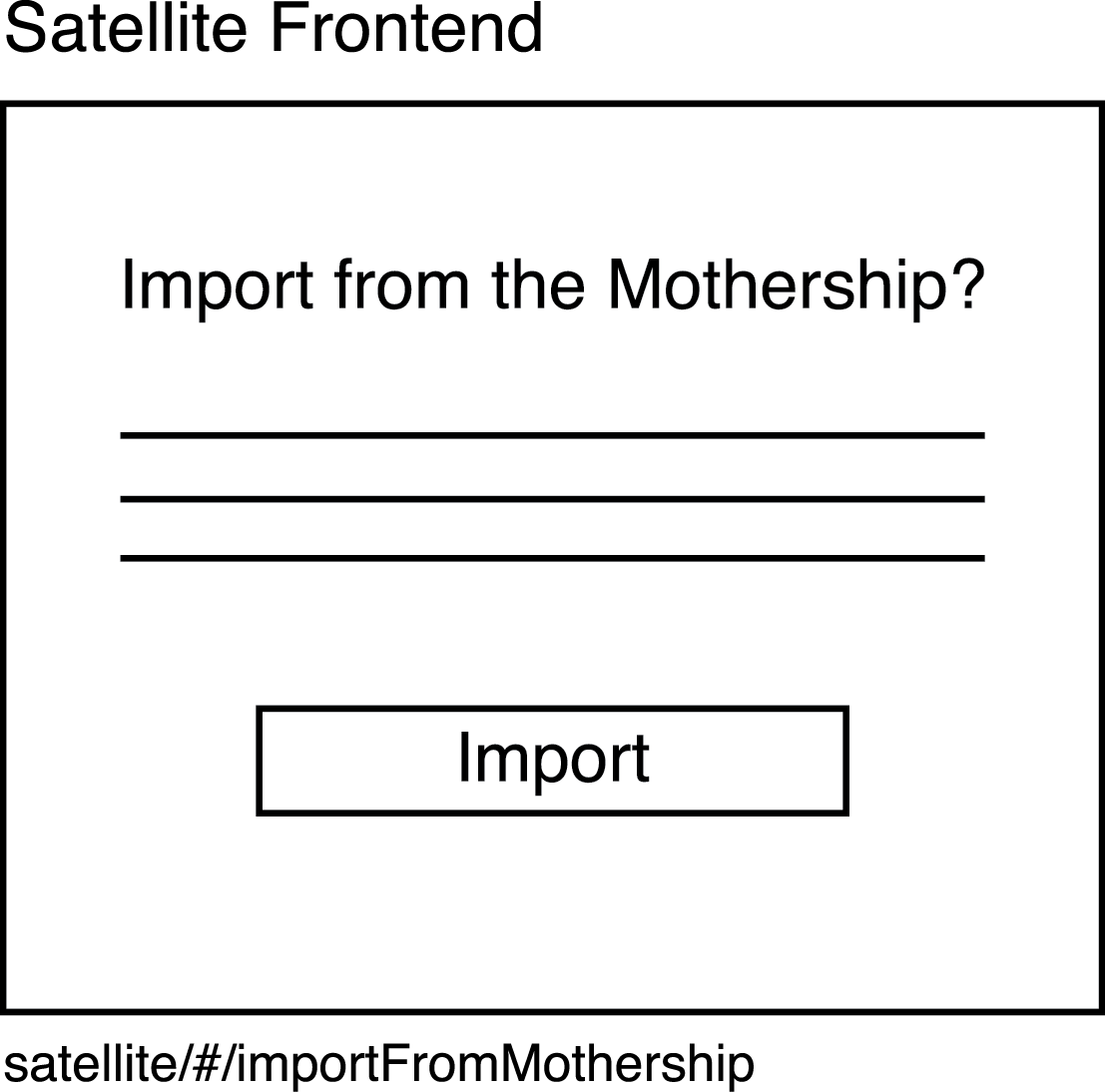
**Anatomy of a Cross-service Import**

Sometimes when building a system, it makes sense to import a user’s data from another service. In well-designed online systems, both services run secure authentication schemes. However, rarely do services use compatible forms of user authentication. As a result, it is complicated to securely exchange data between online services.

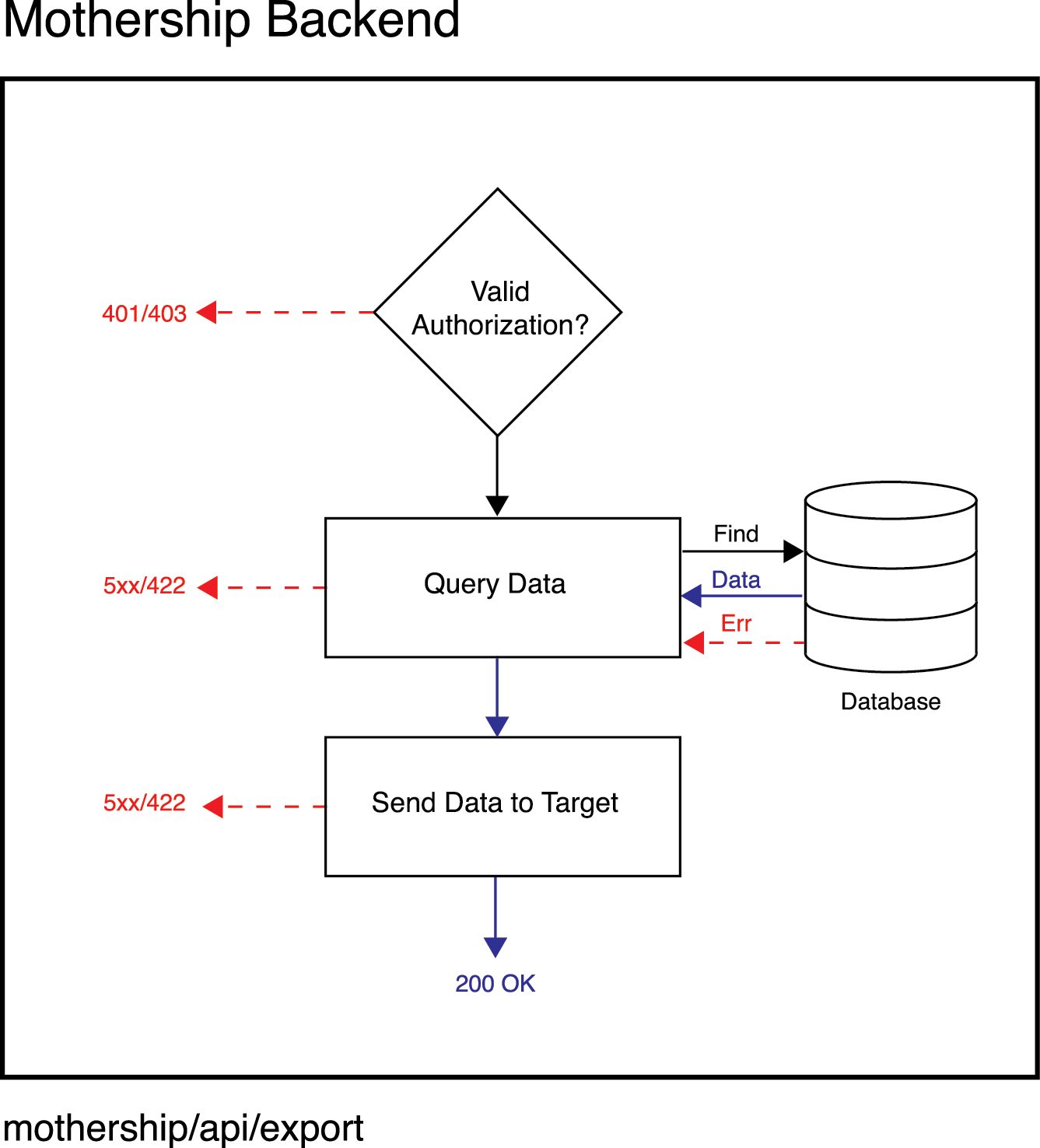
In this article, I present an architecture for performing this exchange without leaking user information at any point to an unauthenticated user.



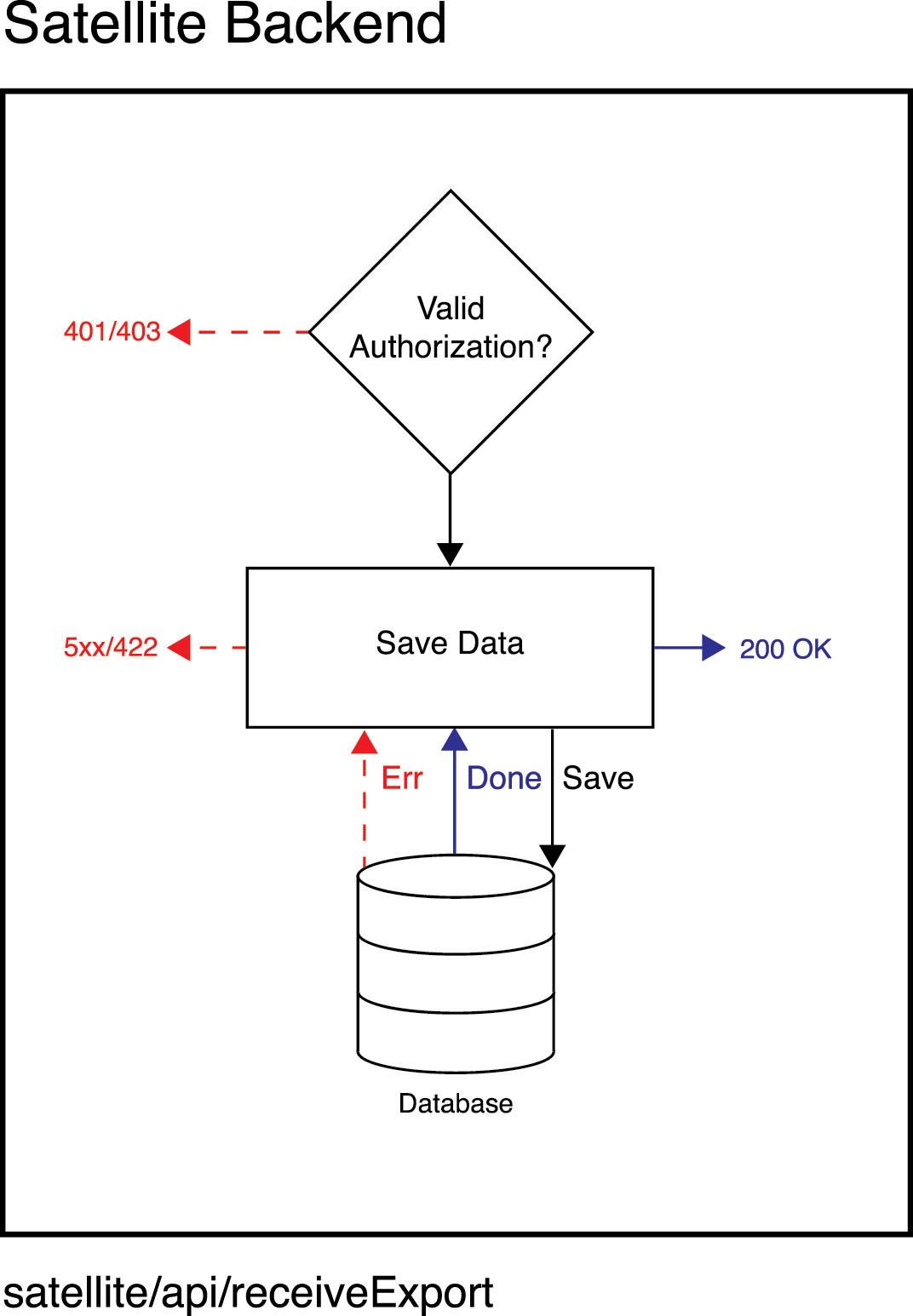
In order to initiate an import of data from a “mothership” service to a “satellite” service, one must first authenticate as a user of the satellite service. By doing this, one acquires an identity on that service in the form of a user id.



The satellite service then directs the user to a specialized page on the mothership service designed to accept requests for information in the query string. If one is not already authenticated with the mothership service, the mothership service must authenticate the user in its system. The mothership service may then choose to provide its own user with additional information to validate that it wishes to export to the identity on the foreign server. For instance, the mothership frontend could reach out to the mothership backend, which in turn reaches out to the satellite backend in order to acquire additional information about the user. Additional user data from the query string cannot necessarily be trusted.



Once the mothership user has approved the export of its sensitive information to the chosen identity on the satellite service, a call is made to the mothership backend containing an authentication token for mothership service and the target identity on the satellite service. The mothership backend then makes a call to the satellite backend with the an interserver token, target satellite identity, and requested data.



When the satellite backend receives a request from the mothership service containing information, it validates the interserver token using a shared secret. Then, it saves the data associated with the target identity in its system. It then returns an OK response and begins a cascade of responses that eventually reaches the user. The user can now access the imported data using his or her identity on the satellite server.

