you are protecting their sensitive data.

**Cross-channel Personalization** – Reaching users across digital channels requires more than managing cross-channel identities. Application developers need better access to user profiles and properties linked to these identities (with privacy and legal compliance) to create personalized experiences.

**Solution TCO** – Many of the existing enterprise solutions are too expensive to serve digital channel patterns of use. There may be a vast number of digital consumers who use your application infrequently, or few employees who use your application daily. Pricing schemes need to acknowledge this. An equally important cost consideration is that organizations can't afford to service many digital consumers that are flooding their traditional support channels. Providing consumers with self-service is crucial.

**Programming Model Integration** – Security is traditionally enforced in the DMZ layer, while subsequent layers are considered more secure. This model assumes a rigid gateway and layering structure that fails to address the need for agility and the evolution of micro-services. Security infrastructure needs to be integrated into the new programming models and run times used to build the micro-services.

**Scale and Performance** – Demand on performance and scaling are much greater now that the number of consumers is growing constantly.

**Cloud and Multi-Cloud Deployment** – With your business logic running on the cloud, it must be accompanied by strong, cloud-based identity and security services to retain overall agility. The solution that you pick should be able to scale as your business grows. App ID runs on public cloud, but we do not lock you into a particular cloud provider or runtime such as Cloud Foundry or Kubernetes. You can use App ID with any application, written in any language, that runs in any environment – public or private.

All the above, and many additional, emerging needs are addressed by the App ID service, which is deeply integrated into the overall IBM Cloud programming model for mobile, web, back-end, and microservice-based applications.

# App ID Tenets

App ID is a SaaS, cloud-native offering provided by IBM Cloud. It can be used to easily protect any application that runs on any cloud, with any kind of identity provider. The service is led by people with vast experience in building cloud-native services and security solutions. Our product's development and design teams combine some of the best minds that IBM has to offer, working as a single, breathing organism.

Many open-source solutions provide packaged software that developers are required to host and maintain themselves. We're taking a different approach to ensure best-in-class, reliable, scalable, highly available and fully managed cloud identity, and access control service that customers can trust. To ensure you that can depend on us, we've shaped our development, operations, and support processes to address our goal.

## Open Specifications and RFC compliance

App ID is based on a set of well-known, industry standard protocols and specifications frequently found in both consumer-facing and enterprise-facing applications. Being compliant with industry-standard specifications is crucial for us and we want to allow developers to leverage any compatible SDK and consume our APIs in a way that they're already familiar with. We don't want to re-invent the wheel.

At the core of App ID, you'll find The OAuth 2.0 Authorization Framework (RFC 6749), and Open ID Connect. The former allows applications to obtain and verify authorization for accessing protected resources, while the latter is responsible for adding an authentication and identity layer.

Being compliant with these two major specifications allows App ID to integrate seamlessly into practically any existing authorization / authentication eco-systems.

Some of the other specifications App ID is based on include:

- The OAuth 2.0 Authorization Framework: Bearer Token Usage RFC 6750,
- OAuth 2.0 Dynamic Client Registration Protocol RFC 7591
- OpenID Connect
- JSON Web Algorithm (JWA) RFC 7518
- JSON Web Token (JWT) RFC 7519
- JSON Web Signature (JWS) RFC 7515
- Proof Key for Code Exchange by OAuth Public Clients RFC 7636
- OAuth 2.0 Token Introspection RFC 7662
- Assertion Framework for OAuth v 2.0 Client Authentication and Authorization Grants RFC 7521
- JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants RFC 7523
- Security Assertion Markup Language (SAML) 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants RFC 7522
- System for Cross-domain Identity Management: Definitions, Overview, Concepts, and Requirements RFC 7642
- System for Cross-domain Identity Management: Core Schema RFC 7643
- System for Cross-domain Identity Management: Protocol RFC 7644

Being compliant with these specifications is not easy as they constantly evolve. With App ID, you don't have to spend time becoming a security expert. You don't need to learn and implement each of the specifications – you can focus on bringing value to your customers and let us handle authentication and authorization.

## API documentation and open-sourced SDKs

App ID goes beyond the above specifications to bring more value. Many of the above specifications allow extending functionality and describe facilities to do so.

Most of App ID's capabilities are built according to these specifications. In some cases, we've added capabilities that haven't been described in any existing specification yet. For this reason, we've decided to take an open approach – App ID server-side APIs are well documented using OpenAPI Specification (Swagger), and all the App ID SDKs are open-sourced. To make it even easier, we provide a Postman collection that covers all the App ID APIs. All the above artifacts are available in our GitHub repo at https://github.com/ibm-cloud-security.

We don't want to re-invent the wheel, but we're certainly working to make it better.

## Development Practices

We've established a set of secure development practices that we adhere to and constantly update. We continuously monitor industry coding best-practices, such as the OWASP project. When it comes to sensitive environments and data access, we've established an access-control system and follow the principle of least privilege. Our environments and micro-services are instrumented with detailed logging that helps us to debug and audit anything that happens in our systems.

We're proud of the product that we've built; of the code that we've written, and that we continue to write. Peer code reviews are a mandatory step in delivering the code. This is part of our nature. Our test coverage is high. We have a set of quality checks along our continuous delivery pipelines to ensure a high quality of the delivered code and catch possible issues at very early stages of development. We continuously re-evaluate App ID components to ensure smooth continuous integration flows. Our developers feel confident delivering the code, knowing that our automation will catch any possible defects long before it affects customers. We operate multiple environments used for development, staging, load and performance testing and production.

We have several support and communication channels to ensure timely responses and awareness of any issues that might be encountered by our users.

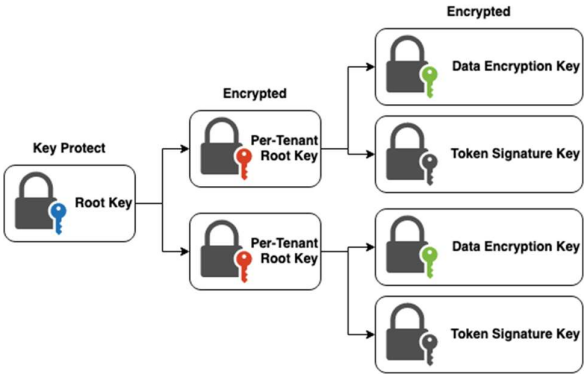
## Cloud-Native Operations

App ID is a scalable, cloud-native service that has been implemented according to the best practices of a 12-factor application (https://12factor.net).

The service is built as a suite of containerized micro-services that are orchestrated by Kubernetes Service. It is highly available, with multiple Kubernetes clusters deployed in each geographical region. Each cluster has at least five worker nodes, two of which are defined as edge nodes. We use Global Load Balancer, CDN, WAF and DDoS protection layers in front of App ID clusters. In addition, each App ID Kubernetes cluster is protected by a set of Calico networking policies restricting all network traffic to approved origins only. A service mesh with boundary protection is achieved by utilizing the Istio Service mesh framework.

App ID clusters are monitored using New Relic and SysDig. For each micro-service, we run a deployment with a liveness probe and a minimum of three replicas. Every container running in our clusters is monitored by two systems – IBM Cloud internal monitoring and New Relic. In addition, we have a self-contained sanity test system in place that executes the most common App ID workflows in all production regions every few minutes. Whenever a component fails – it is automatically recycled with minimal customer impact.

All App ID micro-services are fully stateless. We use NoSQL databases and Redis deployments hosted in the same geographical region to store and cache data. Incoming and outgoing App ID traffic is encrypted with TLS 1.2/1.3; any non-TLS 1.2 clients are rejected. All the databases that App ID uses are encrypted by default (data-at-rest). In addition to full database encryption, App ID also ensures full per-tenant data isolation by encrypting the data with tenant-specific data encryption keys thus making it accessible to service tenant only. App ID uses IBM Cloud Key Protect, an HSM based service, to manage root keys and data encryption keys. Each tenant receives its own root key, which is used to wrap tenant- specific data encryption and token signature keys.




All databases that are used by App ID are backed up daily and a secure copy of the backup is stored across multiple regions as part of our disaster recovery protocols. App ID stores data on a regional basis, meaning data for US service instances is kept in the US, data for EU service instances is kept in the EU, et cetera. All databases used by App ID spawn at least three availability zones per region.

App ID uses IBM Cloud Secrets Manager to store the internal credentials required to access its dependencies. As mentioned above, each region has its own set of dependencies, with different credentials. This allows us to ensure that credentials are kept safe, and no unauthorized party can access them.

Being a cloud-native service is all about continuous integration and continuous delivery, and we are heavy CI/CD advocates. We have fully automated the App ID delivery process from source code to production deployment. In our workflow, we use GitHub Enterprise pull requests to manage our source code, Tekton for continuous integration and continuous delivery, a set of IBM-internal and industry standard tools for quality control (DevOps Insights, SonarQube and more). Our build cycles are short and test coverage is sky-high.

We use the Green/Blue deployment model to ensure that the updated environment is fully tested before it starts to accept requests. Whenever a new version of a particular component is deployed, we run automated test cycle to validate there are no regression defects and to test new functionality. To ensure continuous service consumability and full system functionality we're running an automated sanity test suite every few minutes in all production regions. In addition, we employ a third-party company to run periodic penetration tests on all our systems.

As mentioned previously, we use several monitoring techniques to ensure we're aware of everything that happens in our environments. While most of the issues are addressed automatically, sometimes human intervention is required. When automation can't fix the problem by itself, it will notify the IBM Cloud Support team and App ID team on-call personnel, both available 24/7. We use a combination of Slack and PagerDuty to ensure timely issue resolution. For every single issue that required human intervention we perform a root cause analysis to understand what the core of the issue was, how did it happen and how can we prevent the same problem from happening again.

Having a proper cloud-native, microservice-based architecture, fully automated continuous integration and continuous delivery cycles, as well as an exhaustive monitoring system allows us to minimize any possible blast-radius, ensure secure and effective operations, and provide our customers with an identity service that they can place their confidence in.

## Certifications and Compliance

App ID has successfully completed the GDPR, HIPAA, PCI and SOC2 certifications and audits. Additionally, we've completed:

- *ISO 27001 (Includes GDPR)*
- *ISO 27017*
- *ISO 27018*
- *ISO 27701*
- *CSA STAR Level 1 (Self-Assessment)*
- *HIPAA for Healthcare USA*
- *PCI-DSS for Payment Card Industry - as a Service Provider*
- *SOC1 Type 2*
- *SOC2 Type 2*

# Features, Capabilities and Technology

App ID provides a collection of authorization, authentication, and identity related capabilities for developers. With App ID, developers can easily introduce consumer and enterprise class authentication. Developers can enable progressive authentication for their cloud applications and microservices, so that they can deliver engaging user experiences across multiple digital channels. The capabilities provided by App ID include:

## User Authentication

Getting users to sign-up and sign-in to your applications is clearly central to being able to provide customized, tailored experiences and protect applications, APIs, and back-end resources. Therefore, user authentication is at the core of App ID.

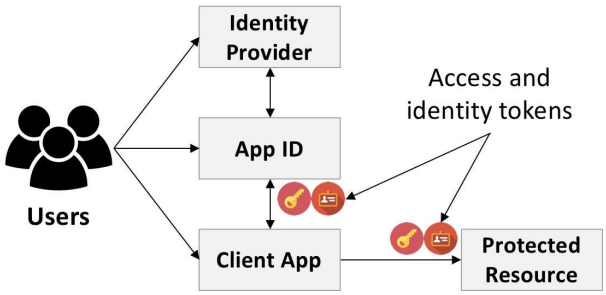
While some might consider the subject of authentication to be simple, it can rapidly get quite complex when you dive into the implementation details. Do you implement a proprietary authentication mechanism, or do you comply with something like OpenID Connect or SAML? How do you federate identities from the identity provider or user repository? How do you ensure identity information is not compromised? How do you establish trust between your authentication mechanisms and back-ends? How much effort you need to spend to add HIPAA, GDPR, SOC2, PCI and other certification badges to your product? Those, and many others, are the questions that you'll need to answer.

App ID addresses all those concerns. With App ID, application developers can easily add user authentication to web and mobile applications without needing to have specialized security knowledge and protect backend APIs in their cloud applications in minutes.

App ID uses industry-standard, customizable approach to ensure high-level of trust between your application components. Your users can authenticate against an existing user registry, or start from scratch with Cloud Directory – a user registry provided by App ID.

## JSON Web Tokens

App ID is built on industry-standard specifications such as OAuth2 and OpenID Connect. Under the hood, App ID provides a standardized way of obtaining information about user authorization and authentication - access and identity tokens. The tokens are implemented as JSON Web Tokens (RFC7519) that clients received after a successful user authentication. Both tokens are cryptographically signed with tenant-specific keys to prevent tampering and contain a set of claims that help developers to customize application behavior and make business logic decisions.




An access token represents authorization to the bearer. It contains claims (JSON properties) that describe who the token was issued by and who it was issued for, the intended audience, what is the token expiration timestamp, which authentication method was used, what is the authorization scope and more. Access tokens are used to understand what user is authorized for.

An identity token represents authentication. It contains most claims found within access tokens, but also includes user identity claims, such as name, email, locale, picture. Identity tokens are used to understand who the user is. User identity claims come from identity providers and depend on how you want your users to authenticate.

Different identity providers implement different authentication protocols and return different user profile claims. App ID normalizes those user profile claims in its identity token, so that developers using it don't have to be concerned with differences between identity providers.

App ID supports token customization which means that you can customize your token expiration and add custom claims or attributes. By default, we only inject a normalized subset of the information that is returned by an identity provider into your tokens. But you can easily configure App ID to inject any other claim or assertion that is returned by an identity provider, or even a custom attribute that you created, into your tokens.

The above tokens are at the heart of App ID's authentication and authorization process. In most cases tokens are handled automatically by the App ID SDKs. However, developers have full access to these tokens should they need to implement highly customized, advanced business logic.
In addition to access and identity tokens App ID also providers anonymous tokens that allow to support anonymous users, and refresh tokens that allow to refresh existing tokens without asking users to re-authenticate.

## Application Authentication

There are several reasons that you might want one application to communicate with another service or app without any user intervention. For example, a non- interactive app that needs to access another application to perform its work. This could include processes, CLIs, daemons, or an IoT device that monitors and reports environment variables to an upstream server. The specific use case is unique to each application, but the most important thing to remember is that the requests are exchanged on behalf of the app, not on an end user, and it is the app that is authenticated and authorized.

App ID allows you to use application credentials to retrieve an access token that uniquely identifies your applications. By using this access token, your protected resources can ensure that the request is originating from a trusted, authorized source.

## Custom User Attributes

As mentioned in the previous section, App ID provides identity token with claims about the user's profile. However, in some cases you might want to add custom claims. These are highly specific to your application

and as such App ID is not aware of them. Usually, these claims are application specific user preferences, such as preferred text size, shopping cart content, application language, et cetera.

To address this use-case, App ID provides support for custom user attributes. Using a simple CRUD API, available as a REST API and as a programmatic API in the App ID SDKs, developers can store any custom properties associated with a user identity, that are then immediately available to all instances of your application. For example, if users access your application from both web and mobile platform, you can store a custom property in the web version of your application, and it will be immediately available for retrieval in the mobile application.

## Access Controls

With custom attributes, you can use App ID to provide highly flexible access control capabilities. You can define roles, scopes, permissions, and groups of your users, and have those attributes automatically injected into authorization and identity tokens. With this information available in the token, your application can make quick business decisions about whether user access should be granted or denied.

If your existing federated identity provider, such as Enterprise SAML, already provides user information that can be used for access control such as roles or scopes, App ID can inject this information into the tokens automatically making integration even more seamless.

## Anonymous User Identity with Progressive Authentication

Authentication doesn't have to be first thing that pops-up on your application landing page. User experience research show that users should not be asked to sign-up or sign-in prior to taking an action that requires it. But how can you provide a tailored experience to a user if you don't know anything about them? In order to address this use-case App ID supports anonymous user identities with progressive authentication. You can immediately assign your users an anonymous identity when they begin using your application and start saving information anonymously - user or application preferences, shopping cart content et cetera. This ensures that a user can return to your app later and have their experience tailored to them based on the information that was previously collected, even though they never actually sign-in.

As users navigate your app, you can ask them to authenticate at any time. For example, when they need to make a payment. This is known as progressive authentication. After a user authenticates, the anonymous identity is converted to a known identity, and any information that was previously collected about that user remains accessible to you.

## Identity Providers

Authenticating users implies being able to validate their credentials and retrieving information about an authenticated user identity. Information about user identities is supplied by Identity Providers – systems that allow to create, manage, and maintain identity information and provide ability for relying parties, such as App ID, to leverage that information for authentication.

App ID supports multiple Identity Provider types and allows seamless Identity Provider replacement and reconfiguration. You can reconfigure existing Identity Provider or replace it with a different one without changing your code or redeploying your applications.

### Cloud Directory

While App ID allows you to leverage existing enterprise and social user repositories, you might want to start building a new, scalable user repository from scratch and keep all the user information in the cloud. In order to address this use-case App ID provides the Cloud Directory – a cloud-hosted user repository that you can use to host accounts for your customers, employees, and users.

As you build out your digital channels and grow your user base, a demand for customer self-service will likely grow as well. You might be able to manually manage dozens of users, but when it comes to thousands and millions of users, providing self-service capabilities is key.

App ID Cloud Directory is based on the SCIM specification and is equipped with administrative management APIs. It also has user self-management capabilities such as sign-up, password change and reset, update user profile and more. You can configure Cloud Directory to allow or disallow user self-service, send confirmation and welcome emails and more.

Based on your application requirements, you might want to use advanced Cloud Directory features such as password policies or multi-factor authentication. With password policies you can define minimum password length, acceptable characters, how frequently password change must be enforced and other rules. Multi-factor authentication allows you to enforce a second factor, such as email or SMS message as an additional user identity confirmation channel.

When using App ID Cloud Directory, the user information is automatically encrypted with tenant-specific data encryption keys and continuously backed up.

### Enterprise Identity Federation – SAML2

Many enterprises want to reuse their existing Identity Providers or User Repositories and federate them into the cloud. This is a common scenario for employee applications – you want to allow your employees to use existing credentials they're familiar with and use daily and have the sign-in experience they're used to.

App ID allows organizations to bring their own Identity Providers through the industry standard SAML2 protocol. Once configured this newly enabled SAML2 identity provider becomes available for immediate use in new and existing applications instrumented with App ID.

### Social Identity Providers

Social identity providers, such as Facebook and Google, have been adopted by a very large number of users, and while the jury is still out on their security value, they allow for easy sign-in to applications using existing Social Identities and sometimes this is exactly what users want. Social identity providers allow you to glean additional profile information, with user permission, that you can leverage to build personalized experiences and make internal business decisions.

App ID comes pre-integrated with a collection of social identity providers available for instant out-of-the box use without any pre-configuration. App ID supports multiple identity provides, so that you can allow your users to decide which social identity provider they want to sign-in with.

### Custom Authentication

While most identity providers today have adopted standard protocols such as SAML and OIDC, we've seen many cases where enterprises have a legacy or proprietary identity system with a completely custom authentication protocol. Integrating these kind of custom identity providers in every application that you have can be painful, which is why App ID provides custom identity provider support.

App ID allows you to use any identity provider, regardless of what protocol it uses for authentication, to obtain standard access and identity tokens, in order to leverage the service's capabilities.

## Personalized Digital Experience

Application users expect consistent, personalized, and relevant experiences when they interact with the various digital channels you provide - web, mobile, bots, API, et cetera. To achieve this, you need to have a centralized view of the user that spans across all your channels, including their preferences and

engagement status. App ID allows your developers to be able to easily access this information when building business logic into their applications. This includes:

- Normalized user information such as their name, email, and picture are usually obtained from an identity provider
- Organizational information and user preferences such as an address, or contact preferences
- Custom business-related data that is important in the context of your applications that you might want to persist between invocations

By requiring sign in through an identity provider, social or Cloud Directory, organizations can build user profiles that contain personal data and app preferences. Using the SCIM standard as the main connection, helps to access and manage personal user information and preferences.

## SDKs and Authorization Filters

App ID provides a set of easy to use open-sourced SDKs for iOS and Android applications, as well as web applications and back-ends that are implemented in Node.js and Java. Developers can easily install the App ID SDKs into new or existing applications using industry-standard package managers, such as Cocoapods, Gradle, or NPM.

The App ID SDKs abstract technical complexity by implementing the supported OAuth2 and OpenID Connect flows, as well as App ID specific features, such as the Login Widget, anonymous user identity, progressive authentication custom user attributes, user self-service and more.

Our server-side SDKs (Node.js, Java) provide authorization filters that can be used by developers to protect their APIs and web applications. With a few lines of code developers can define policies in a language they feel comfortable with to protect their applications (e.g. passport.js strategy for Node.js applications), and then leave the complexity of validating authorized requests and managing authentication flows to the App ID SDKs.

Using App ID SDKs developers can instrument their applications with authentication and authorization functionality with few lines of code without learning all the details on how things work “under the hood”.

If you’re building an application in a language we do not have an SDK for yet, you can still use App ID. Since App ID is OAuth2 and OpenID Connect compliant, you can choose to either leverage an existing third-party OAuth2/OpenID Connect SDK available for the language you’re using or alternatively consume the App ID REST API directly.

## Login Widget and UI Customization

At the heart of the App ID SDK is the App ID Login Widget, a dynamic authentication UI component, that’s available for both web and mobile applications. The Login Widget UI is built dynamically, based on your identity provider configuration. The Login Widget picks up any changes that you make automatically which means that you can update your authentication configuration without re-building and re-distributing new versions of your applications.

The Login Widget allows users to sign-in with any configured Identity Provider. In addition, the Login Widget provides Cloud Directory self-service UI, such as sign-up, password reset and more.

For applications that require a fully branded UI, App ID provides a set of easy-to-use REST APIs that conform to industry-standards. With Cloud Directory as your identity provider and these APIs at your fingertips, you can implement a fully customized self-service UI where your users can sign-up, sign-in, change or reset their password, update their profile and more.

As previously described, the Login Widget is available both as part of App ID SDK and via REST API, which allows you to leverage it regardless of the language your web application is implemented with.

### Using Gateways: IBM Kubernetes Service

App ID is integrated with other IBM Cloud components, such as API Connect, Kubernetes Service and Code Engine.

You can add App ID protection to your apps, APIs, and backends that run in IBM Cloud Kubernetes Service clusters by creating a declarative configuration for the Ingress Controller that enforces policy-driven security in a consistent way. You can do all of this with zero code updates which means that you don't have to instrument each of your applications separately or even redeploy them. All authorization and authentication flows are handled for your apps by the Ingress Controller.

### Easy App ID configuration and management

App ID comes with a simple to use web-based dashboard where you can configure authentication properties, setup identity providers, download samples, customize Login Widget, manage Cloud Directory and more.

The management functionality provided by the App ID dashboard is also available as a set of secured REST APIs that you can use to automate management activities in your DevOps processes, such as exporting and importing your configuration on demand or updating test and production instances of the service with the latest from development streams.

App ID supports the automation of these activities and their integration into a cohesive DevOps pipeline via REST interfaces and a Terraform provider. Everything that can be done with App ID via desktop browser, can also be done with REST APIs, and can be automated and integrated to match the needs of the development and operations team.

### Event Audit

As a part of IBM Cloud Platform, App ID is integrated with the Activity Tracker service. All the management actions that are performed on App ID instances, such as configuration change, user creation, profile updates are immediately reported to Activity Tracker with a high level of detail. Developers and security personnel can view those events as they occur in the Activity Tracker service or export them to a third-party tool.

## Using App ID in your Applications

Developers can incorporate App ID and its capabilities into their applications in different ways, as described below.

REST APIs – App ID provides a well-documented REST API based on specifications such as OIDC, OAuth2 and SCIM. Developers can invoke REST interfaces directly, or use an available, standard compliant library to incorporate App ID into an application or micro-service. To support this option, App ID provides Swagger based documentation that can also be used as an interactive exploration and test tool.

App ID SDKs – While the REST APIs provide a flexible interface, the developers using them need to code the interactions on their own, including responding to challenges, presenting UIs, caching tokens, and associating them with requests. To abstract away the technicalities of underlying specifications, App ID provides client and server-side SDKs for iOS, Android, Java and Node.js applications. Developers can use these SDKs to easily incorporate user sign-in into their applications, make use of user profile claims in application logic, and add security and personalization logic into their micro-services.

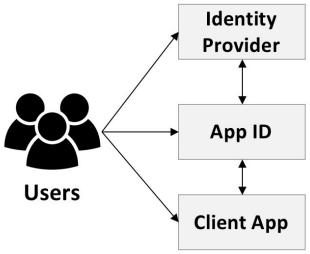
3rd party SDKs/frameworks – If you're building your applications in a language that we do not provide an official SDK for, you can still leverage App ID. Instrument your application with any OAuth2/OIDC 3rd party SDK or framework and configure it with credentials of your App ID instance. Being OAuth2/OIDC compliant allows App ID to be used in any application written in any language.
IBM Cloud Environment – App ID is integrated with multiple other products in IBM Cloud, such as IBM Cloud Container Service, IAM and Watson Data platform. This integration can be used to enforce policy-driven security in a consistent manner using declarative approach at the general gateway entry point instead of managing each application separately.

# Usage Patterns

Depending on the requirements and use-cases that you're trying to implement, the App ID SDKs provide support for different authentication and authorization integration patterns. Each of these patterns represent a separate scenario, however they can be combined to create a more robust authorization and authentication flows. Using several patterns in synergy will provides additional value and, depending on your use-case, can help to improve your overall security posture.

## Identity Only

Some applications might only want to leverage the authentication and identity capabilities of App ID.

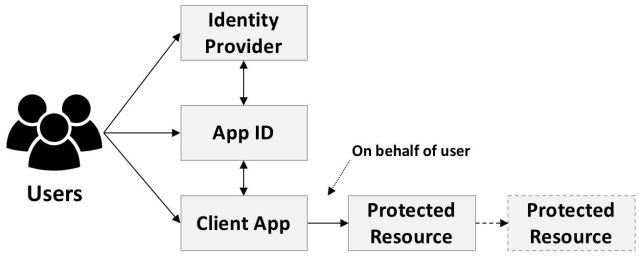



An example of such use-case might be an application that doesn't need to access any protected back-end APIs, but still wants to the customize application experience for users based on user identity.
Such an application, web or mobile, can use the App ID SDK to invoke the Login Widget, and after a successful authentication will use the claims from the identity token. It is up to developer to decide whether the Login Widget should be invoked immediately on application load, or at a later stage, for instance when a user explicitly clicks a sign-in button.

In other cases, applications will likely need to access protected resources, such as back-ends and microservices that run on cloud. App ID SDKs help to support this pattern as well.

## Client to Protected Resource on Behalf of User

This pattern includes a client-side agent, such as a mobile application, that invokes a protected resource, such as back-end API, on behalf of the end user. When using the App ID SDK, after a successful authentication the access and identity tokens provided by App ID are automatically added to the protected resource requests, meaning that subsequently the client application makes requests to the protected backend resources on behalf of the user.



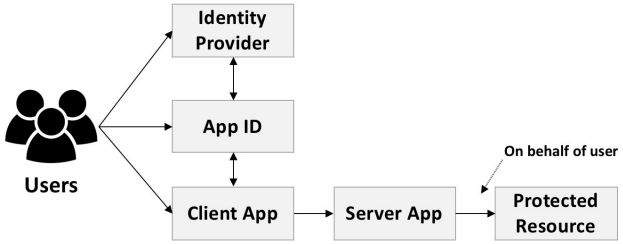

Upon receiving these tokens, a protected resource can easily validate them with authorization filters provided by the App ID SDK, or by following guidelines described in specification documents and using JWT/JWS encoded claims. In addition, as mentioned above, when using a protect-at-gateway approach, e.g. with Kubernetes Service, the token validation is done automatically for you.

Note that resources can propagate user authorization access to additional downstream layers of protected resources. This can be done by using the received access token in subsequent resource requests. This allows all services to use the same standard security and identity logic - independent of where they are in the microservice architecture and provide an easy way for downstream services to get user authorization information.

This pattern is suitable for client-side agents that can be trusted with access and identity tokens, such as mobile applications.

## Server to Protected Resource on Behalf of User

Not all client applications should be blindly trusted with access and identity tokens. Depending on the use-case you might want tokens to remain at the server side in order to not expose them.




With this pattern, a user operates a client application, which communicates with the application server. Whenever access to a protected resource is required, the client application makes request to the server application, which in turn makes a request to the protected resource on a user's behalf.

Similar to the previous use-case, access and identity tokens received by a protected resource can be further propagated by making requests on a user's behalf, downstream to other protected resources.

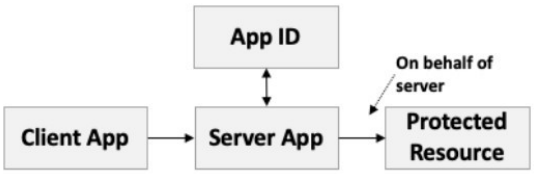
This pattern is suitable for client-side agents with a lower level of trust, such as browser applications (including single page applications, or SPAs). Depending on the use-case, developers can still decide to expose tokens to browser applications but reduce the token expiration time for better security.

A common way for developers to implement the above patterns is to use the App ID SDKs that are available for iOS, Android, Node and Java. Another way is to follow a language agnostic approach and use the REST APIs directly.

## Server to Protected Resource on Behalf of Server

Users are not always involved in every workflow. Occasionally, developers might want to use App ID to protect a resource that does not require user authentication or identity, but still needs to be accessed in a

secure way. Consider a micro-service or an API that provides a list of products. All users should see the same list, however in your system users don't access this API directly – instead it should only be accessed by an authorized server app which then sends the information to the users.




Another scenario is a developer that needs to isolate a certain segment of the microservice mesh to ensure that it can only be called by a particular microservice and not by user. A basic example of this use case is exposing administrative APIs that should only be accessible by authorized applications.

App ID provides support for this use-case by supporting client credentials OAuth2 flow, which allows OAuth2 clients to authenticate with their client ID and secret. After successful authentication, your application receives an access token that represents the application itself. This token can be used to make requests to protected resources the same way as in previous patterns. You can use the same authorization filters provided by the App ID SDKs, and propagate tokens to downstream protected resources, if required.

## Summary and Next Steps

Today's digital ambitions require more robust identity and personalization capabilities than traditional enterprise identity management solutions provide. Organizations need to protect access to sensitive resources for a wide range of customers and employees without affecting their users experience and build a unified view of the customer so they can provide consistent and engaging interactions across channels.

Capabilities provided by App ID cover the key requirements driven by the new digital channel workloads, and the technologies that App ID uses allow easy integration into any existing or new eco-system. App ID makes it easy to interact with users across digital touch points, while securing their access in an industry standard way. The service provides the ability:

- To add access controls to your apps, by quickly verifying users with your choice of identity providers.
- To personalize engagements with user profiles.
- To enhance developer productivity through SDKs and standard based REST interfaces.
- For cloud-based agility, speed, and TCO, while reusing existing investments in identity.

As you build your apps and microservices, check out IBM Cloud App ID, available in the IBM Cloud Services Catalog. We'd love to hear your feedback and questions. For technical questions at StackOverflow.com using the "ibm-appid" tag. For support use the Support section in the IBM Cloud menu.

## Useful links

- Create new App ID instance in the IBM Cloud Services Catalog
  https://cloud.ibm.com/catalog/services/app-id

- App ID Documentation
  https://cloud.ibm.com/docs/appid/about.html

- App ID Key Concepts
  https://cloud.ibm.com/docs/appid?topic=appid-key-concepts

- Discover App ID – Blogs, Tutorials, Samples
  https://cloud.ibm.com/docs/services/appid/relatedlinks.html

- App ID SDK for iOS
  https://github.com/ibm-cloud-security/appid-clientsdk-swift

- App ID SDK for Android
  https://github.com/ibm-cloud-security/appid-clientsdk-android

- App ID SDK for Node.js
  https://github.com/ibm-cloud-security/appid-serversdk-nodejs

- App ID Terraform provider
  https://registry.terraform.io/providers/IBM-Cloud/ibm/latest/docs/resources/appid_action_url

## Videos and blogs

- Introducing IBM Cloud App ID
  https://www.youtube.com/watch?v=XlrCjHdK43Q

- IBM Cloud App ID Demo
  https://www.youtube.com/watch?v=HYomAFlNxqw

- Add sign-up and sign-in to your apps with IBM Cloud App ID
  https://www.youtube.com/watch?v=cDSYNFn4rX8

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. This document is distributed “as is” without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity. IBM products and services are warranted per the terms and conditions of the agreements under which they are provided.
IBM products are manufactured from new parts or new and used parts.
In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply.” Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.

Performance data contained herein was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those
customers have used IBM products and the results they may have achieved. Actual performance, cost, savings, or other results in other operating environments may vary.
References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs, or services available in all countries in which IBM operates or does business.
Workshops, sessions, and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer’s responsibility to ensure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer’s business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements, or other publicly available sources. IBM has not tested those products about this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM’s products. IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose.
The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks, or other intellectual property right.