Configure a PasswordExpirationJob

> **Version Warning**
>
> The content below is for Apache Syncope <= 1.2 - for later versions the Reference Guide is available.

This page explains how you can configure a PasswordExpirationJob inside Apache Syncope. The PasswordExpirationJob searches all users whose passwords are expired from a certain number of days and suspends them.

1. Create a new Java class for your Scheduled Job.
2. Configure a new Scheduled Task.
3. From the Apache Syncope console, create a new configuration schema and set the expiration days.

PasswordExpirationJob

```
public class PasswordExpirationJob extends AbstractTransactionalTaskJob {

    @Autowired
    private UserController userController;

    @Autowired
    private ConfDAO confDAO;

    @Autowired
    private EntitlementDAO entitlementDAO;

    @Autowired
    private EntityManager entityManager;

    @Override
    protected String doExecute(final boolean dryRun) throws
JobExecutionException {
        if (!(task instanceof SchedTask)) {
            throw new JobExecutionException("Task " + taskId + " isn't a
SchedTask");
        }

        //Take the xDays parameter from Syncope configuration
        final CAttr expirationDays = confDAO.find("expirationDays", 10);

        // Build your time condition with expirationDays parameter
        final Calendar yourTimeCondition = ....

        // Search all user that match your condition
        final Query query = entityManager.createNativeQuery(
                "SELECT id FROM SyncopeUser WHERE changePwdDate < ?1");
        query.setParameter(1, yourTimeCondition);

        @SuppressWarnings("unchecked")
        final List<Long> users = (List<Long>) query.getResultList();
```

```
        if (!dryRun) {
            try {
                // Exec the operation with admin user
                final List<GrantedAuthority> authorities = new
ArrayList<GrantedAuthority>();
                for (Entitlement entitlement : entitlementDAO.findAll()) {
                    authorities.add(new SimpleGrantedAuthority(entitlement.
getName()));
                }
                final UserDetails userDetails = new User("admin",
"FAKE_PASSWORD", true, true, true, true, authorities);
                SecurityContextHolder.getContext().setAuthentication(
                        new UsernamePasswordAuthenticationToken
(userDetails, "FAKE_PASSWORD", authorities));

                // for all user
                for (Long userId : users) {
                    final StatusMod statusMod = new StatusMod();
                    statusMod.setId(userId);
                    statusMod.setOnSyncope(true);
                    statusMod.setType(StatusMod.ModType.SUSPEND);
                    userController.status(statusMod);

                }
            } finally {
                // Remove admin permission
                SecurityContextHolder.clearContext();
            }
        }

        return (dryRun
                ? "Will Suspend"
                : "Suspended") + " " + users.size() + " utenti";
    }

    @Override
    protected boolean hasToBeRegistered(final TaskExec execution) {
        return true;
    }
}
```