AuthOne

Authentication Service

v1.0a

October 2018

Protocol One

Who might be interested in this document

Data presented in this document is relevant to a wide range of video game market participants and the developers of websites, applications and platforms.

This document describes the technical and procedural aspects of the AuthOne component of the Protocol One ecosystem. Description of the entire Ecosystem can be found within the Protocol One Whitepaper document.

Introduction

AuthOne is an open source authentication and authorization service. We believe that the developers do not have to waste weeks and months of their time to build authentication and data storage services of their own. Creating a custom authentication service providing sufficient scalability, flexibility, ease of use and administration is a complex and highly expensive task.

Why use AuthOne?

We want to make integration of authorization via social networks, gaming platforms, native or mobile application as simple as possible.

AuthOne will be useful to you, if:

- You do not want or cannot implement your own authentication solution, taking care of resetting passwords, creating users and their accounts within applications, blocking, deleting, filtering spam, preventing bruteforcing of passwords and developing administration interfaces to handle it all.
- Your application or API requires OAuth or JWT authentication.
- You have a lot of applications and websites, and you need a Single Sign-On.
- You need a flexible frontend and the ability to customize the behavior and external forms for authorization and registration.
- You want to authorize users with one-time codes sent via SMS and e-mail instead of a password
- You need a service that meets the requirements of GDPR, COPPA, FZ-153 and other laws concerning the storage of personal data.
- You need proactive protection against DDoS attacks and automated blocking of IP addresses responsible for multiple failed authorization attempts.
- You want to block accounts or reset passwords if the user's credentials have been compromised on a public resource.

What standards does AuthOne support?

The “login using...” has become a widespread and commonly accepted authorization method all over the Internet. Users employ dozens of authorization services in order to log into all kinds of applications and websites. It was made possible by a number of open and detailed specifications and protocols that assist user identification and authorization.

AuthOne supports the following open protocols:

- **OAuth 2**. This is a standard authorization protocol, aiming to simplify the client development. It provides specific authorization mechanisms for web applications, desktop applications, mobile phones and consumer electronics. This specification and its extensions are being developed as part of the IETF OAuth working group. You use this standard whenever you log in using your Google account, and you are asked whether you agree to share your email address and the list of your contacts with this site.
- **OpenID Connect**. This is a simple level of authentication running above OAuth 2.0 protocol. This protocol allows clients to verify the end user’s identity via authentication performed by the authorization server, and to obtain basic profile information about the end user in a compatible and REST-like manner.
- **JSON Web Tokens (JWT)**. This is an RFC 7519 open standard defining a compact and autonomous way to securely transfer data between the parties as a JSON object. This information is digitally signed and therefore can be verified and trusted. JWT can be signed with a secret (using the HMAC algorithm) or a public / private key pair using RSA or ECDSA.
- **Security Assertion Markup Language (SAML)**. An open standard for exchanging authentication and authorization data between participants, providing simple pass-through authentication (Single Sign-On).

Additional information about specific identification methods supported by AuthOne can be found in the “Identification Methods” section.

We are considering the possibility of adding support for the Decentralized Identifiers (DIDs) technology. DID is a new standard for decentralized user identification and authentication.

Roadmap

At this AuthOne Uttu is under active development, preparing for a public launch. The roadmap for the next 6 iterations of AuthOne is outlined below.

Version	Date	Key issues
1.0a	November 2018	AuthOne specifications, implementation for major JWT and Oauth 2.0 providers, control panel implementation.
1.0a-0	January 2019	First version of the product, integration with Protocol One, universal authorization form, SSO, built-in threat protection methods.
1.0b	March 2019	Integration of all primary identification methods, integration

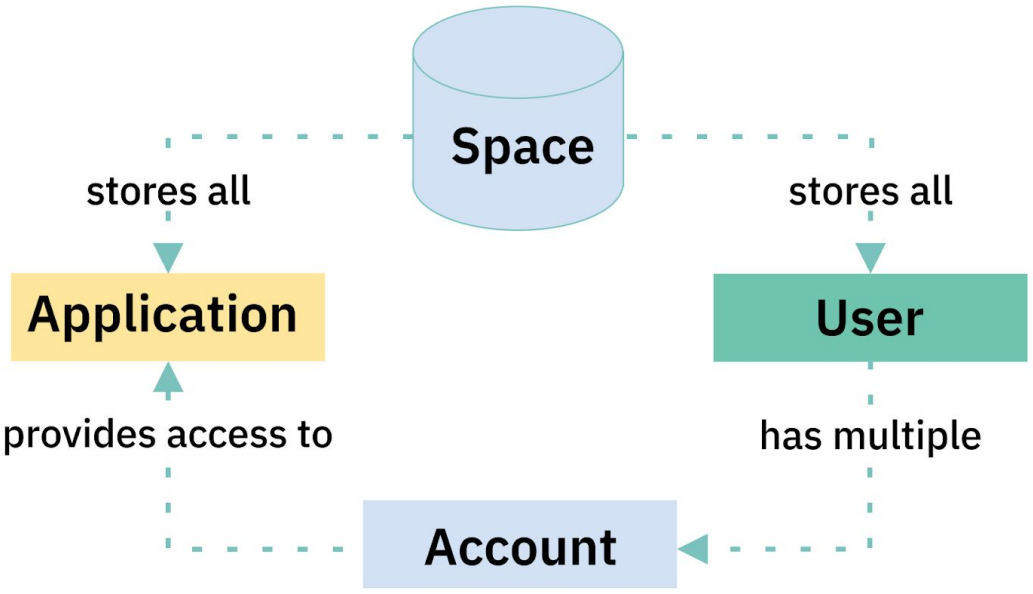
		with Uttu.
1.0b-0	May	Implementation of SSO plugins, scenario construction set, MFA.
1.0rc	July	SSO integration for game engines, preparation for release.
1.0r	August 2019	Release and the start of commercial exploitation.

Starting work

You can integrate any applications written in any language or stack with AuthOne, and choose any identification providers you wish. Depending on the technology your application uses, you only need to choose and integrate one of our SDKs and configure your AuthOne project.

Storing accounts

AuthOne provides a number of ways to store users and their account information. You can employ a simple scenario with a single application and its users, or a platform where a range of applications use a common user base, flexibly managing permissions and access rights. AuthOne allows to implement more complex scenarios, where a single user of a space and an application may have several accounts.



The space

Spaces are containers holding users, applications, settings and the data associated with them. The data within each space is completely isolated from others.

A space contains one or more applications using a divided or unified user base. Each space can approach user storage and authentication in two ways:

1. **Unified space.** All users belong to the space and can access different applications according to rights and permissions specified in the user account in an application.
2. **Divided space.** All users within the space belong only to their specified application. In this case, the user must register again in order to access different applications inside the same space.

The most common choice is a project featuring one space and one application using it, where all users belong to the space.

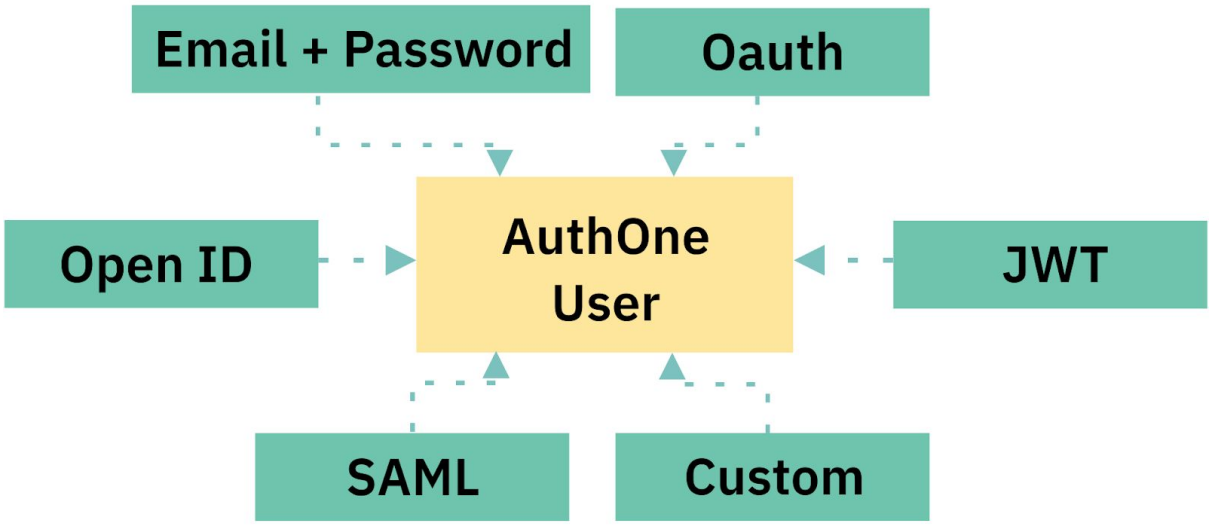
Divided space

This approach is typical for spaces where applications use different ways to authenticate users, or need to use various kinds of user identifiers in their business logic. It grants a number of advantages:

- The user of each application in a divided space is isolated and stored separately.
- It is possible to delete the users of such an application without affecting the work of other applications and the rest of the space.
- In applications within divided space, the user may work with different services without knowing that all of them are served by a single authentication service.

The user

AuthOne user stands for a container using a variety of user identification methods.



You can use the login and password authentication mechanism, or authorization via Facebook, Twitter, Twitch and other applications using the OAuth 2.0 protocol. You can also use authentication via JWT, SAML and Open ID. Or integrate an authentication / authorization protocol of your own.

A user can rely on a variety of authentication methods to log into applications. AuthOne does all linking and merging of accounts. Thus, when a user is already registered with login and password, and is trying to authenticate via Facebook or Steam, he will be offered to link the account already authorized with the new data.

Accounts

Accounts describe the user's data and access rights in a particular application. Generally speaking, each user has a single account.

In more complex cases, a user can have different accounts in different applications, flexibly managing his rights, contact information and permissions. This is often required when it is necessary to implement different authorization rules across different resources while using a single authentication mechanism. For example, requesting additional access rights to enter a reports portal or an administration panel — in this case, a single user will have several accounts with different rights for each application.

Sessions

The sessions functionality can be easily explained from the point of view of the HTTP protocol. A session is a period of time covering the user's Internet activity from the moment of opening the first link all the way to the last link visited. It is calculated as the time difference between the initial and the final request. Often, session assists in correct

identification of the user and calculation of time he has spent on the site. It can track periods of inactivity in order to request reauthorization.

The session functionality allows to uniquely identify the client and to store this client's additional data as temporary files on the server. This functionality helps to track requests sent by the same end user while viewing different pages or sending requests to a server that does not support continuous connection to the user.

- Every time a user is successfully authorized, we create a special JWT authorization token. This step opens the session for a space.
- When AuthOne is accessed, we prolong the session and update its data.
- When the user logs out, we invalidate the authorization token and close the session.

AuthOne supports several mechanisms for working with sessions: unmanaged and managed. In either case, the AuthOne user can browse his list of active sessions (an AuthOne-enabled end resource can get this data via API) and force any of them to close.

Unmanaged sessions

This type of sessions is suitable for interacting with all kinds of APIs and for developing SPA applications. Much like all other types of sessions, an unmanaged session is initiated after successful authorization. AuthOne creates a JWT authorization token and sends it to your application for verification. Unmanaged session is stateless and does not maintain any state on the server.

Authorization tokens of an unmanaged session serve as authorization tool for every request from the client to the server. Tokens are generated on the server from a secret key (stored on the server) and certain useful data. The token is then stored on the client side and used whenever it is necessary to authorize a request.

Moreover, you can change the lifetime of a token, thereby controlling the session duration. Tokens can be used without set lifetime, if needed.

In addition to the authorization token, AuthOne returns a refresh token. The latter is used to obtain a new authorization token after the old one expires. The use of refresh tokens allows to avoid requesting the user's authentication data over and over.

Managed sessions

Only living humans should use managed sessions. Like other types of sessions, this type of session opens after the user is authorized. Managed sessions are tracked by AuthOne separately. They additionally allow to:

- Get access to all active sessions by users, including IP-based location information, duration, starting time and the time of last activity.
- After logging out, the session is closed for all sources where it is used. For example, for all websites that used a shared user session.

- Session deletion is available not just to the user, but to the space administrator as well.

Identification methods

An identification provider is a server that provides identification data to other servers. Google and Facebook are examples of such providers, for example. If you log in to a website using your Facebook account, the Facebook server will send your identification data to the site.

AuthOne is placed between your application and the provider, and functions as an aggregator of authentication methods supporting great many providers using different protocols (OAuth, OpenID Connect, JWT, and others). Your application is isolated from any changes at the side of the end providers, and from the peculiarities of integrating their service.

Social networks and platforms

The following identification providers are maintained out of the box: Facebook, Box, Dropbox, Evernote, Github, Google, Instagram, Linkedin, Microsoft Accounts, PayPal, Saleforce, Shopify, Twitter, Vkontakte, Wordpress, Yahoo, Yandex, Steam, Twitch, Discord, FacebookInstantGames, Apple Game Center, Kongregate, Google Play, XBox Account, Nintendo Switch, PlayFab, Telegram, WhatsApp, WeChat.

You can also use the Custom OAuth Provider to integrate any other provider using OAuth 2.0.

Enabling any identification provider will only require you to set a few individual options for each. As a rule, this means pointing to identifiers and keys that you obtain on each provider’s website on your own.

Login and password

AuthOne's own infrastructure is used to store user data by default. This ensures superior performance for the authentication process. The AuthOne database is highly secure and compatible with GDPR. Passwords are hashed using bcrypt, they are never saved as plain text. Different levels of password security requirements may be applied.

We believe that a good password must meet the OWASP criteria:

- **Length.** Longer passwords using characters from a correct set are harder to compromise. We believe that passwords under 10 characters long are weak.
- **Complexity.** We recommend using passwords made of lower and upper case letters, numbers and special characters.
- **Passphrases.** A combination of words is often much longer than a regular password, and far easier to remember.

Each AuthOne space is able to independently manage the password creation criteria, setting different complexity levels or specifying a template that regulates the password length, the character set and the need to use passphrases.

When you change the default password complexity, you can enable forced password update. In this case, the authentication form will request the users to update their passwords after a successful login.

External database

If you have an existing user repository or you want to store user account data on your own server, AuthOne allows you to connect to a database or a repository in order to use it as an identification provider. In this case, you have to specify the connectivity data for your external source and associate the internal structure of your repository with AuthOne.

We support any ODBC-compatible databases (Microsoft SQL, MySQL, PostgreSQL, Oracle) and MongoDB.

You can set up a seamless migration of your user base to AuthOne.

Password-free authentication

This form of authentication allows users to log in without having to remember a password. This makes it easier for the user to interact with your application, especially for native or mobile applications. The users only need to remember their e-mail or phone number.

This method releases you from having to implement the password reset and recovery logic, and the entire infrastructure will be less prone to the insecure practice of using the same password across a range of resources.

The method allows to send out SMS and e-mail notifications using the AuthOne infrastructure or external providers.

Multi-factor authentication

Multi-factor authentication (MFA) is an advanced method of authentication which requests users to present more than one “proof of authentication” to gain access to information. This approach adds another layer of security, making it significantly harder to gain unauthorized access to authorization data. Possible proofs include:

- **Knowledge:** information known to the subject. For example, password or PIN.
- **Possession:** an item owned by the subject. For example, an electronic or magnetic card, a token, flash memory, a smartphone with code-generating application (such as Google Authenticator).
- **Inherent** attributes associated with the subject. For example, biometrics and inherent unique differences: face, fingerprints, iris, capillary patterns, DNA sequence.

AuthOne supports the following MFA types:

- Google Authenticator or Authy one-time passwords.
- One-time passwords sent over SMS.
- One-time passwords sent to secondary e-mail.
- One-time passwords sent via Push notifications (requires the AuthOne application).

Scenario Construction Set

AuthOne features a visual tool for managing the rules of authentication and multi-factor authorization. Among other abilities, this tool can be used for:

- Enabling context-dependent multi-factor authorization. For example, depending on the location of previous authorization, or IP address.
- Realtime notifications of other systems using Webhooks.
- Creating complex authorization scenarios. For example, those requiring the input of a phone number when registering from particular regions, or preventing frequent registration calls from the same IP address.
- Implementing the user whitelist or asking for forced authentication during a valid session.

The scenario construction set supports webhooks to send event notifications to external services. All actions that occur in AuthOne produce events which can be used by the scenario construction set or forwarded to external services.

Analytics and marketing

Analytic tools help to track users on a site or inside an application. The integration of analytics into Auth0 allows you to evaluate user behavior and use this data to segment users, optimize user retention, improve the registration process and lead generation.

AuthOne supports the Uttu, Facebook and Google analytics. If using Uttu, the data on user behavior during registration and authorization is automatically synchronized with Data Hub — the central hub of Qilin and the Protocol One DRM ecosystem, acting as an independent intermediary for the creation and storing of data common to all ecosystem participants.

Security

Security is a key objective and purpose of AuthOne. We make every effort to store, transfer and process data in a most secure manner. The Universal Login and AuthOne API are developed and tested in order to avoid critical threats and the ways to compromise authentication.

Common threats

Many threats and vulnerabilities are in some way connected to the operation and shortcomings of the HTTP protocol. This section lists common threats that have been addressed in AuthOne.

MITM

AuthOne uses encryption to access its resources and SSL-based API for the HTTP protocol. The use of HTTPS provides protection against attacks based on listening to network connection, such as sniffer and man-in-the-middle attacks. AuthOne uses HSTS (abbr. from eng. HTTP Strict Transport Security) — a mechanism to force a secure connection through the HTTPS protocol. This security policy allows to establish a secure connection immediately rather than using the HTTP protocol. HSTS helps to prevent some of the attacks aiming to intercept the connection between the user and the website, particularly downgrade attacks and cookie hijacking.

Replay attack

Replay attack is an attack on the authentication system by the means of saving and then replaying a previously transmitted valid message, or portions of it. Shortcomings of data flow authentication methods are exploited to carry out this type of attack. A typical illustration of such an attack is stealing and reusing cookies in order to perform actions without a proper authentication. For example, the session data kept in local storage, or using XSS attacks.

AuthOne helps preventing such attacks by using one-time passwords and authorization tokens. Each token contains additional information that allows to verify access based on IP and browser fingerprints. In addition, all AuthOne JWT tokens have a limited lifespan in order to avoid an infinite number of markers, and provide means to blacklist discredited tokens. To counter such attacks, AuthOne uses `jti` and `aud`.

XSRF and CSRF

CSRF (Cross Site Request Forgery) is a type of attack targeting website visitors that exploits the flaws of the HTTP protocol. When the victim enters a malicious site, it secretly sends a request to another server (for example, the payment system server), performing some kind of malicious operation (such as transferring money to the attacker's account). For the attack to succeed, the victim must be authenticated on the server where the request is sent, and the request must not require any user confirmation that could not be ignored or forged by the attacking script.

To avoid this threat, AuthOne uses special tokens in accordance with OWASP recommendations.

Personal data

AuthOne makes every effort to ensure the security and privacy of all data collected. Personal data of users from the European Union (EU) is handled according to the provisions of GDPR. In the near future we intend to receive the Privacy Shield and SOC 2 certifications. AuthOne meets the basic requirements of GDPR, as listed below.

Consent to data processing

In accordance with articles 6-8 of the regulation, it is required to:

- request for user's explicit consent in an accessible form to processing of his personal data.
- allow the user to revoke this consent or to prohibit processing of his personal data.

AuthOne uses a registration form to obtain consent for processing of personal data. If the user has changed his decision and refuses to permit processing of his personal data any further, he can cancel it by filling out a form on the AuthOne website.

- may request and receive a copy of his personal data, request to delete or correct them.

User rights

In accordance with articles 15-17 and 19 of the regulation, the user has the following rights:

- The user is entitled to receive a copy of his personal data.
- To request to change incorrect personal data.
- To request to delete personal data.

If user data is stored in the AuthOne repository, the user can use the request form in order to cancel processing, change or delete his data. If user data is stored on your side, you can manage it on your own.

Data processing and storage

In accordance with article 5 of the regulation, it is required to store and process personal data as follows:

- to only collect the data that has to be processed in order to achieve the purpose.
- to ensure the security of data in order to avoid its damage, destruction or illegal copying.

AuthOne utilize user data that may pertain to the sending of e-mails, blocking of the user, payment processing and other applications of personal data. Personal data of the users is storage and transmitted in encrypted form, in accordance with the AES-256 algorithm.

Downloading data

In accordance with article 20 of the regulation, the user has the right to transfer and store his personal data. In order to comply with this, it is required to provide the user with his data in a machine-readable electronic format.

The user can always use the form to receive his data in JSON format.

Data protection

In accordance with article 32 of the regulation, it is required to provide reliable level of data security, utilizing but not limited to:

- data encryption.
- data confidentiality.
- data integrity.
- failproof data processing systems.

To meet the requirements, AuthOne utilizes data encryption, brute-force protection, password validation, MFA, and restriction of access to support data.

Attack detector

AuthOne implements a number of built-in mechanisms to protect accounts from unauthorized access. These mechanisms will complicate the authentication process or block the attacker's IP address. The attack detector is implemented on the basis of the scenario construction set. You can change the default scheme by adding notifications and new conditions.

Brute-force protection

AuthOne utilizes several strategies to identify a large amount of invalid authentications:

1. If failed login attempts to more than 10 different accounts were registered within 1 hour from the same IP, the IP address will be blocked for 24 hours.
2. If a specific user has failed 5 login attempts within 24 hours, a captcha will be used on authorization forms and the IP address will be blocked for 24 hours.
3. If a failed login attempt was registered from the IP with unusual geolocation (country, region, city, provider), a captcha will be used.

Whenever brute-force protection is triggered, the user will be alerted via e-mail. All protection options are enabled by default. It is possible to configure or turn them off for each application or space.

Detection of compromised passwords

Often, users use the same passwords across a variety of resources. The Protocol One team monitors security breaches on third-party sites, along with the information posted on sites belonging to the hacker community. If compromised user passwords have been detected, we will alert the affected user. You can enable forced blocking of such a user and the forced initiation of the password change procedure, if necessary.

AuthOne SSO (Single Sign-On)

Single Sign-On is a technology allowing the user to move from one portal section to another without requiring re-authentication. For example, when there are several large independent

sections on a web portal (a forum, a chat, a blog and so on), the user, having passed authentication for one of these services, gains access to the rest, saving him from repeated account data entry.

Single Sign-On usually utilizes the Central Service, which handles the user's pass-through authentication across multiple clients. Common algorithm looks this way:

1. The user visits site A.
2. Gets redirected to the Central Service, which creates the authorization cookie.
3. The user then visits site B.
4. Gets redirected to the Central Service once more to verify the cookie.
5. The Central Service verifies the cookie. If the authorization cookie is valid, the user is redirected to site B with his user data.

For AuthOne, the AuthOne authorization server is used as the Central Service. SSO can only function when the universal authentication form is used. The algorithm of the AuthOne authorization server functions in the same way as outlined above:

1. The user of a website or an application is redirected to the login page.
2. AuthOne checks for a valid SSO cookie.
3. If the SSO authorization cookie does not exist, the user needs to authorize using one of the available methods.
4. After authentication, a new cookie is created and saved.
5. The authorization server performs a redirect to the original page, passing a token containing the user's identification information.

In case of a second visit, the authorization server will update the cookie file and immediately redirect to the original page, passing the token with the identification information.

Available integrations

AuthOne allows to integrate joint authorization with Zendesk, Disqus, SalesForce, Slack, DescPro, PhpBB, Vanilla, XenForo, vBulletin using ready plugins for all these solutions. Wherever possible, AuthOne uses SSO based on JWT and SAML as a backup solution.

Unity and Unreal integration

You can integrate AuthOne Single Sign-On with games based on Unity and Unreal Engine. To do this, you need to install plugins for these engines. PC games based on these solutions require running the default browser to complete Single Sign-On. Games for mobile platforms require the special AuthOne application.

Universal login

The universal login is the AuthOne built-in authentication and authorization page. The appearance of the authentication form can be configured flexibly for each space and application. By default, the universal form is used to login to all AuthOne resources. Using

the AuthOne API, you can completely replace the universal form with a solution of your own, if needed.

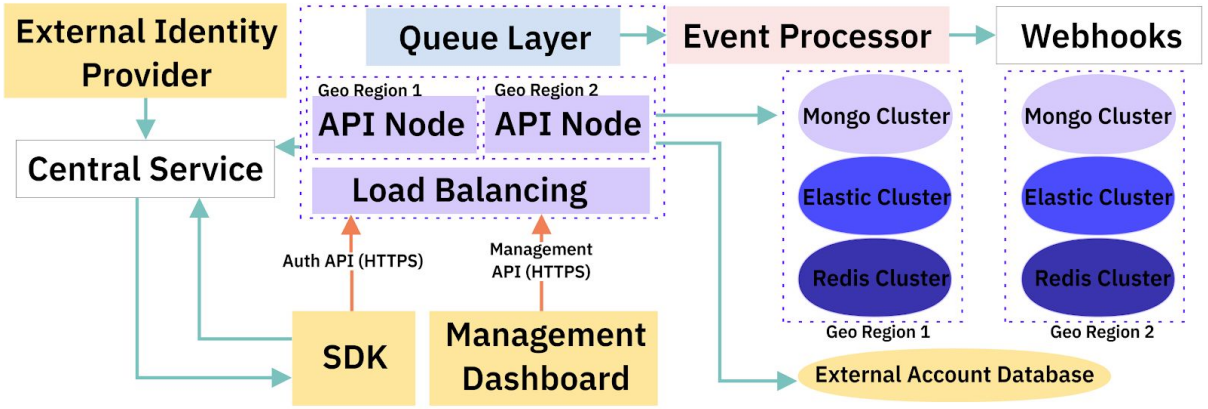
AuthOne opens the universal login form whenever the user initiates authentication or performs identification via SSO. In case an external identification provider is utilized, the universal form will not be shown. Instead, AuthOne will redirect the user to the corresponding page of the identification provider.

The main reason to use the universal login is the ease of installation and security. Often, the universal login is the best (and sometimes the only) way to implement Single Sign-On and password-free authentication on native platforms (such as iOS and Android).

The universal login is available as:

- JavaScript SDK to integrate web page authentication and electron applications.
- Native SDK (C++, C#) to integrate authentication into desktop applications.
- Android and iOS SDK for integration into mobile applications.
- Unity and Unreal packages for integration into games utilizing these engines.

Architecture



Technology stack

- DNS routing based on geographic location.
- Nginx-based HTTPS
- Data enrichment API servers based on auto-scalable group in kubernetes
- Go and Node.js as main languages.

Protocol One or own installation

AuthOne is an open solution based on open source code. However, Protocol One utilizes an AuthOne installation of its own, offering a number of advantages and differences from the open solution. The table below describes the operational and functional differences between these two models.

Feature	Protocol One	Open Source
Infrastructure with guaranteed SLA and support	Yes	No
Infrastructure meeting GDPR requirements	Yes	No*
SSO availability	Yes	Yes**
SSL support	Yes	Yes**
MFA	Yes	Yes***
Search for users	Yes	Yes****
Detection of compromised passwords	Yes	No
Support	Yes	No
GDPR, Privacy Shield and SOC 2 certifications.	Yes	No

* Unaided installation of Uttu relies on the owner to meet the requirements of the laws concerning storage of personal data on his own. As for us, we provide the best practices and any help necessary to create such an infrastructure.
** You will need to configure Central Service and receive and configure TLS certificates on your own.
*** You will need to configure the Google Authenticator integration, pay and maintain the sending of SMS and e-mail alerts on your own. Push notifications are not available.
**** AuthOne utilizes ElasticSearch to search for users according to arbitrary attributes. Unaided installation of AuthOne may require additional configuration of ElasticSearch.

Using unaided installation of AuthOne is and will remain free. Listed below are examples of upcoming Auth One SaaS fees as a part of the Protocol One platform (the fees will be described in detail in WP 1.0rc in 2019):

- Free as long as the amount of stored users does not exceed 10,000 people and the number of connected identification providers does not exceed 10.
- $10 for every 30,000 stored users per month;
- Paid installation and consulting for unaided installations of AuthOne.

Suggested hardware

Choosing optimal hardware for unaided installation is no easy task, it depends on a large number of factors: the volume of incoming and stored data, the amount and complexity of integrations and the SLA requirements.

We give the following recommendations for unaided installation of AuthOne on your own hardware, provided that the number of your users does not exceed 100,000 users:

- The fault tolerance policy suggests that you run at least 3 servers for each cluster group (Kubernetes, Mongo, Redis, ElasticSearch, Kafka).
- Recommended server configuration:
    - Redis, Mongo, Kafka, ElasticSearch:
        - 1 x Intel Xeon E5-2630 v4
        - 6 GB RAM
        - SSD 32 GB
    - Kubernetes:
        - Virtual machines with 8 GB RAM and 8 VCPU

Disclaimer

This document does not represent a solicitation of investment or any other form of material support for AuthOne or any component of the Protocol One project. The purpose of this document is to provide a detailed and comprehensive description of AuthOne.

Statements and other information of a declarative nature contained within this document must not be construed as direct assertions or promises unless they are expressly specified as such.

In the current edition, we actively collect feedback from analysts, marketers, video game developers and publishers, stores and platforms to improve the specifications and make the product we create easier and more convenient for all participants.