# Why

We used to use below methods to get access to JCR repository for adding/deleting/moving or any such actions

The following methods are deprecated:

* ResourceResolverFactory.getAdministrativeResourceResolver
* ResourceProviderFactory.getAdministrativeResourceProvider
* SlingRepository.loginAdministrative

All the above are an overkill on admin login. So we use Service Authentication to overcome discussed limitation

# Service Authentication

Service authentication can be attained by 3 ways

# 2.1 Service User Mapping

Instead of directly using the admin login session, we can use

1. Administrators can create a service user who has necessary access on specific parts of JCR
2. The above service user has to be used by all the OSGi bundles to manipulate JCR
3. A list of all bundles which can make use of service user are to configured in /syste/console/configMgr OR using PID { org.apache.sling.serviceusermapping.impl.ServiceUserMapperImpl}in crx. This is in full control of admins .
4. bundleSymbolicName: subServiceName= serviceUserID

serviceUserID 🡪 userId created in system with all necessary privileges over repository

bundleSymbolicName 🡪 fully qualified name of the bundle

subServiceName 🡪 an optional one . Just for readability. For example we have a bundle com.work.manipulate.Repo. It again has many services as com.work.manipulate.Repo.write 🡪Service used to write to repo

com.work.manipulate.Repo.read 🡪 Service used to read

com.work.manipulate.Repo.update 🡪 Service used to update repo

The optional subservice names can be read, write and update for above services.

Hence *com.work.manipulate.Repo:* read=readServiceUser

readServiceUser is created by admins with appropriate privileges to read the JCR.

1. And in backend code, **serviceUserMapper** api has to be invoked to get the service user.
2. String getServiceUserID(Bundle bundle, String subServiceName);

The implementation uses two fallbacks in case no mapping can be found for the given sub ServiceName

* 1. Use user/principal mapping for the serviceName only (not considering subServiceName)
  2. Use default user 🡪 user.default (if one is configured in the OSGi configuration for PID org.apache.sling.serviceusermapping.impl.ServiceUserMapperImpl)
  3. Use default mapping 🡪 user.enable.default.mapping (if it is enabled in the OSGi configuration for PID org.apache.sling.serviceusermapping.impl.ServiceUserMapperImpl) which looks up a user with id serviceuser--<bundleId>[--<subservice-name>] (since Service User Mapper 1.3.0, SLING-6227).

# ResourceResolverFactory

* ResourceResolverFactory service is enhanced with a new factory method as a support for service access to the Resource Tree.
* This method allows for access to the resource tree for services where the service bundle is the bundle actually using the ResourceResolverFactory service.

ResourceResolver getServiceResourceResolver(Map<String, Object> authenticationInfo **throws** LoginException;

* authenticationInfo is a Map which can have below keys and respective values :

**ResourceResolverFactory.PASSWORD**

**ResourceResolverFactory.SUBSERVICE**

**ResourceResolverFactory.USER**

**ResourceResolverFactory.USER\_IMPERSONATION**

* This method allows for access to the resource tree for services where the service bundle is the bundle actually using the ResourceResolverFactory service.

# Sling Repository

* loginService is enhanced to support JCR Repository access for services.

Session **loginService**(String subServiceName, String workspace)

**throws** LoginException, RepositoryException;

* This method allows for access to the JCR Repository for services where the service bundle is the bundle actually using the SlingRepository service.

# NOTE

After the configuration for service user mappings are done, the user/principal must exist at the point in time where ResourceResolverFactory.getServiceResourceResolver(...) or SlingRepository.loginService(...) is called.

We can defer the bundle from starting if we don’t have a particular service mapping by using **serviceUserMapper** api.

This can be used in **@Activate** method and see if a mapping is available or else bundle won’t be started.

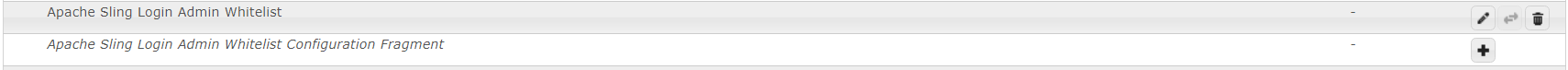
# Configuration

* For each service/subservice name combination an according mapping needs to be provided. The mapping binds a service name/subservice name to a JCR system user or a principal.
* org.apache.sling.serviceusermapping.impl.ServiceUserMapperImpl.amended - PID to be used for creating OSGiConfigs using the property names ***user.mapping*** in below format



# Whitelisting bundles

If we don’t want above methods to get access to JCR/resource tree and rather need to by-pass above process, we should whitelist the bundles using the PID -   
**{ org.apache.sling.jcr.base.internal.LoginAdminWhitelist.fragment }**



Whitelist – to bypass all bundles and use admin resource resolver

Whitelist Configuration Fragment – to whitelist only few bundles

# Useful Resources

Clear explaination : <https://sling.apache.org/documentation/the-sling-engine/service-authentication.html>