Authentication

Apstrata Authentication Methods

In Apstrata, all requests must be authenticated. There are four methods for authenticating Apstrata requests:

1. Default Signature: This is the most secure method of authentication because it requires hashing all content of a request along with the secret of the account or the password of the user or the device and then sending the hash. (read more)
2. Simple Signature: This is the easiest method of authentication. It requires hashing a few select parameters along with the secret of the account or the password of the user or the device and then sending the hash. It is recommended for testing and for applications that do not have access to all parameters, *e.g.*, files, in a request. (read more)
3. Token-Based Authentication: This is the recommended method of authentication for applications that make most requests with Apstrata users and devices, as opposed to owners, for use with SSL encrypted connections over HTTP POST. It provides a similar experience to sessions since a Token is generated and renewed over a period of time, without the need to generate a signature for every request. (read more)
4. Bearer Token Authentication: This authentication allows the users and devices to issue a request using a bearer token in the header. In order to issue a request with a token bearer header, you first need to generate a token for a user or a device. Users and devices make authenticated requests with a bearer token using the Authorization request header field. (read more)

The authentication roadmap for the Apstrata database includes optional replay prevention.

Authentication Methods Description

Apstrata services only accept authenticated requests. Authenticating an Apstrata service request means either sending it with the apsws.authSig parameter or with a valid value for the apsdb.token parameter.

- Creating the signature parameter for owner requests requires the developer to know their authentication key (apsws.authKey) and their authentication secret.
- Creating the signature parameter for user requests requires the developer to know their authentication key (apsws.authKey), the user login, and the user password.
- Generating tokens requires generating a user request to VerifyCredentials or generateToken in order to generate a token and then sending the token with every consecutive request in order to authenticate it. The token needs to be constantly renewed and requires re-authentication when its lifetime expires. The authentication key and the secret are provided upon registration. The user login and the password are chosen when creating an Apstrata user. Note that the secret works as a password for accessing our services suite and hence your data.
- Using a bearer token authentication enables users and devices to access protected resources without sending their credentials or tokens as parameters. Instead, users and devices can set in the Authorization header of any Apstrata HTTP request a bearer token that will be computed based on the application authentication key, their unique identifier and their Apstrata token.

Never reveal your authentication secret to a third party, an Apstrata database affiliate will never ask for your secret. In case you feel that your authentication secret has been compromised, you can generate a new one.

Two types of signatures exist, one which is more secure and complex to construct, and another version which is simpler to construct but less secure.