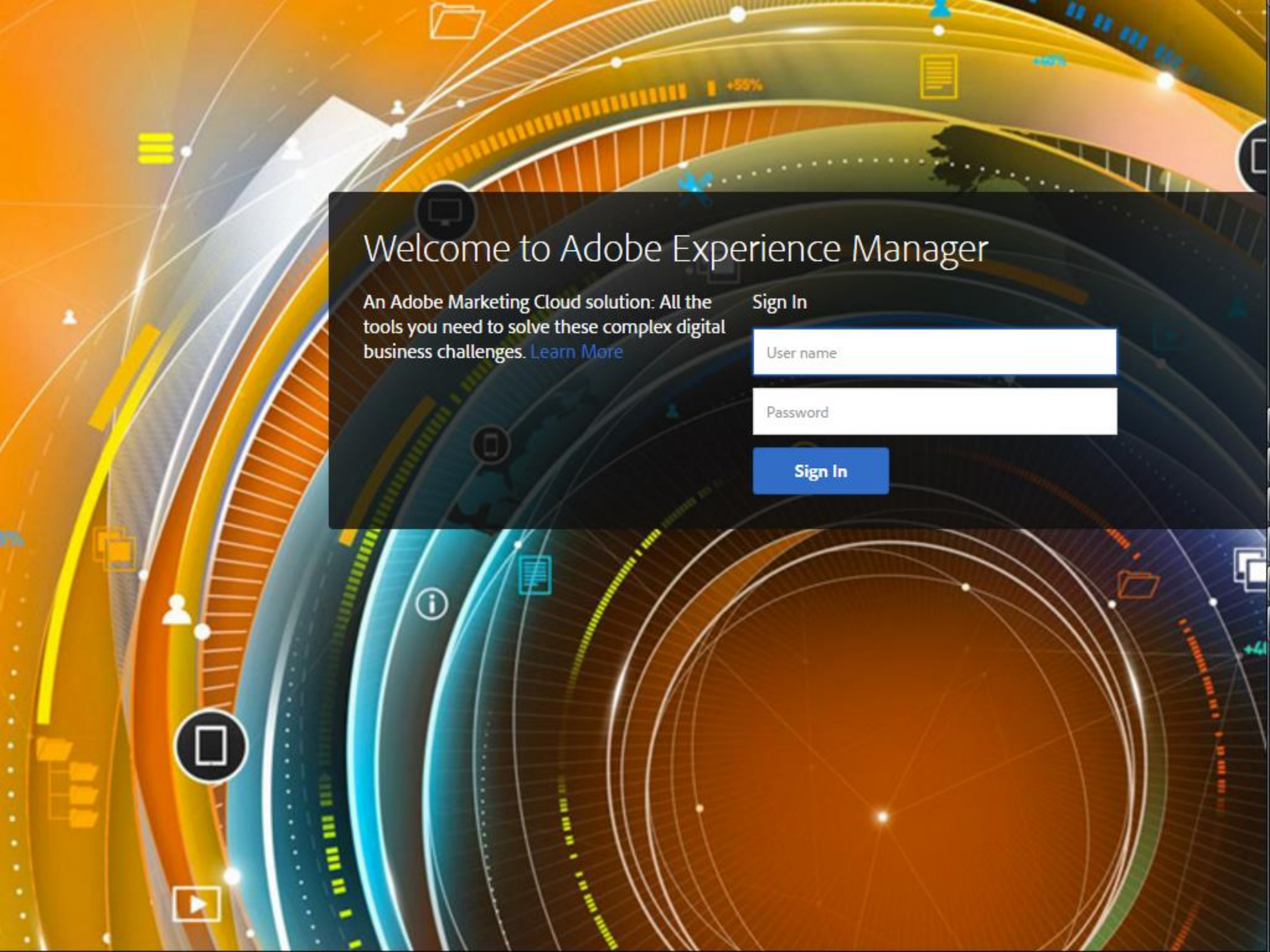
JCR 2.0 Authorization Fundamentals

With Apache Jackrabbit Oak examples

AEM
COMMUNITY
BELARUS

by Vitaly Kiselev



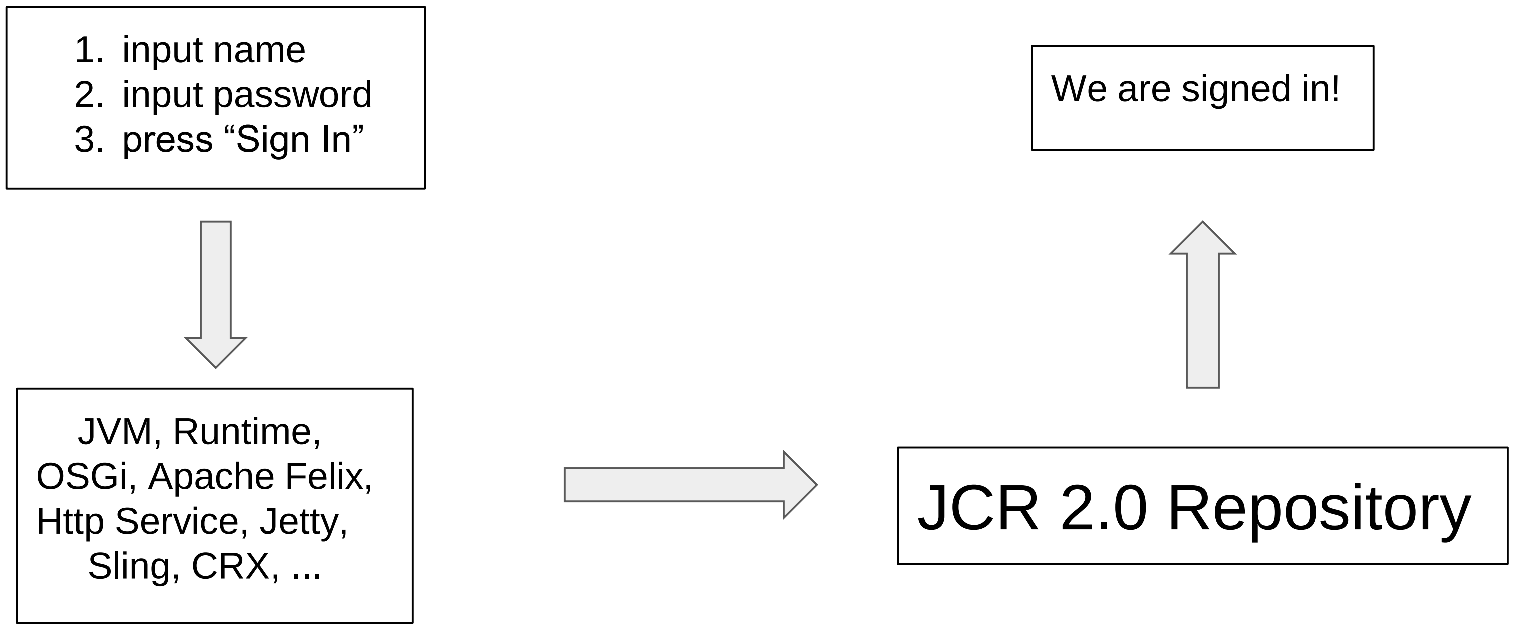
Welcome to Adobe Experience Manager

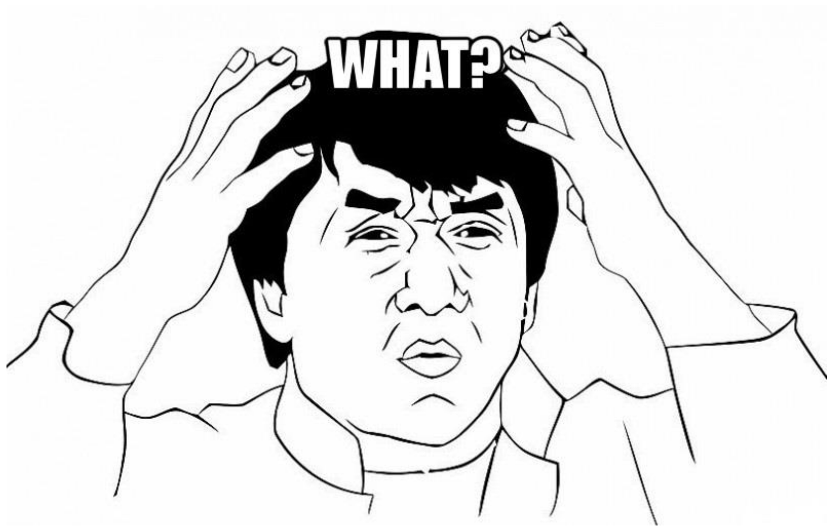
An Adobe Marketing Cloud solution: All the tools you need to solve these complex digital business challenges. Learn More

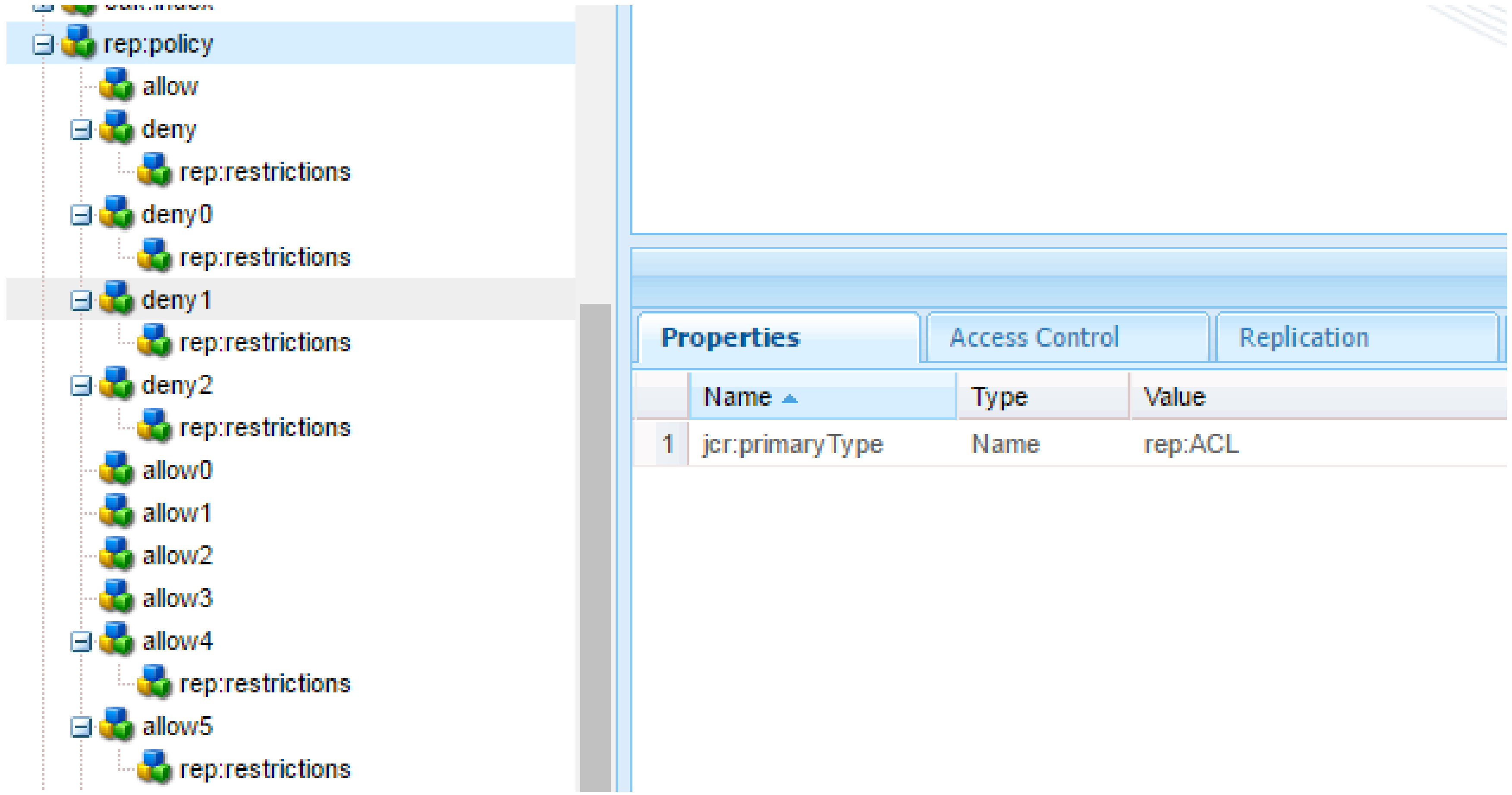
Sign In

1. input name
2. input password
3. press “Sign In”

We are signed in!


1. input name
2. input password
3. press “Sign In”
JVM, Runtime, OSGi, Apache Felix, Http Service, Jetty, Sling, CRX, ...
JCR 2.0 Repository
We are signed in!


WHAT?

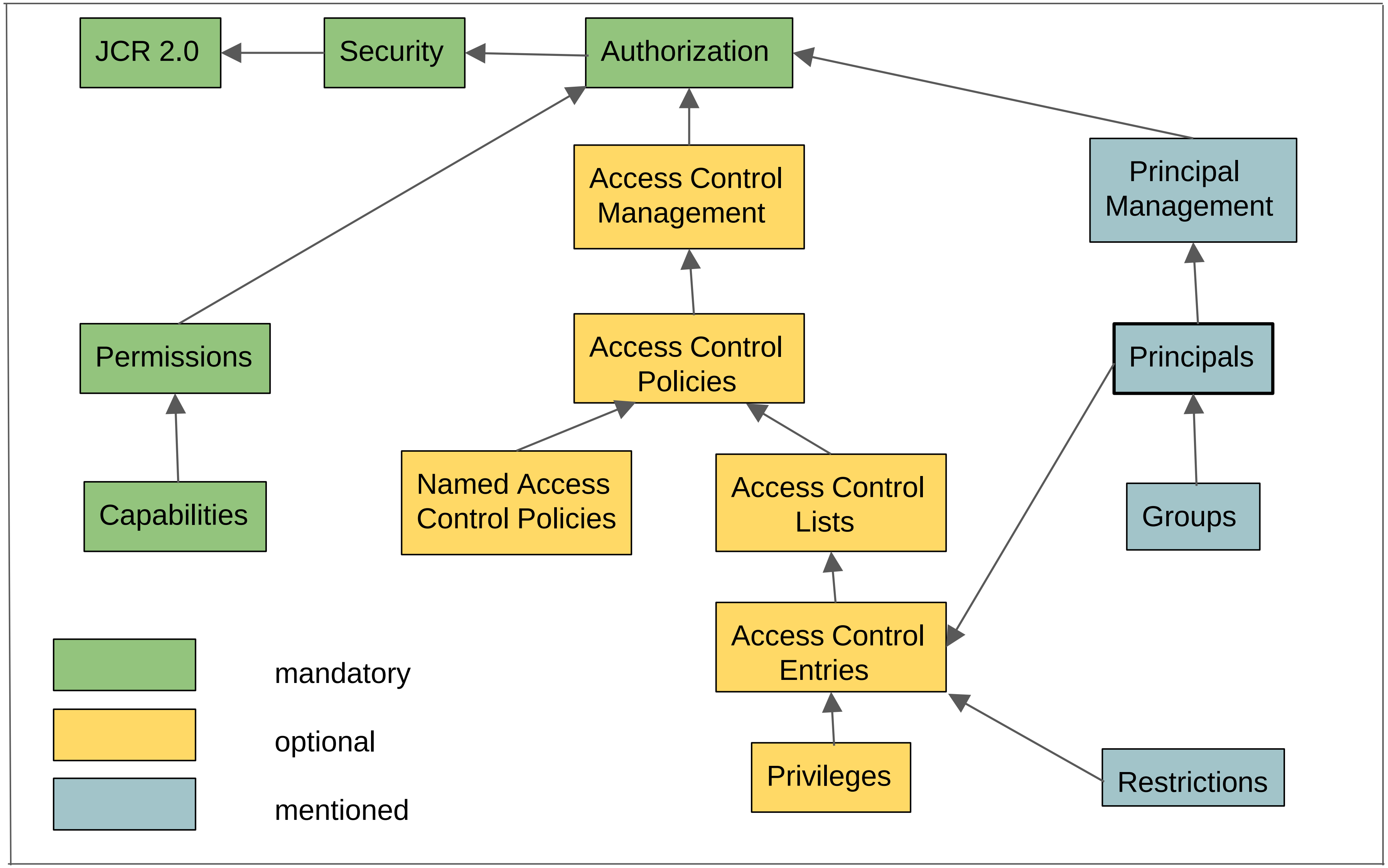


```
rep:policy
    allow
    deny
        rep:restrictions
    deny0
        rep:restrictions
    deny1
        rep:restrictions
    deny2
        rep:restrictions
    allow0
    allow1
    allow2
    allow3
    allow4
        rep:restrictions
    allow5
        rep:restrictions
```

Properties | Access Control | Replication

	Name	Type	Value
1	jcr:primaryType	Name	rep:ACL

Agenda


JCR 2.0
Security
Authorization
Access Control Management
Principal Management
Permissions
Access Control Policies
Principals
Capabilities
Named Access Control Policies
Access Control Lists
Groups
Access Control Entries
Privileges
Restrictions
mandatory
optional
mentioned

JCR

JCR - Specifications of the Java Content Repository API

Repository = File System + Database + other features

JCR 1.0 - JSR-170 from 17 Jun, 2005

JCR 2.0 - JSR-283 from 25 Sep, 2009

JSR - Java Specification Request

Relationship between JCR and Apache Oak

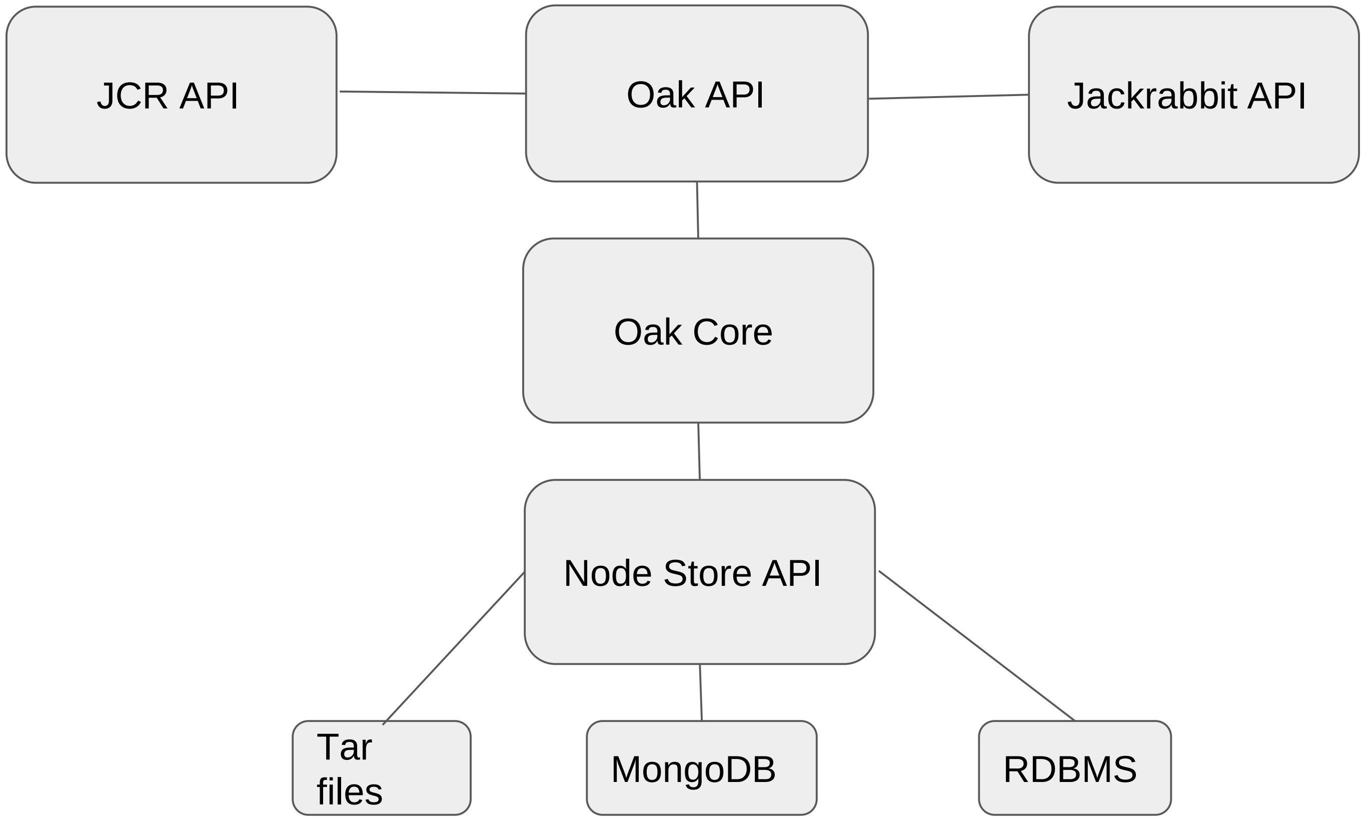
Apache Jackrabbit - a fully conforming implementation of JCR specification.

CRX - upgraded Jackrabbit repository was started to develop by Day Software Company.

Jackrabbit was developed in 2000-s years but in 2010-s its architecture doesn’t correspond to the modern web demands or it becomes too hard to implement needed improvements.

Therefore Apache Oak has been created which is a scalable and performant modern complementary implementation of the JCR specification.

Apache Jackrabbit Oak Architecture



JCR Security

JCR Security:

- Authentication
- Authorization
- JCR has Session Management and hasn’t User Management.

Usually JCR Implementations add User Management.

Apache Jackrabbit (and therefore Oak) has User Management.

Difference between Authentication and Authorization

Authentication	Authorization		
greeting	give a smoke	lend money	romantic date
anonymous	+	-	-
collegue	+	+	-
girlfriend	+	+	+

JCR Authorization

- Mandatory
  - Permissions Management
    - permissions
    - capabilities
- Optional
  - Access Control Management
    - policies
    - lists
    - entries
    - privileges
    - restrictions
  - Principal Management

Permissions

Permissions (JSR-283, §9.1) encompass the restrictions imposed by any **access control** restrictions that may be in effect upon the content of a repository, either **implementation specific** or **JCR-defined**. Permissions are reported through:

```
boolean Session.hasPermission(String absPath, String actions)

void Session.checkPermission(String absPath, String actions)
                    throws AccessDeniedException
```

The actions parameter is a comma separated list of action strings:

- read (Session.ACTION_READ)
- add_node (Session.ACTION_ADD_NODE)
- set_property (Session.ACTION_SET_PROPERTY)
- remove (Session.ACTION_REMOVE)

Methods for testing restrictions more broadly are provided by the capabilities.

Capabilities

Capabilities (JSR-283, §9.2) encompass the restrictions imposed by **permissions**, but also include any further restrictions **unrelated to access control**. For checking whether an operation can be performed given as much context as can be determined by the repository, including:

- Permissions granted to the current user, including access control privileges
- Current state of the target object (reflecting locks, checked-out status, retention and hold status etc.)
- Repository capabilities
- Node type-enforced restrictions
- Repository configuration-specific restrictions

boolean Session.hasCapability(String methodName, Object target, Object[] arguments)

Access Control Management

Repository.OPTION_ACCESS_CONTROL_SUPPORTED - repository descriptor about supporting access control by particular implementation

Access Control Management (JSR-283, §16) : package **javax.jcr.security**

- Privilege discovery
- Assigning access control policies

Access control (JSR-283, §16.1) is exposed through a

    javax.jcr.security.AccessControlManager

acquired from the Session using

    AccessControlManager Session.getAccessControlManager()

Privileges

A privilege (JSR-283, §16.2) represents the ability to perform **a particular set of operations** on a node. Each privilege is identified by a JCR name and may be **aggregate** and **abstract** (implementation specific):

jcr:all (**is never abstract**) jcr:read jcr:write jcr:readAccessControl jcr:modifyAccessControl jcr:lockManagement jcr:versionManagement jcr:nodeTypeManagement	jcr:write jcr:modifyProperties jcr:addChildNodes jcr:removeNode jcr:removeChildNodes

Privileges discovery

```
Privilege[] AccessControlManager.getSupportedPrivileges(String absPath)

Privilege AccessControlManager.privilegeFromName(String privilegeName)

public interface javax.jcr.security.Privilege {
        String Privilege.getName()
        Boolean Privilege.isAbstract()
        Boolean Privilege.isAggregate()
        ...
}

boolean AccessControlManager.hasPrivileges(String absPath, Privilege[] privileges)

Privilege[] AccessControlManager.getPrivileges(String absPath)
```

Access Control Policies

Access Control Policies (JSR-283, §16.3) are assigned to nodes for **controlling the privileges** granted to a user.

JCR provides a marker interface **AccessControlPolicy** and means to:

- find which policies are available to be bound to a node
- bind a policy to a node
- get the policies bound to a given node (including transient modifications)
- get the policies that affect access to a given node
- unbind a policy from a node

Any effect that a policy has on a node is always reflected in the information returned by the privilege discovery methods.

Access Control Policies discovery

AccessControlPolicyIterator AccessControlManager.getApplicablePolicies(String absPath)

AccessControlPolicy[] AccessControlManager.getPolicies(String absPath)

AccessControlPolicy[] AccessControlManager.getEffectivePolicies(String absPath)

void AccessControlManager.setPolicy(String absPath, AccessControlPolicy policy)

void AccessControlManager.removePolicy(String absPath, AccessControlPolicy policy)

Named Access Control Policies

```
interface NamedAccessControlPolicy extends AccessControlPolicy {
                String NamedAccessControlPolicy.getName();
}
```

Named Access Control Policy (JSR-283, §16.4) represents an **opaque**, **immutable** policy **with a name**, which must be a JCR name.

Access Control Lists (ACLs)

```
interface Access Control Lists extends AccessControlPolicy {
                AccessControlEntry[] AccessControlList.getAccessControlEntries();
                Boolean AccessControlList.addAccessControlEntry(Principal principal, Privilege[] privileges);
                Void AccessControlList.removeAccessControlEntry(AccessControlEntry ace);
}
```

Access Control List (JSR-283, §16.5) represents **a list of AccessControlEntry (ACE) objects**. Before being bound to a node, the AccessControlList is mutable.

The user must have privileges:

- **jcr:modifyAccessControl** to add or remove access control entries
- **jcr:readAccessControl** to read access control entries from an AccessControlList

Access Control Entries (ACEs)

AccessControlEntry (JSR-283, §16.5.1) represents **the association** of one or more **Privilege objects** with a specific **java.security.Principal**.

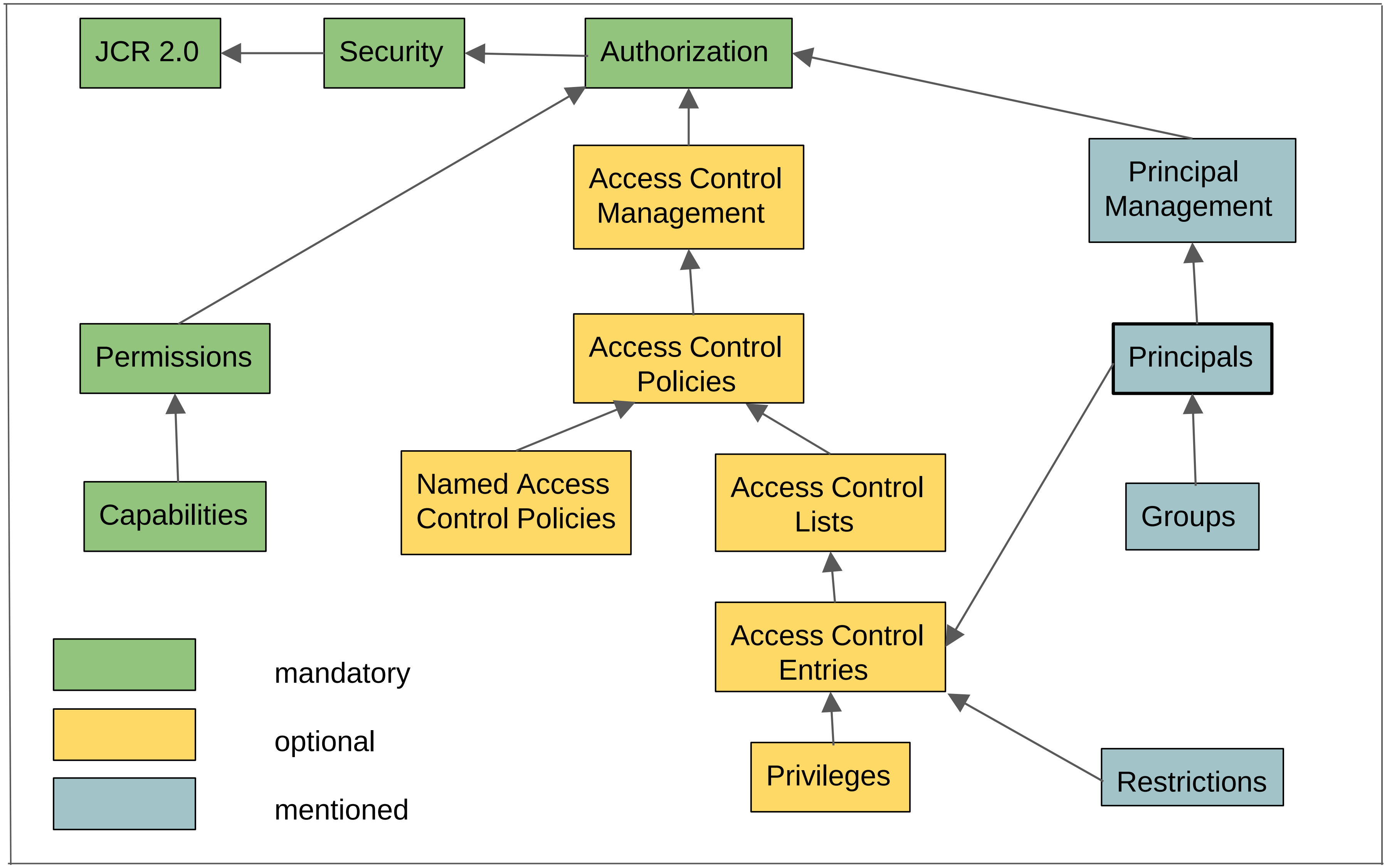
```
public interface AccessControlEntry {
        Principal getPrincipal();
        Privilege[] getPrivileges();
}
```

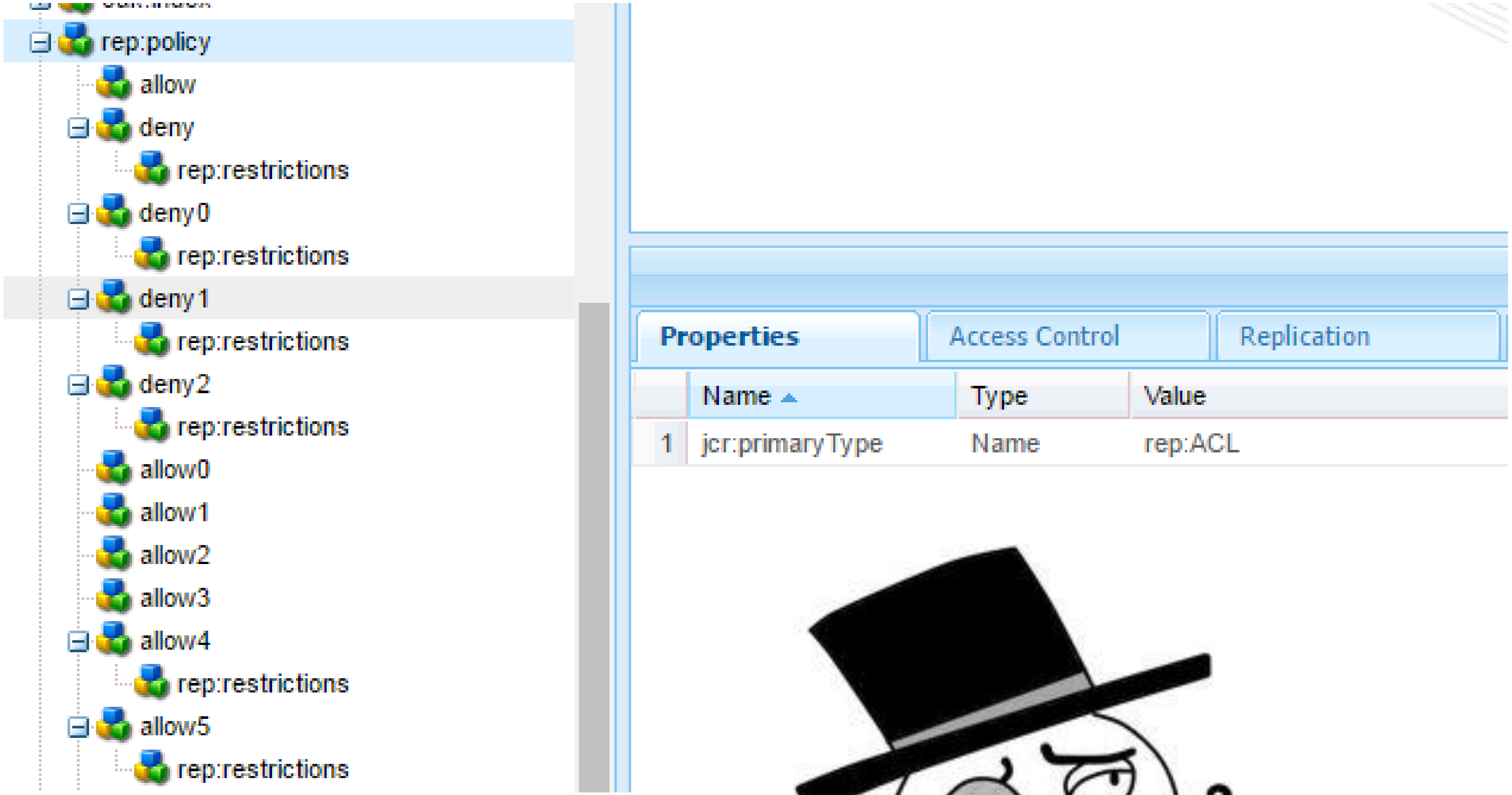
Principals

The discovery (JSR-283, §16.5.7) of principals (**java.security.Principal**) is outside the scope of specification.

```
public interface Principal {
        public boolean equals(Object another);
        public String toString();
        public int hashCode();
        public String getName();
}

public interface java.security.acl.Group extends Principal {
        public boolean addMember(Principal user);
        public boolean removeMember(Principal user);
        public boolean isMember(Principal member);
        public Enumeration<? extends Principal> members();
}
```


JCR 2.0
Security
Authorization
Access Control Management
Principal Management
Permissions
Access Control Policies
Principals
Capabilities
Named Access Control Policies
Access Control Lists
Groups
Access Control Entries
Privileges
Restrictions
mandatory
optional
mentioned



	Name	Type	Value
1	jcr:primaryType	Name	rep:ACL

Contacts: vitaly.kiselev@axamit.com