# KIP-432: Additional Broker-Side Opt-In for Default, Unsecure SASL/OAUTHBEARER Implementation

*This page is meant as a template for writing a KIP. To create a KIP choose Tools->Copy on this page and modify with your content and replace the heading with the next KIP number and a description of your issue. Replace anything in italics with your own description.*

## Status

**Current state**: DISCARDED

**Discussion thread**: *here*

**JIRA**: *KAFKA-7960*

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

## Motivation

The default implementation of SASL/OAUTHBEARER, as per KIP-255, is unsecured.  This is useful for development and testing purposes, and it provides a great out-of-the-box experience, but it must not be used in production because it allows the client to authenticate with any principal name it wishes.  To enable the default unsecured SASL/OAUTHBEARER implementation on the broker side simply requires the addition of `OAUTHBEARER` to the `sasl.enabled.mechanisms` configuration value (for example: `sasl.enabled.mechanisms=GSSAPI,OAUTHBEARER` instead of simply `sasl.enabled.mechanisms=GSSAPI`). To secure the implementation requires the explicit setting of the `listener.name.{sasl_plaintext|sasl_ssl}.oauthbearer.sasl.{login,server}.callback.handler.class` properties on the broker.  The question then arises: what if someone either accidentally or maliciously appended `OAUTHBEARER` to the `sasl.enabled.mechanisms` configuration value?  Doing so would enable the unsecured implementation on the broker, and clients could then authenticate with any principal name they desired.

This KIP proposes to add an additional opt-in configuration property on the broker side for the default, unsecured SASL/OAUTHBEARER implementation such that simply adding `OAUTHBEARER` to the `sasl.enabled.mechanisms` configuration value would be insufficient to enable the feature.  This additional opt-in broker configuration property would have to be explicitly set to true before the default unsecured implementation would successfully authenticate users, and the name of this configuration property would explicitly indicate that the feature is not secure and must not be used in production.  Adding this explicit opt-in is a breaking change; existing uses of the unsecured implementation would have to update their configuration to include this explicit opt-in property before their cluster would accept unsecure tokens again.  Note that this would only result in a breaking change in production if the unsecured feature is either accidentally or maliciously enabled there; it is assumed that 1) this will probably not happen to anyone; and 2) if it does happen to someone it almost certainly would not impact sanctioned clients but would instead impact malicious clients only (if there were any).

## Public Interfaces

A new broker configuration property will be added that defaults to `false` and that must explicitly be set to `true` before the default unsecured token validation callback handler (`org.apache.kafka.common.security.oauthbearer.internals.unsecured.OAuthBearerUnsecuredValidatorCallbackHandler`) will successfully validate unsecured tokens. The configuration property must explicitly and visibly identify that an unsecure option that must not be used in production is being enabled.  The configuration property is:

```
yes.virginia.i.really.do.want.to.allow.unsecured.oauthbearer.tokens.because.this.is.not.a.production.cluster
```

The above property has no effect if a different (i.e. non-default, and presumably secure) token validation callback handler is being used.

## Proposed Changes

There are no changes beyond those as already described above.

## Compatibility, Deprecation, and Migration Plan

As mentioned above, adding this explicit opt-in is a breaking change in the sense that existing uses of the unsecured implementation would need to update their configuration to include this explicit opt-in property before their cluster would accept unsecure tokens again.  This would only result in a breaking change in production if the unsecured feature is either accidentally or maliciously enabled there.  It is assumed that 1) this will probably not happen to anyone; and 2) if it does happen to someone it almost certainly would not impact sanctioned clients but would instead impact malicious clients only (if there were any).

## Rejected Alternatives

None