Spring Security Architecture Overview

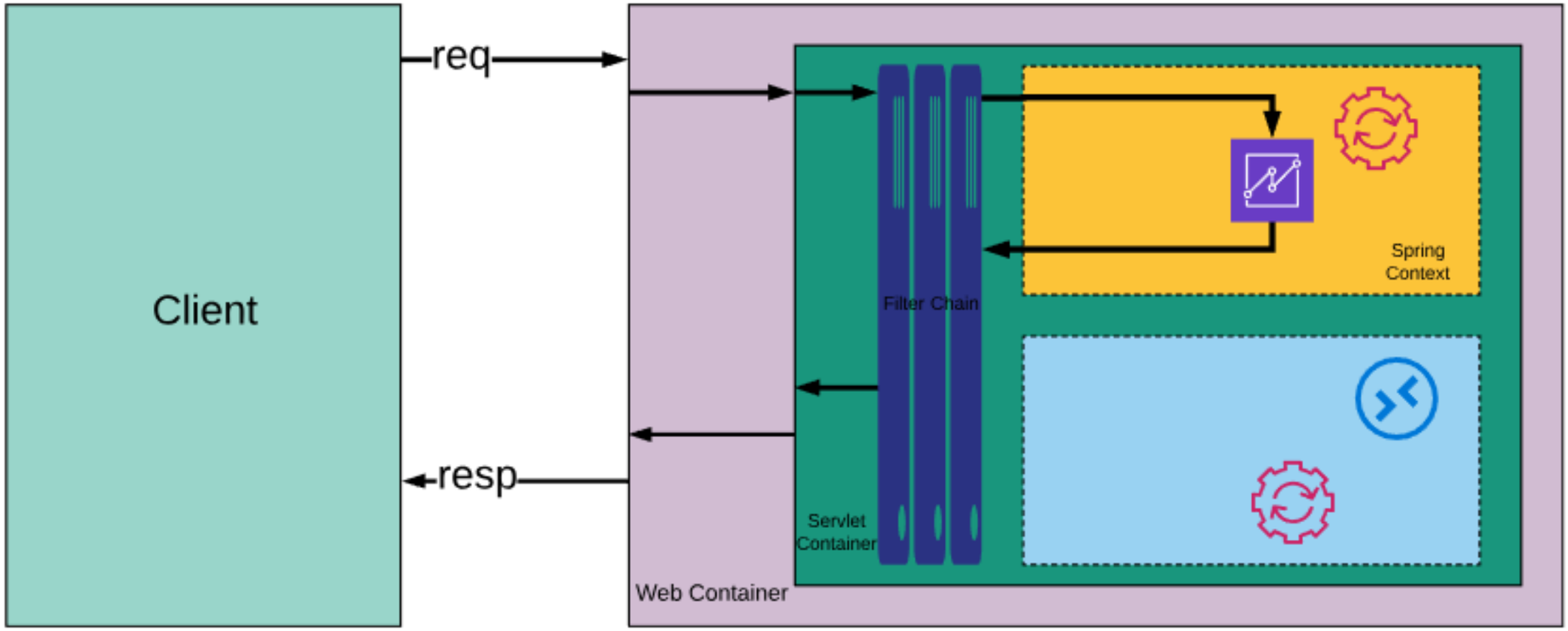
https://github.com/arunveersingh/spring-security

Basics

- web.xml / deployment descriptor
- Web Server
- Servlet Container
- Java class loading

- executable jar

- Spring Context
- Security Context

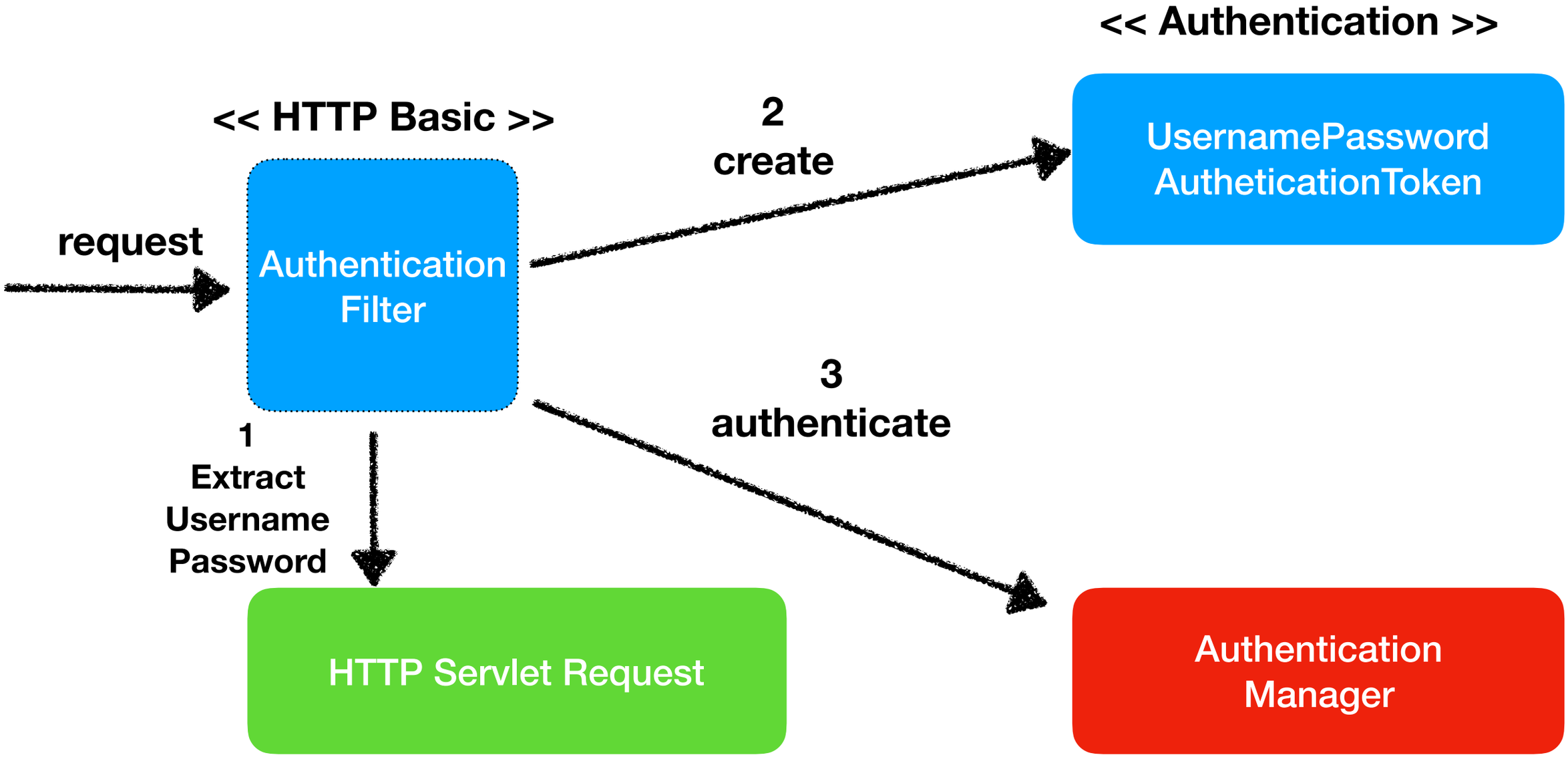

Client
req
resp
Filter Chain
Spring
Context
Servlet
Container
Web Container

Authentication

Authorization / Access Control

Exception Handling

Authentication Filter



Authentication

AUTHENTICATION	
Principal	emailid@provider.com
Credentials	password
Authorities	
Authenticated	FALSE

AUTHENTICATION	
Principal	emailid@provider.com
Credentials	password
Authorities	
Authenticated	FALSE

AuthenticationToken

authenticate

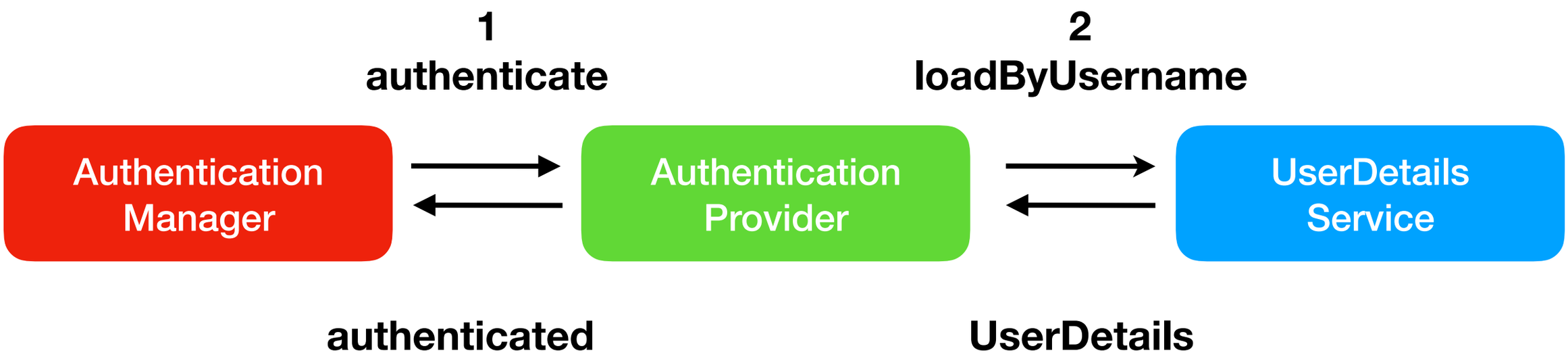
Authentication Manager

<< Authenticated >>

UserNamePassword AuthenticationToken

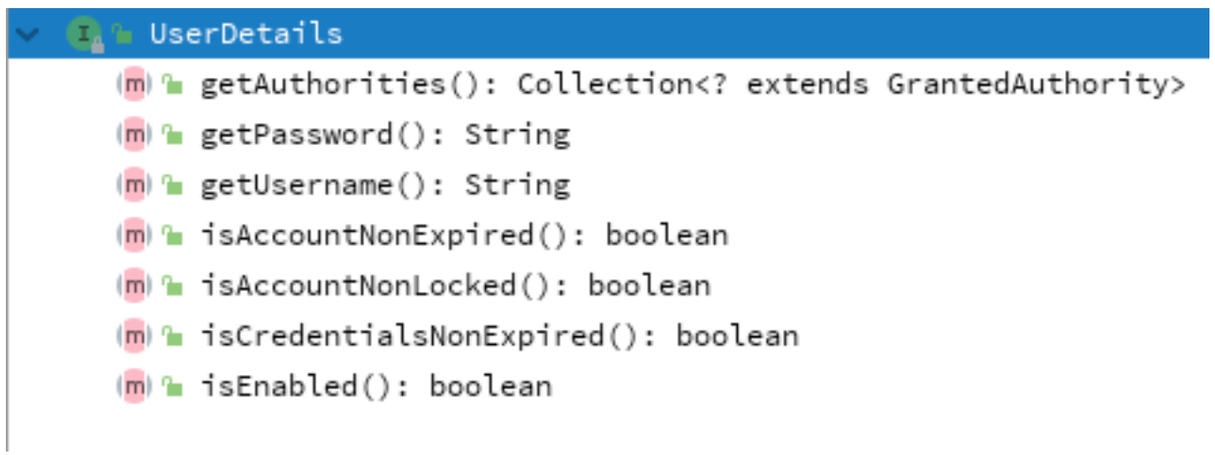
AUTHENTICATION	
Principal	emailid@provider.com
Credentials	
Authorities	ROLE_USER
Authenticated	TRUE

```java
public interface Authentication extends Principal, Serializable {
    Collection<? extends GrantedAuthority> getAuthorities();

    Object getCredentials();

    Object getDetails();

    Object getPrincipal();

    boolean isAuthenticated();

    void setAuthenticated(boolean var1) throws IllegalArgumentException;
}
```

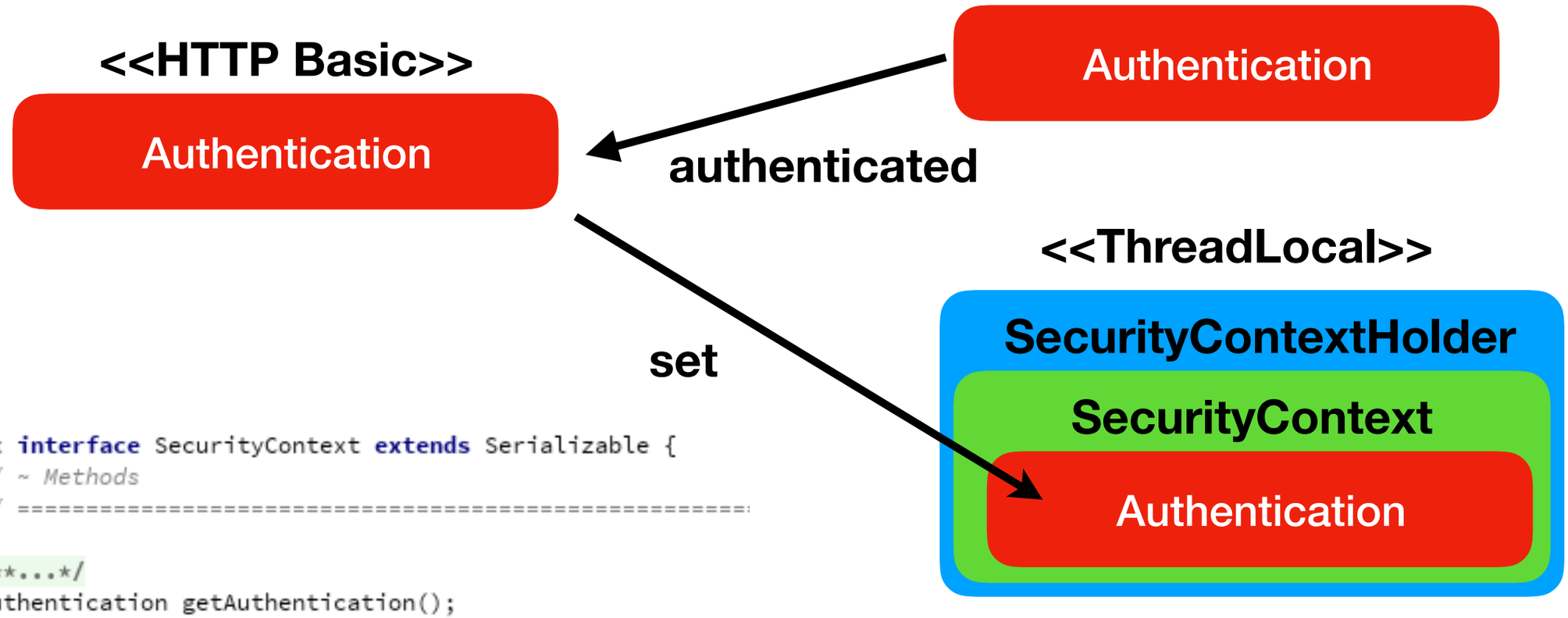


AUTHENTICATION	
Principal	emailid@provider.com
Credentials	
Authorities	ROLE_USER
Authenticated	TRUE

```java
public interface UserDetailsService {
    // ~ Methods
    // ========================================================================================

    /**...*/
    UserDetails loadUserByUsername(String username) throws UsernameNotFoundException;
}
```



SecurityContext



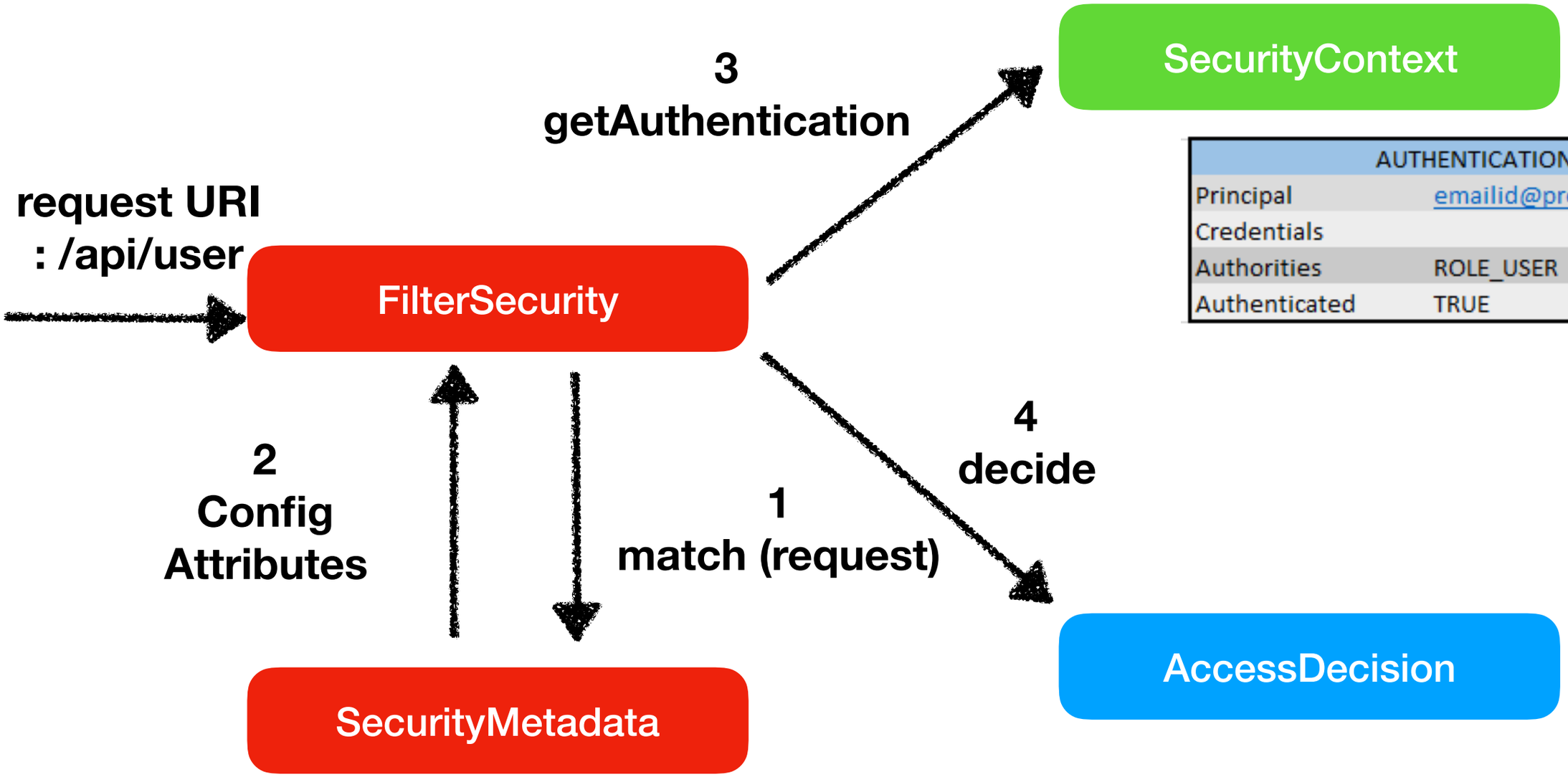

```
public interface SecurityContext extends Serializable {
    // ~ Methods
    // ========================================================

    /**...*/
    Authentication getAuthentication();

    /**...*/
    void setAuthentication(Authentication authentication);
}
```

SecurityContextHolder

```
    /**...*/
    public static void setContext(SecurityContext context) { strategy.setContext(context); }
```

```
private static void initialize() {
    if (!StringUtils.hasText(strategyName)) {
        // Set default
        strategyName = MODE_THREADLOCAL;
    }

    if (strategyName.equals(MODE_THREADLOCAL)) {
        strategy = new ThreadLocalSecurityContextHolderStrategy();
    }
```

Authorization



AUTHENTICATION	
Principal	emailid@provider.com
Credentials	
Authorities	ROLE_USER
Authenticated	TRUE

SecurityMetadata	
Request Pattern	/api/user
Config Attributes	ROLE_USER

Access Decision

AUTHENTICATION	
Principal	emailid@provider.com
Credentials	
Authorities	ROLE_USER
Authenticated	TRUE

SecurityMetadata	
Request Pattern	/api/user
Config Attributes	ROLE_USER

request URI
: /api/user

HTTP Servlet Request

Authentication

Security Metadata

decide

AccessDecision Manager

granted

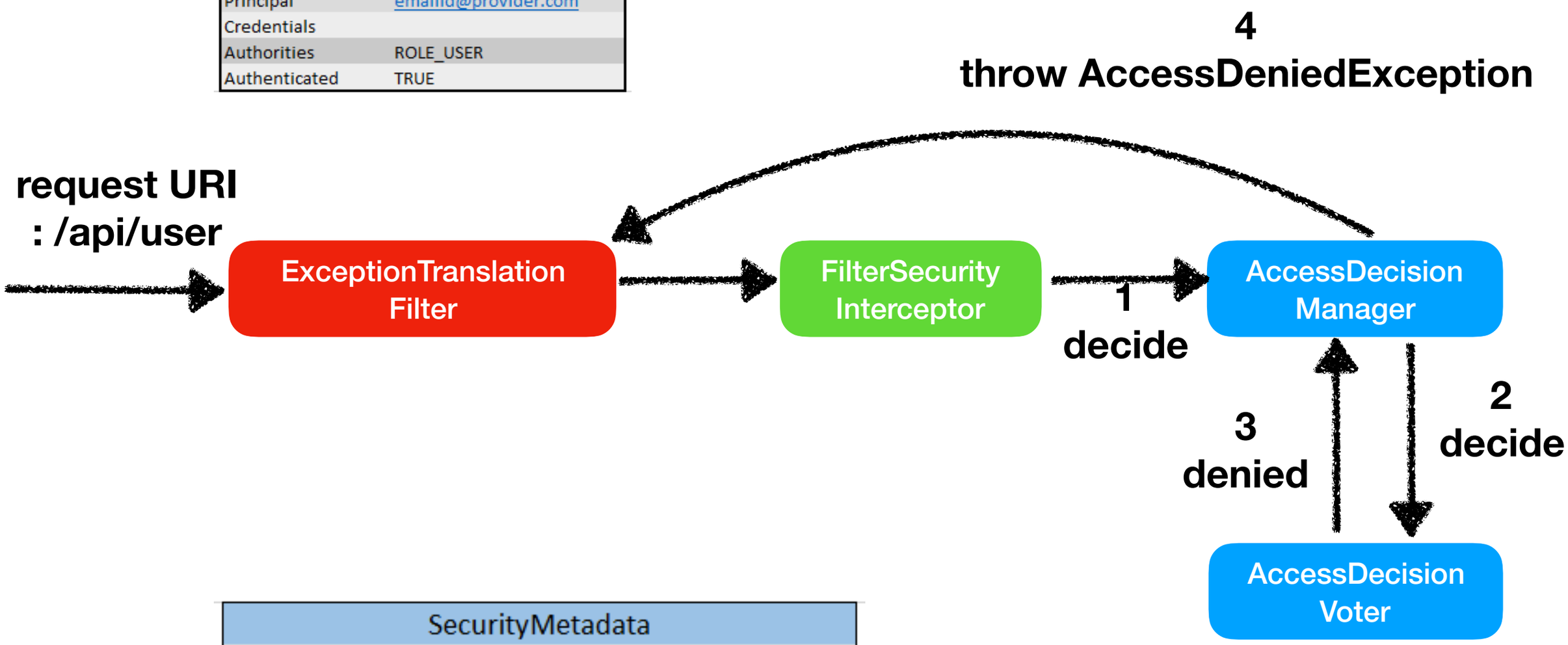
vote

Access Decision Voter

<<Role Voter>>

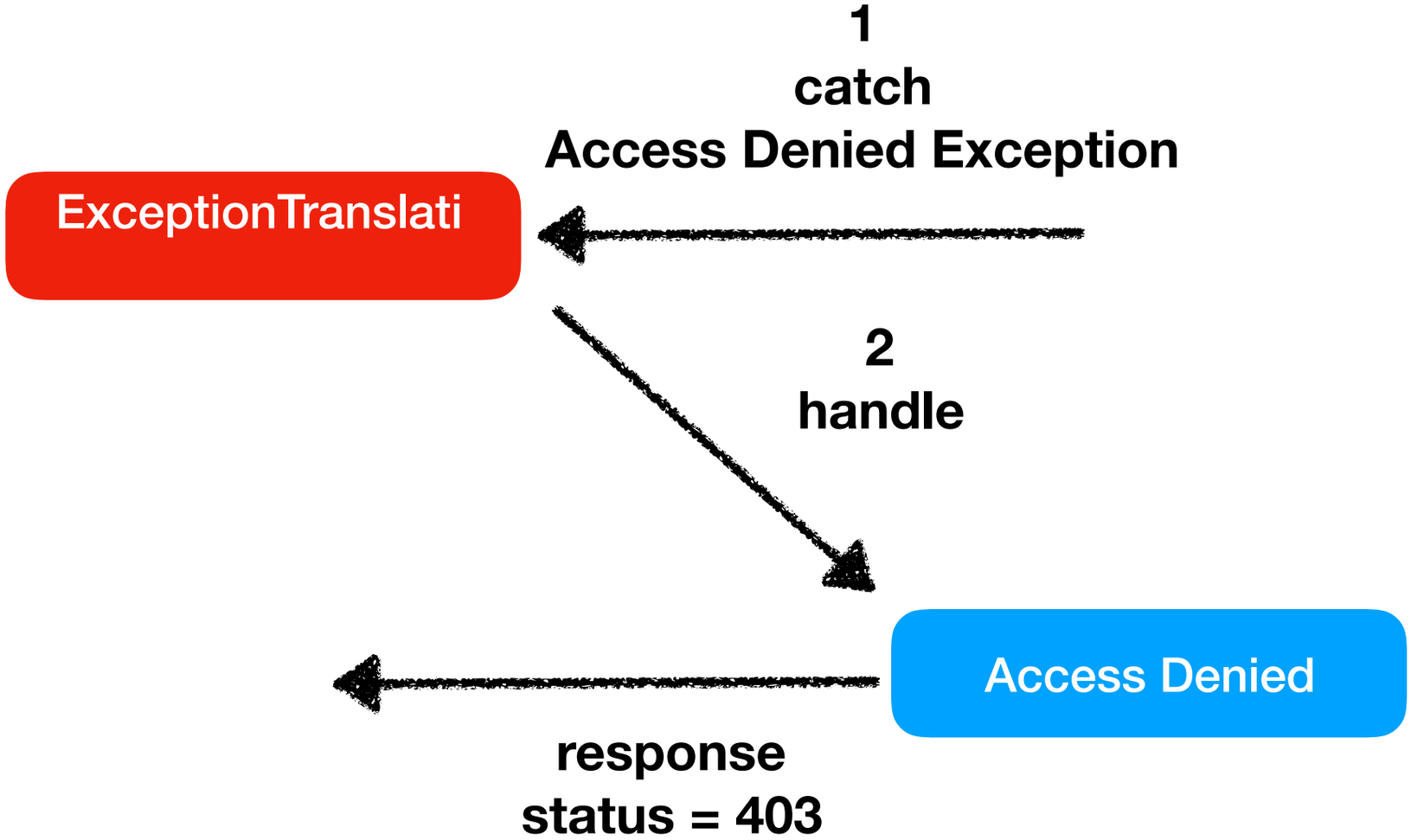
Access Denied

AUTHENTICATION	
Principal	emailid@provider.com
Credentials	
Authorities	ROLE_USER
Authenticated	TRUE



SecurityMetadata	
Request Pattern	/api/user
Config Attributes	ROLE_ADMIN

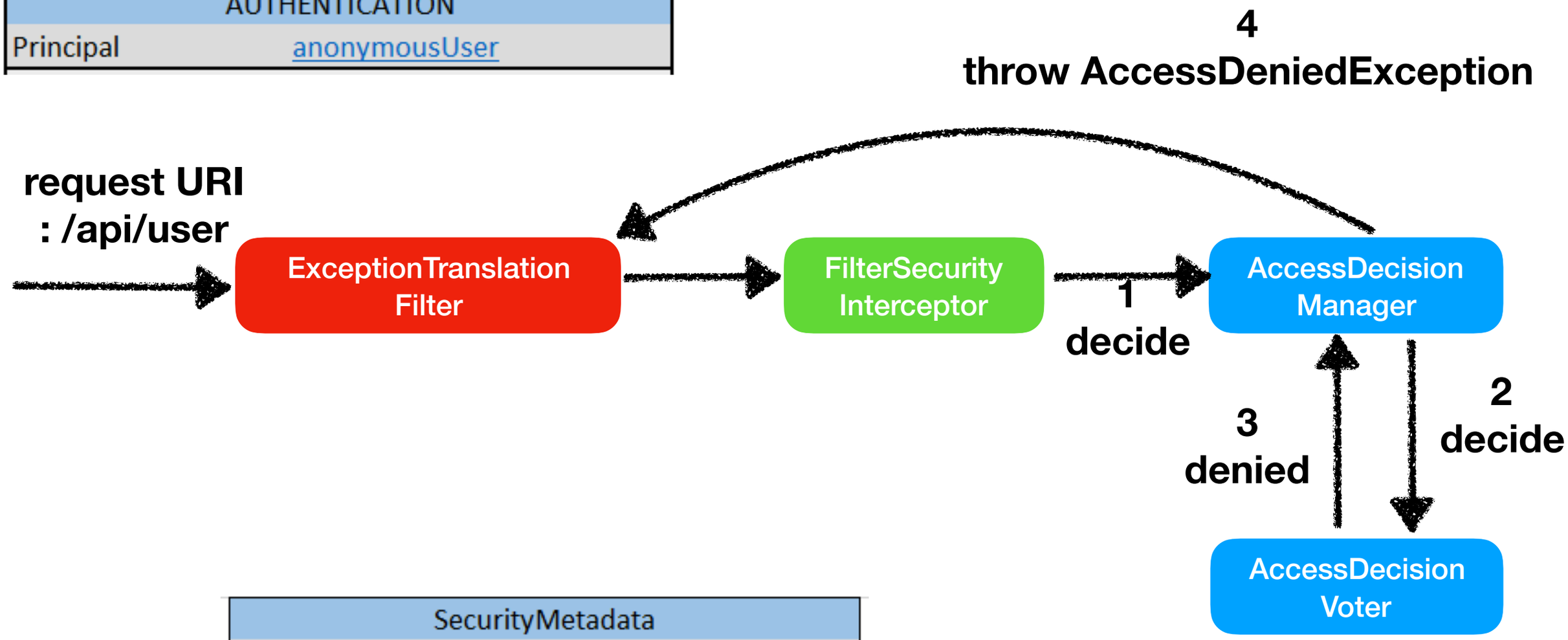
Access Denied Handler




```java
public interface AccessDeniedHandler {
    //...

    /**...*/
    void handle(HttpServletRequest request, HttpServletResponse response,
            AccessDeniedException accessDeniedException) throws IOException,
            ServletException;
}
```

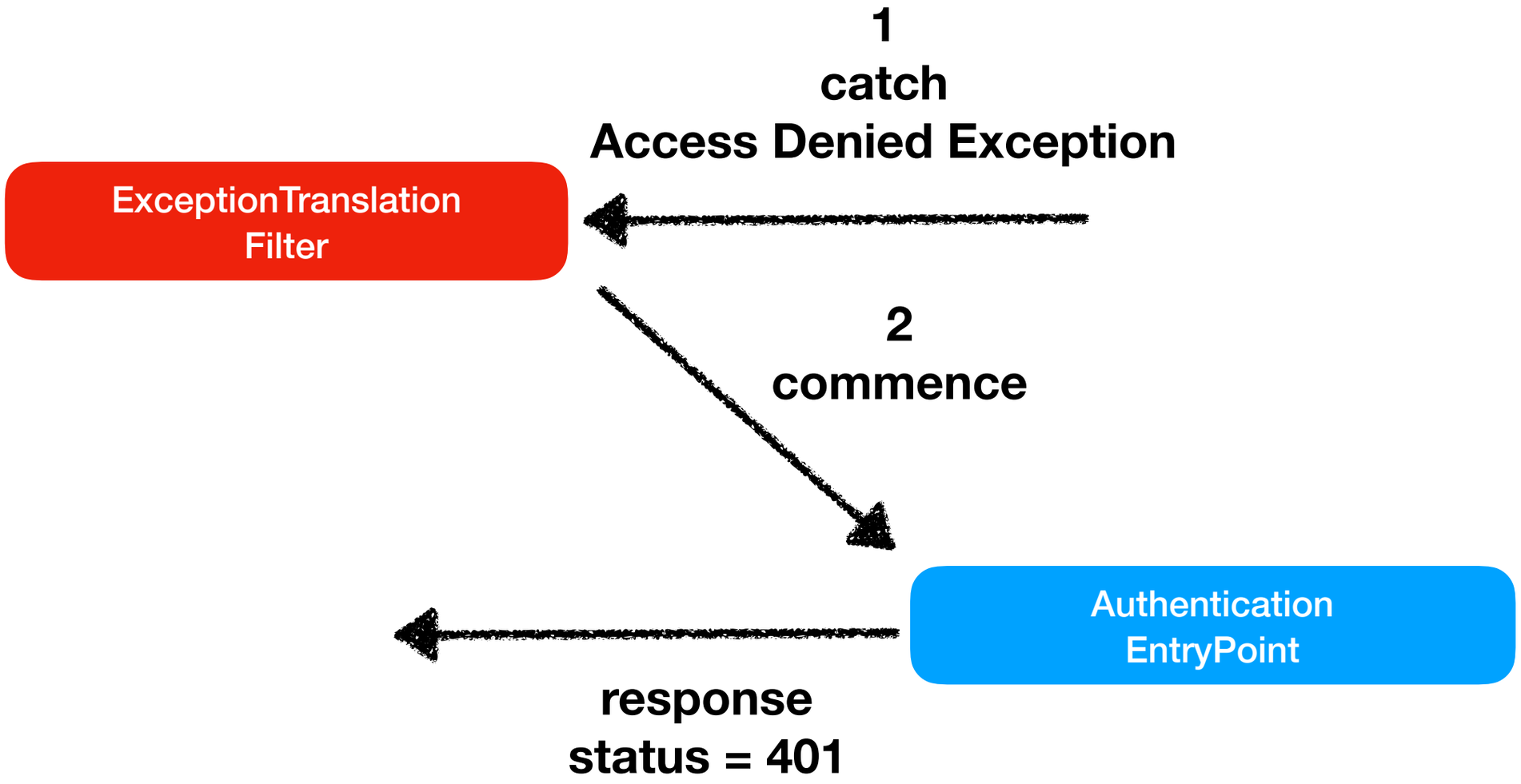
Unauthenticated

AUTHENTICATION	
Principal	anonymousUser



SecurityMetadata	
Request Pattern	/api/user
Config Attributes	ROLE_ADMIN

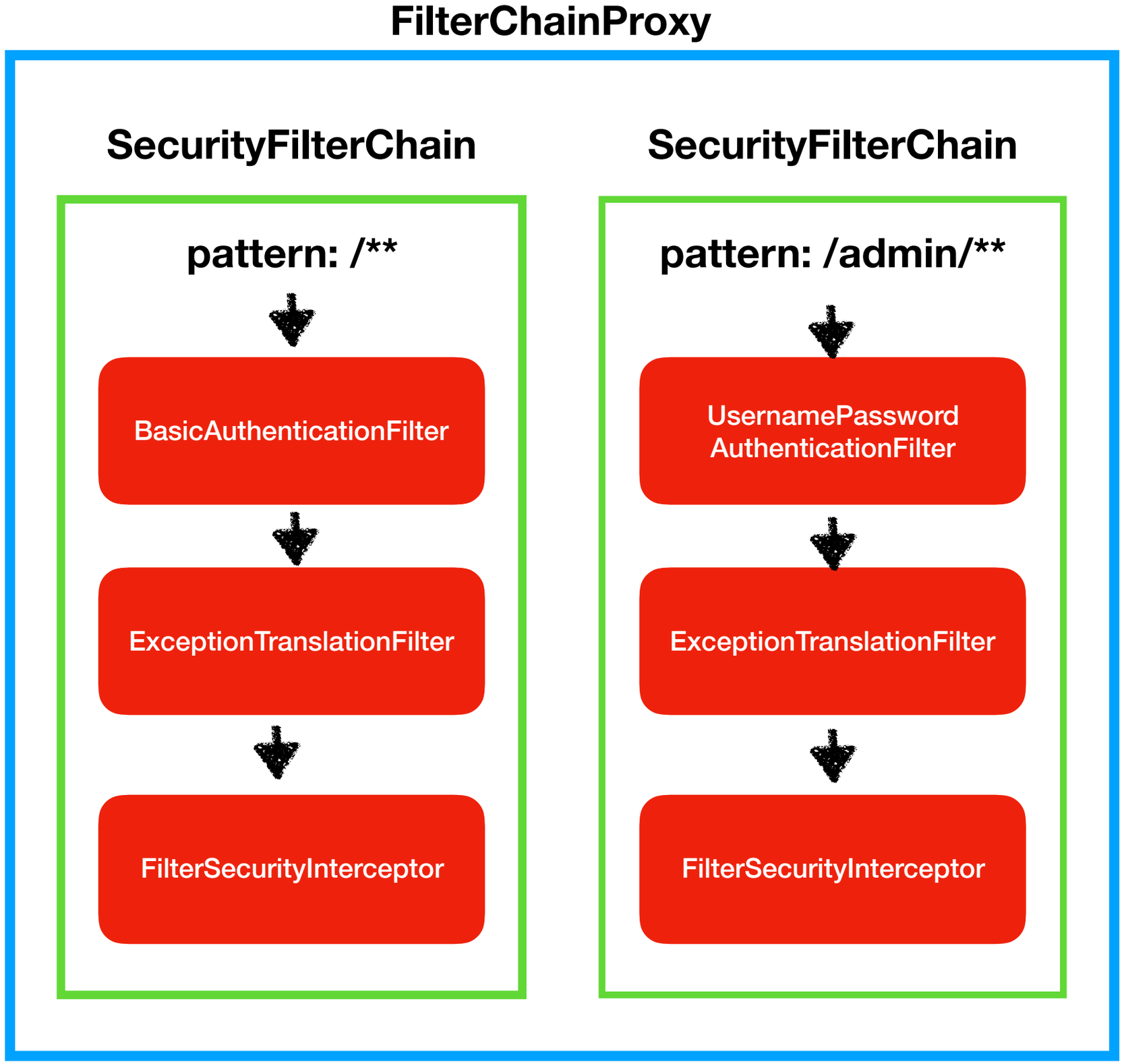
Start Authentication



WWW-Authenticate: Basic realm=spring

```
public interface AuthenticationEntryPoint {
    //...

    /**...*/
    void commence(HttpServletRequest request, HttpServletResponse response,
            AuthenticationException authException) throws IOException, ServletException;
}
```

Order matters



Configuring Spring Security in a non Boot Application using XML

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
 http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">
        <context-param>
                <param-name>contextConfigLocation</param-name>
                <param-value>
                        /WEB-INF/spring/*.xml
                </param-value>
        </context-param>

        <filter>
                <filter-name>springSecurityFilterChain</filter-name>
                <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
        </filter>
        <filter-mapping>
                <filter-name>springSecurityFilterChain</filter-name>
                <url-pattern>/*</url-pattern>
        </filter-mapping>

        <listener>
                <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
        </listener>
</web-app>
```

```
<b:beans xmlns="http://www.springframework.org/schema/security"
                xmlns:b="http://www.springframework.org/schema/beans"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:schemaLocation="http://www.springframework.org/schema/beans http://
www.springframework.org/schema/beans/spring-beans.xsd
                                                http://www.springframework.org/schema/security http://
www.springframework.org/schema/security/spring-security.xsd">

        <http authentication-manager-ref="mainAuthenticationManager"
                                entry-point-ref="serviceAccessDeniedHandler">
                <intercept-url pattern="/**" access="authenticated"/>

                <intercept-url pattern="/test/zone1/**" access="hasRole('ROLE_TEST')"/>
                <intercept-url pattern="/api/admin/" access="hasRole('ROLE_ADMIN')"/>

                <access-denied-handler></access-denied-handler>

                <form-login></form-login>

                <csrf></csrf>

                <logout></logout>

                <openid-login></openid-login>

                <remember-me></remember-me>

                <x509></x509>

                <session-management></session-management>


        </http>

        <user-service>
                <user name="user" password="{noop}password" authorities="ROLE_USER" />
        </user-service>
</b:beans>
```

What if no access is defined here as in <intercept-url pattern="/api/admin/"> ??

Username	Password	Roles
supercommandodhruv	raj_nagar_ki_tabahi	ADMIN
naagraj	Narak_Nashak_Nagraj	SERVICEX_EDIT, SERVICEX_CREATE
kroorsingh	chandrakanta@1	
shaktiman	choti.magar.moti.baatein	USER

Method Level Security is the answer

JSR 250

@Secured("ROLE_VIEWER")

@RolesAllowed("ROLE_VIEWER")

@PreAuthorize and @PostAuthorize annotations provide expression-based access control

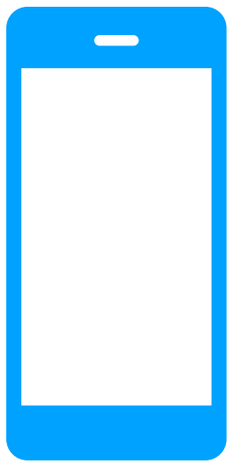
@PreAuthorize("hasRole('ROLE_VIEWER') or hasRole('ROLE_EDITOR')")

Tamedview Of OAuth2.0

Resource Owner

Aothorization Server

ROLES



Client

Resource Server

Tokens

Access Token Scope

View listed Restaurant?

Rate listed Restaurant?

Delete the listing?

HTTP vs HTTPS for OAuth2

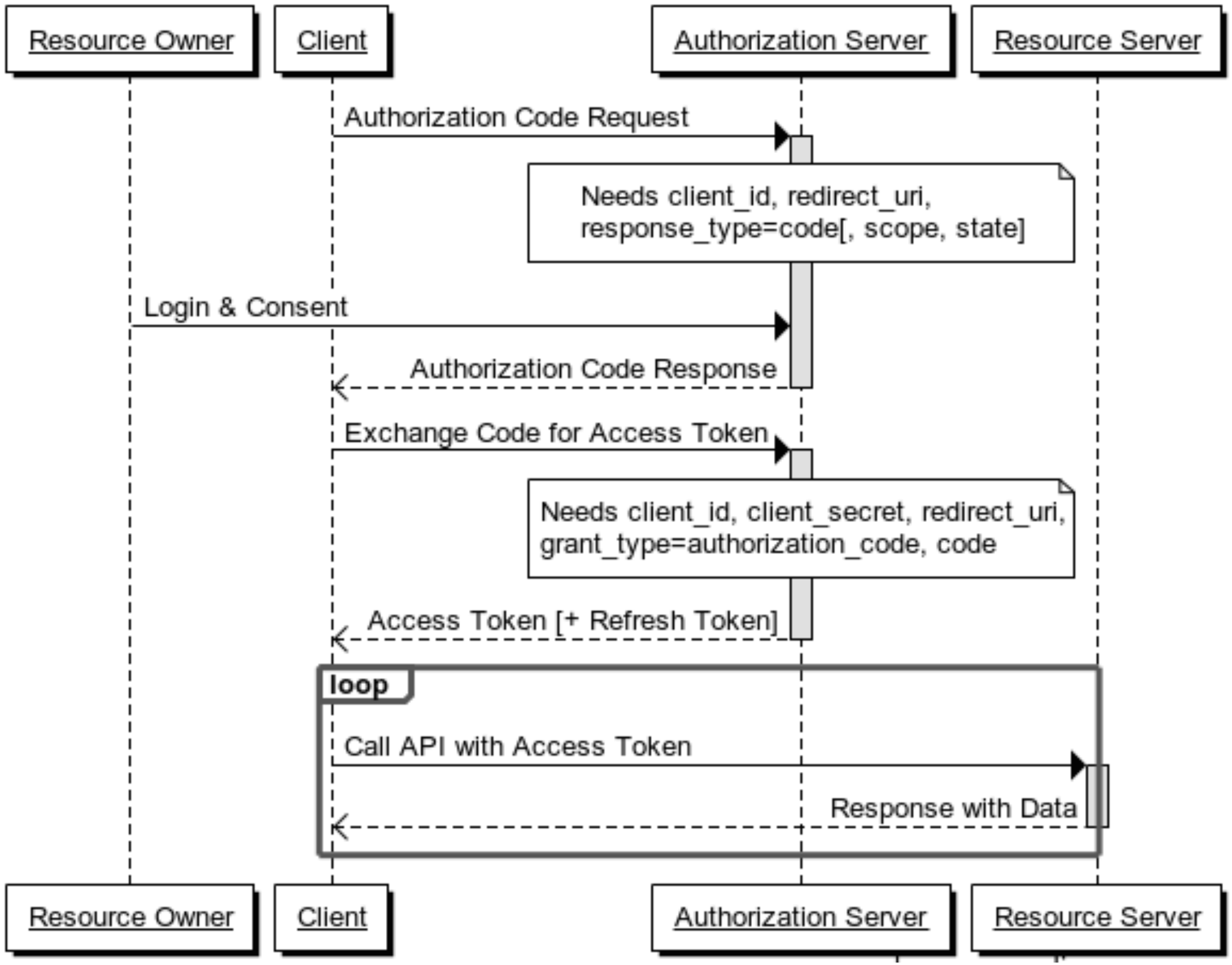
Client Registration

- Application Name
- Redirect URLs : URL's of the client to receive authorization code and access token code
- Grant Types: Authorization Types that will be used by client
- Javascript Originated requests (optional): the hostname
- Authorization server response: Client Id, Client Secret

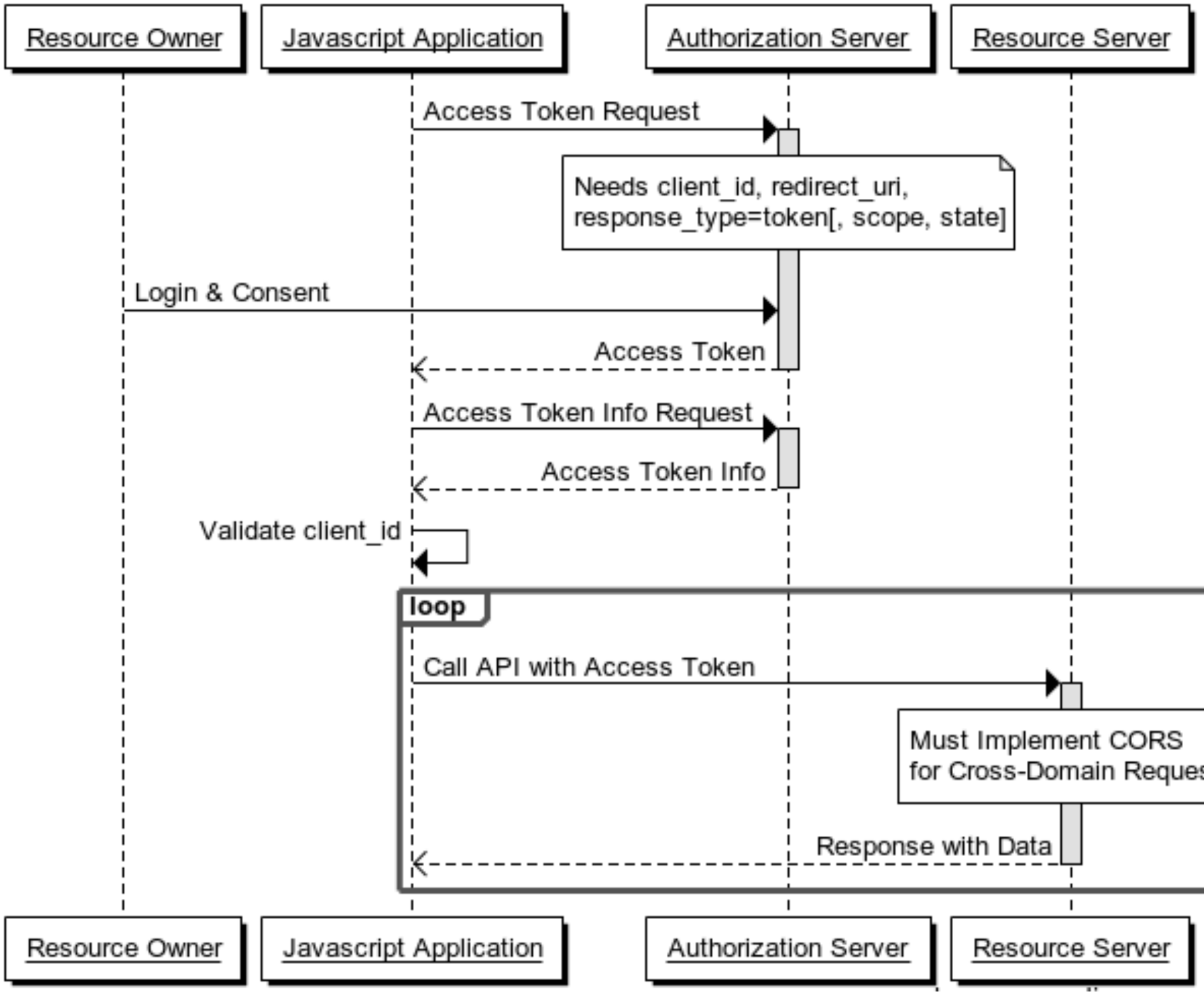
Auth Grant Types

- Authorization Code Grant / (Refresh tokens involved)
- Implicit Grant / (Refresh tokens not involved, Java Script Application)
- Resource Owner Password Credentials Grant
- Client Credentials Grant

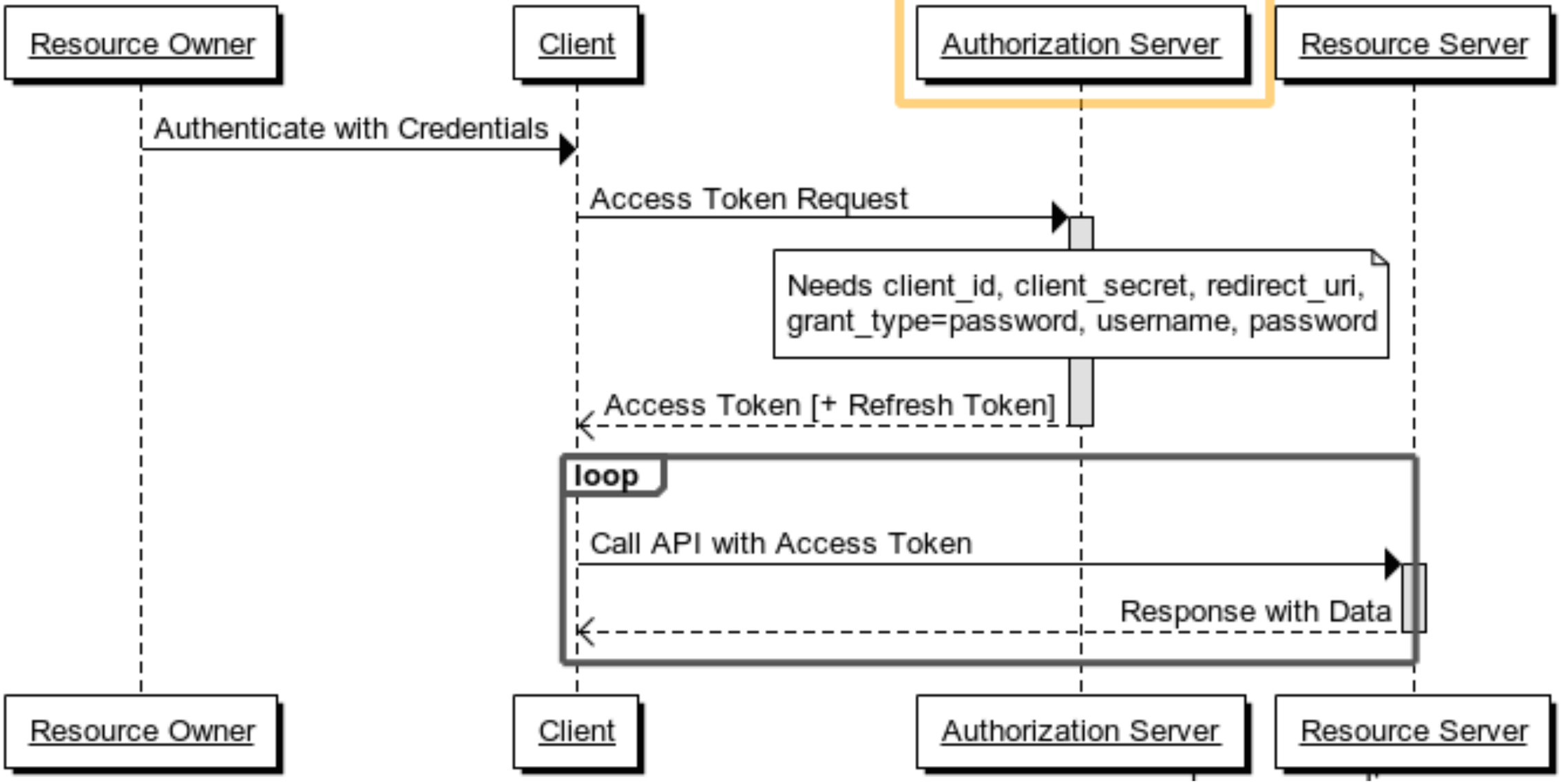
Authorization Code Grant Flow



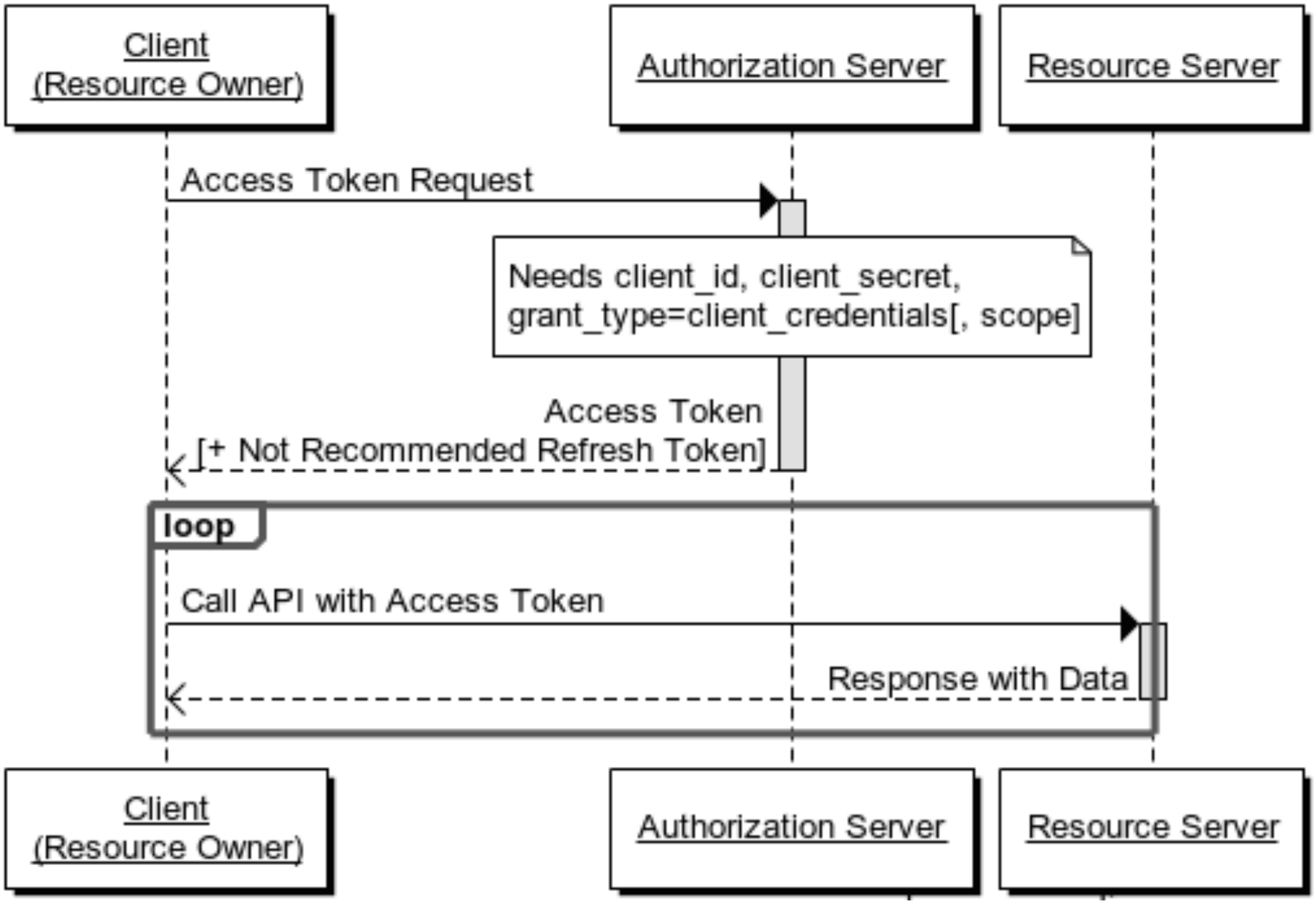
Implicit Grant Flow



Resource Owner Password Credentials Grant Flow



Client Credentials Grant Flow



More to come ...