

3DE 权限配置详解

基于 2020x



3DEXPERIENCE®

Version 1.0 - 1/21/2021

Written by: Zhao Liang
Validated by: Zhao Liang

Table of contents

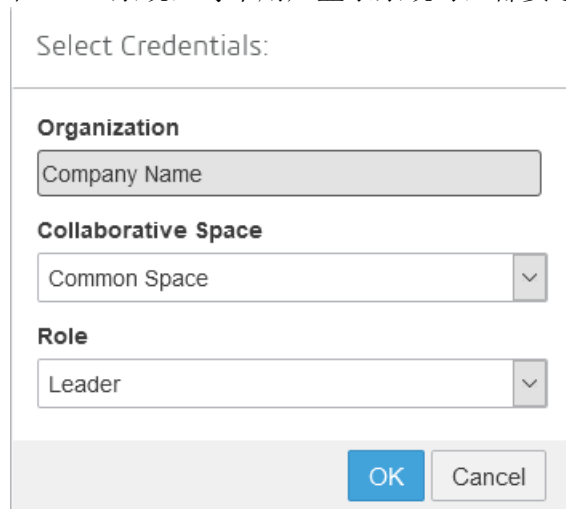
1. 概述	3
2. 用户凭据	3
3. 用户默认权限	6
4. 3DE 相关的一些权限概念与机制	8
5. 固定权限设定示例解析	11
5.1. Policy 权限设置的一些新参数	12
5.2. In Work 状态权限设置解析	13
5.3. Release 状态权限设置解析	15
5.4. FROZEN 状态权限设置解析	16
6. 临时赋权	18
6.1. Grant 赋权模式	19
6.2. Ownership 赋权模式	21
6.2.1. Explicit SOV	21
6.2.2. Inherited ownership	22
6.2.3. Inherited access	23
6.2.4. Inherited ownership VS Inherited access	23
6.2.5. 对象进行 clone 和 revise 操作后, 临时赋权的变化	24

1. 概述

对于 3DE 权限的设定，在大部分项目都是一个绕不开的话题，但是，我们现在大多的 enovia 开发工程师所掌握和习惯的，还是 15x 之前的那种权限设定管理方式，原来的权限管理方式并非是 3DE 以后的主要方向，为了大家更好的理解 3DE 目前的权限机制，基于我个人的理解，我整理了这份文档，希望能给大家带来一定的帮助。

2. 用户凭据

在 3DE 系统，每个用户登录系统时，都要选择一个上下文凭据，如下图：

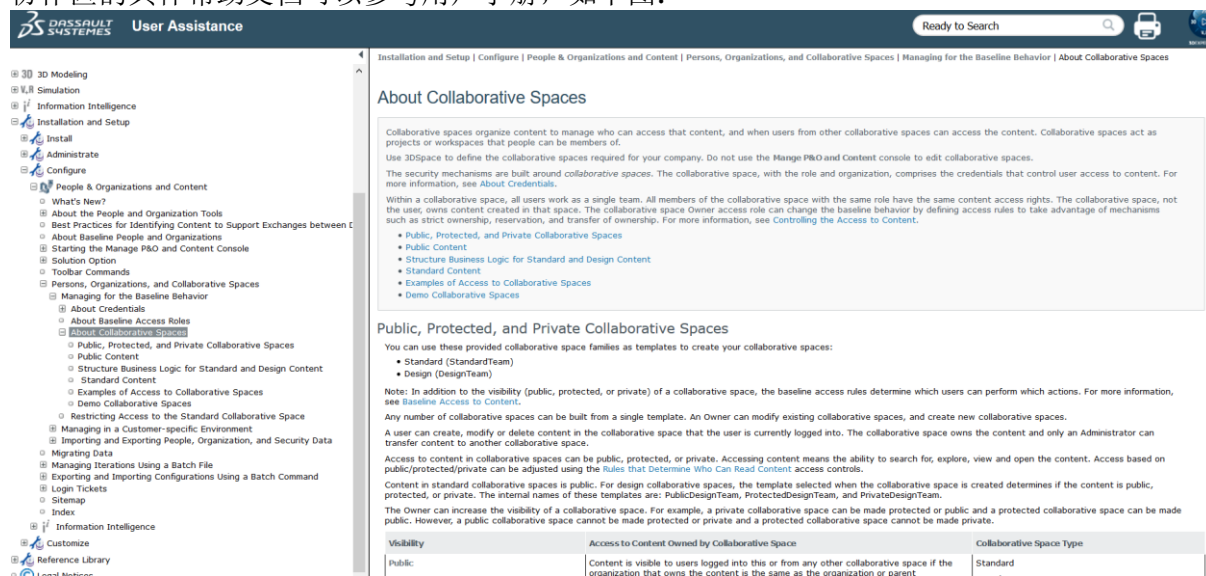


The dialog box titled "Select Credentials:" contains three input fields: "Organization" with a text box labeled "Company Name", "Collaborative Space" with a dropdown menu showing "Common Space", and "Role" with a dropdown menu showing "Leader". At the bottom are "OK" and "Cancel" buttons.

这个凭据的选择，就决定了当前用户可以在当前选择的身份下，做什么操作以及对哪些数据有什么样的操作权限。凭据包含了三部分信息：组织（Organization）、协作区（Collaborative Space）和角色（Role）。

组织基本就是根据客户的组织架构进行创建和进行人员的归属设置，这里就不展开介绍了。

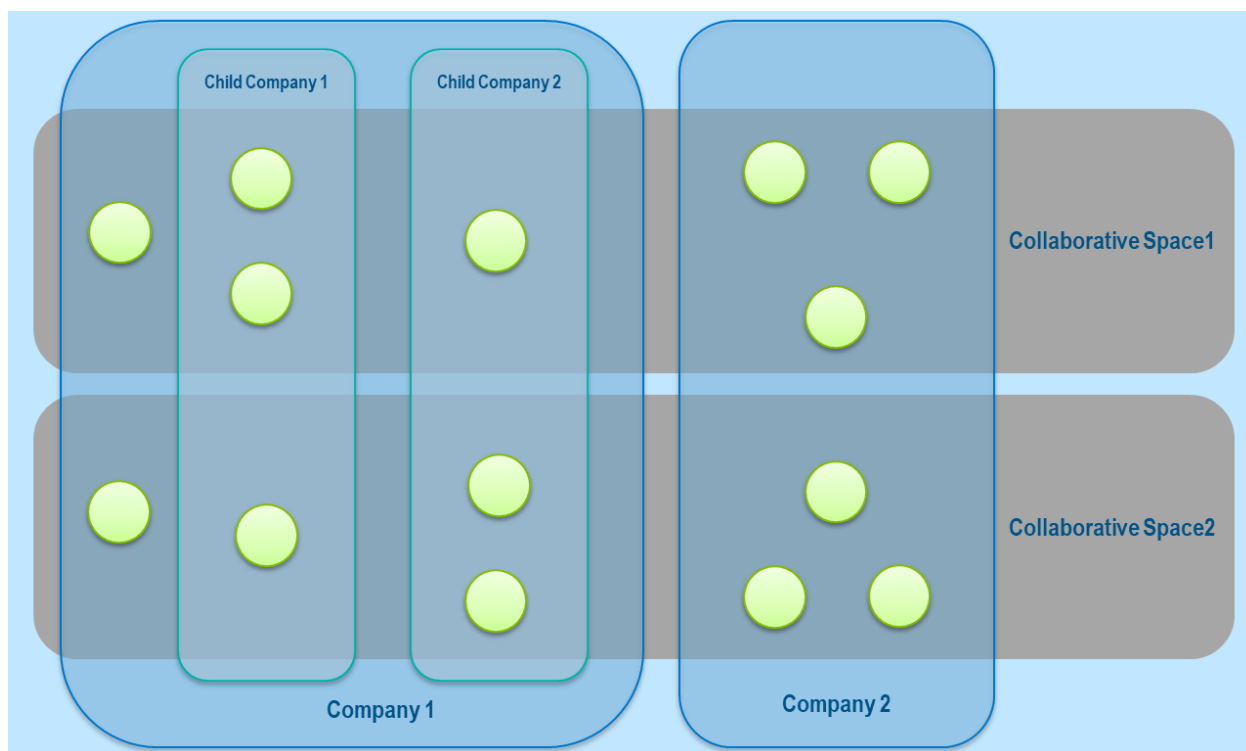
协作区的具体帮助文档可以参考用户手册，如下图：



The screenshot shows the "User Assistance" window with a search bar and a navigation pane on the left. The main content area is titled "About Collaborative Spaces" and contains text explaining collaborative spaces and their access rules. Below this, there is a section titled "Public, Protected, and Private Collaborative Spaces" with a table summarizing the visibility and access rules for different collaborative space types.

Visibility	Access to Content Owned by Collaborative Space	Collaborative Space Type
Public	Content is visible to users logged into this or from any other collaborative space if the organization that owns the content is the same as the organization or parent	Standard

组织和协作区最大的作用是通过二者的结合使用，将系统里面的所有对象进行了一定的归属划分，如下图所示例：



对于单个的对象，每个对象默认都会有 **organization** 和 **project** 两个基础属性信息，如下图所示例：

```

MQL<29>print bus 55624.58181.47588.16388 !history;
BusinessObject Document testAccessDoc-1133525410 A
  lattice eService Production
  policy Document Release
  description
  created 1/20/2021 1:05:25 AM
  modified 1/20/2021 5:49:51 PM
  owner lz5
  organization Company Name
  project privateDocTest
  ownership businessobject 55624.58181.7056.50948 as all
  unlocked
  locking not enforced
  Title testAccessDoc
  
```

Organization 就是对应的组织，**project** 就是对应的协作区，对象的这两个属性的来源是，当用户创建对象时，系统会抓取当前用户的凭据信息里面的组织和协作区信息，然后填写到当前用户创建的对象上面的 **organization** 和 **project** 属性上。

用户凭据的第三部分角色，在前面两个部分组织和部门圈定的对象范围，角色就表明了对于圈定的对象，我们根据当前的角色，可以进行什么样的数据操作。角色大的分类为：**Reader**、**Contributor**、**Author**、**Leader**、**Owner** 和 **Administrator**，关于角色的介绍文档，我们可以参考用户手册，如下图：

User Assistance

Ready to Search

3D Modeling
V, R Simulation
Information Intelligence
Installation and Setup
Install
Administrate
Configure
People & Organizations and Content
What's New?
About the People and Organization Tools
Best Practices for Identifying Content to Support Exchanges between
About Baseline People and Organizations
Starting the Manage P&O and Content Console
Solution Option
Toolbar Commands
Persons, Organizations, and Collaborative Spaces
Managing for the Baseline Behavior
About Credentials
About Baseline Access Roles
About Collaborative Spaces
Public, Protected, and Private Collaborative Spaces
Public Content
Structure Business Logic for Standard and Design Content
Standard Content
Examples of Access to Collaborative Spaces
Demo Collaborative Spaces
Restricting Access to the Standard Collaborative Space
Managing in a Customer-specific Environment
Importing and Exporting People, Organization, and Security Data
Migrating Data
Managing Iterations Using a Batch File
Exporting and Importing Configurations Using a Batch Command
Login Tickets
Sitemap
Index
Information Intelligence
Customize
Reference Library
Legal Notices

About Baseline Access Roles

Access roles determine what content a user can access, and what operations the user can perform. You cannot edit roles.

See Also

- In Other Guides
- Content Categories
- Controlling the Access to Content

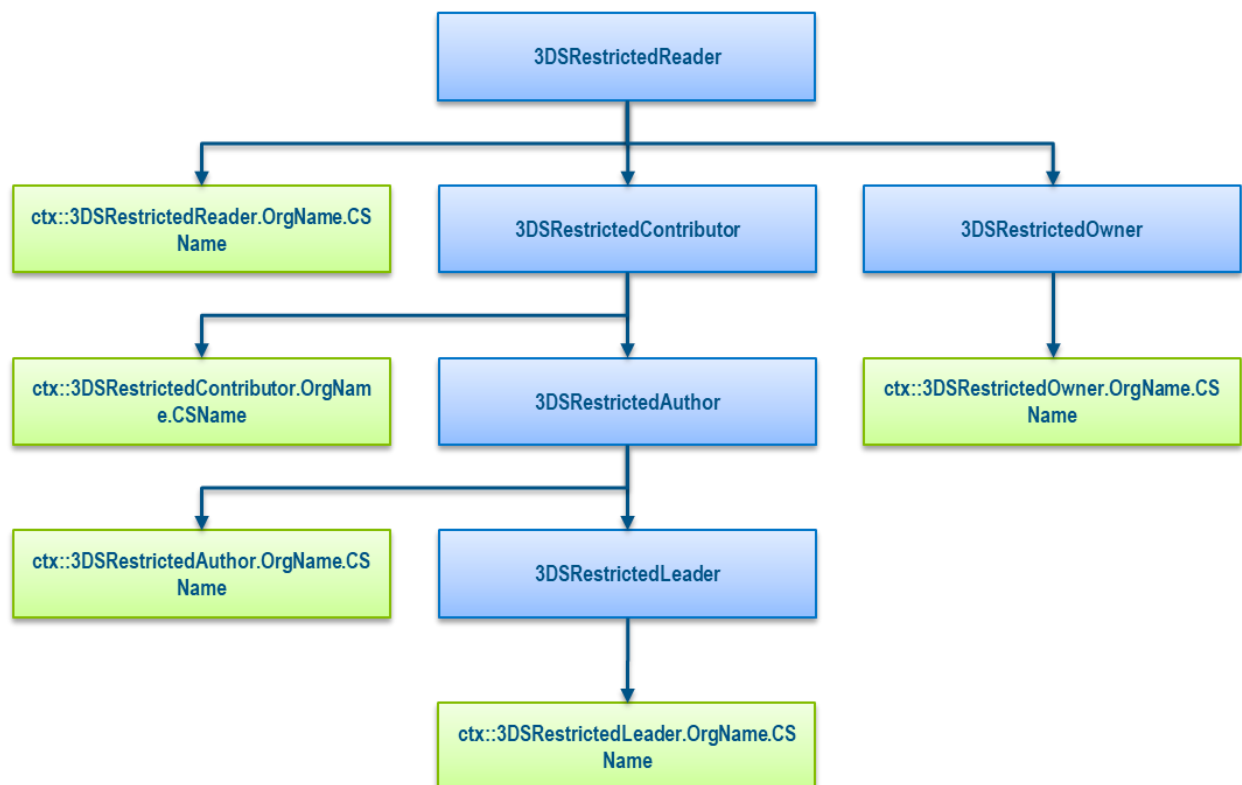
Combined with the collaborative space and organization, the role determines what content a user can access. See [About Credentials](#) for more information. For more information about app-specific accesses, see the User Guide for the apps.

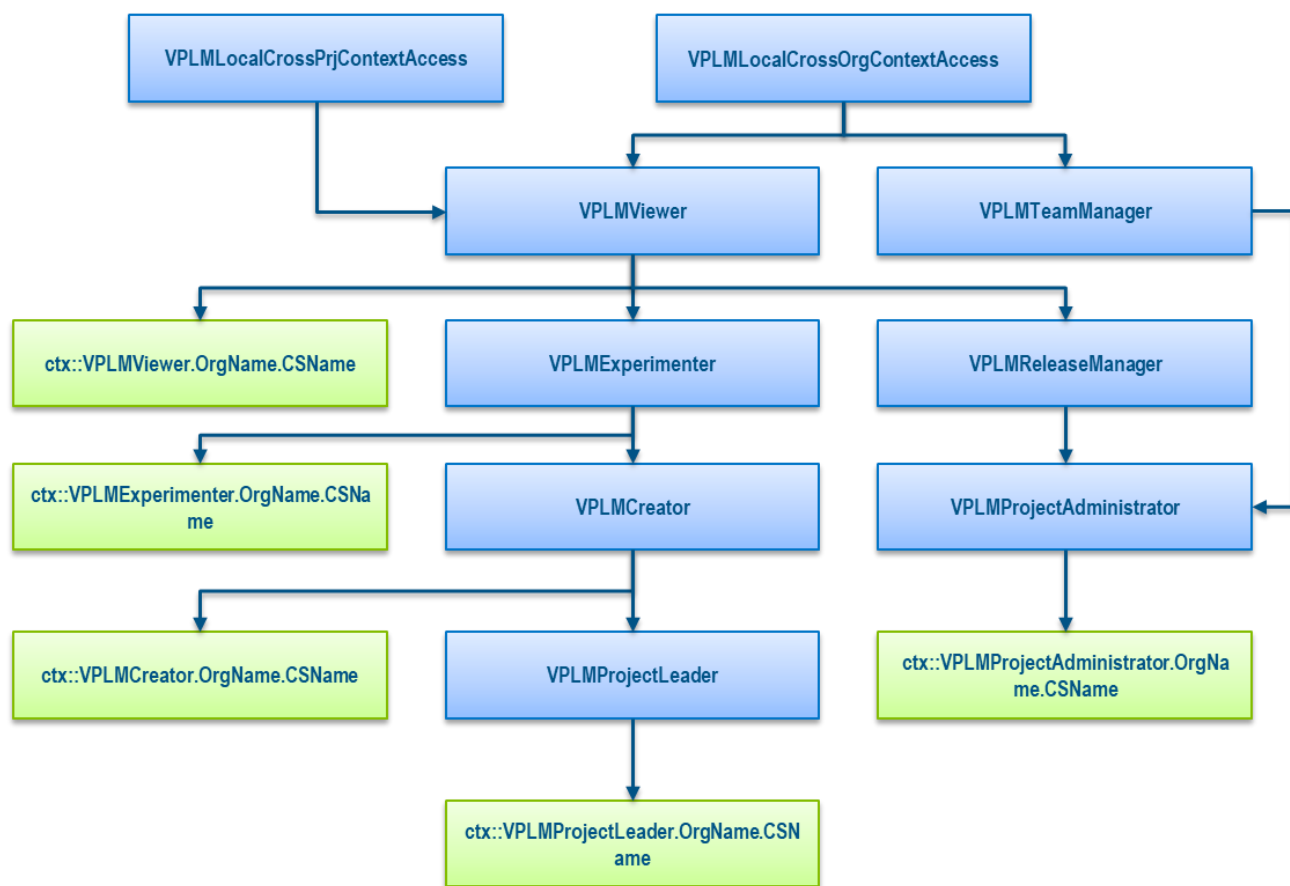
Accessing content means the ability to search for, explore, view and open the content. The role also defines the type of content a user can create or delete. Role accesses build hierarchically. For example, the Reader role can view content, but cannot create or evaluate content. The next role, Contributor, contains the Reader access plus the ability to evaluate content. The next role, Author includes all the access of the Reader and Contributor roles, plus the ability to create content. Accesses accumulate in this way for the Leader, Owner, and Administrator roles, except that the Owner and Administrator cannot create content.

This table describes the access rights for regular tasks.

Access Role for Regular Tasks	Description
Reader (VPLRViewer)	<ul style="list-style-type: none"> Read any content of collaborative spaces and organizations the user is assigned to. Create Personal Management content (such as Favorites, Personal Folders, and so on). This kind of content is only accessible by the content owner.
Contributor (VPLRContributor)	<ul style="list-style-type: none"> Inherits all Reader access rights. Create Evaluation content (such as Review, Simulation, and so on).
Author (VPLRCreator)	<ul style="list-style-type: none"> Inherits all Contributor access rights. Create Definition content (such as Requirements, Physical Product, XCAD, EBOM Part, System, and so on).
Leader (VPLRProjectLeader)	<ul style="list-style-type: none"> Inherits all Author access rights. Create Design Resource content (such as Library, Project Template, and so on).
Reader (Restricted) 3DSRestrictedReader	Used to restrict read access to content owned by the collaborative space AND organization. Within these restrictions, the user can perform the same actions as a regular Reader.
Contributor (Restricted) 3DSRestrictedContributor	Used to restrict read access to content owned by the collaborative space AND organization. Within these restrictions, user can perform the same actions as a regular Contributor.
Author (Restricted) 3DSRestrictedAuthor	Used to restrict read access to content owned by the collaborative space AND organization. Within these restrictions, user can perform the same actions as a regular Author.
Leader (Restricted) 3DSRestrictedLeader	Used to restrict read access to content owned by the collaborative space AND organization. Within these restrictions, user can perform the same actions as a regular Leader.

从文档上，我们可以知道这些角色大致分为 **Restricted** 和非 **Restricted** 两类，并且两类对应这不同的后台定义角色，从文档上我们可以得知名称与后台实际定义角色名的一个对应，但是各个角色之间的关联关系如何？我下面整理了两张权限关系表。





我们可以从文档，以及我提供的这两张权限对应表，得出几个角色之间的权限叠加关系。

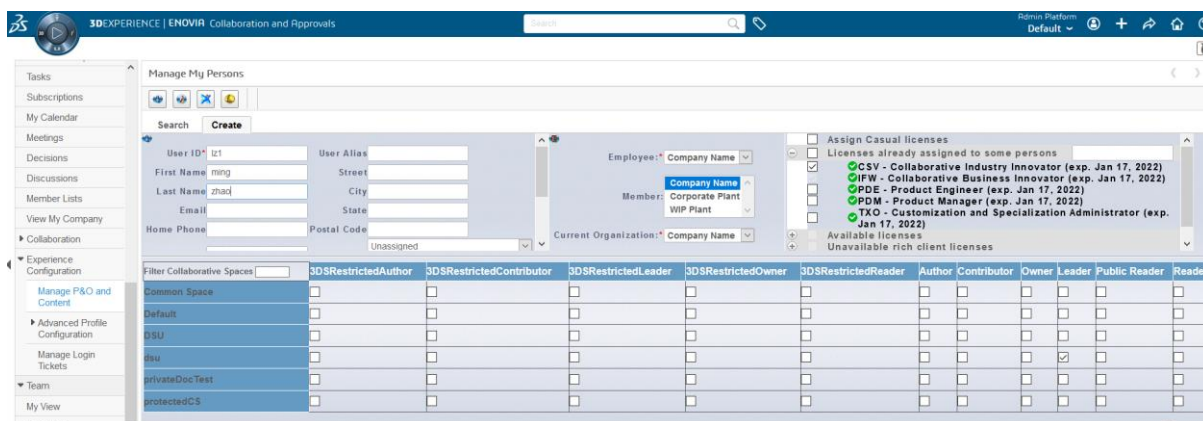
从上面的权限继承图，我们可以看到有几个底端权限名称为：

`ctx::VPLMProjectLeader.OrgName.CSName`，这种命名的 `role` 是什么含义呢？这块我们将在下面的章节，用户默认权限的部分介绍。

3. 用户默认权限

我们要对用户的权限进行客制化，那么我们需要知道，当我们创建一个新用户，新用户有哪些默认权限呢？我们下面进行一些操作来获取我们想要知道的默认权限。

我们实际创建一个，如下图：



在上图示例里面，我们创建了一个名为 **lz1** 的用户，这个用户的所属组织为默认的组织“**Company Name**”，并且将一个名为“**dsu**”的协作区的“**Leader**”的角色分配给 **lz1** 这个用户，当这个用户创建成功后，我们用 **MQL** 语句打印当前这个新创建的用户信息，如下图：

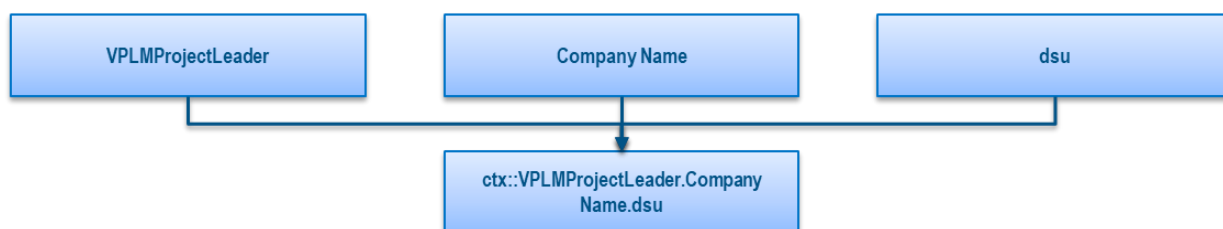
```
MQL<34>print person lz1 !history;
person lz1
  email
  fullname ming zhao
  phone
  fax
  address
  comment
  password <RESTRICTED>
  lattice eService Production
  access all
  type application,full
  admin none
  enable email
  enable iconmail
  assign role Grant.Company Name.lz1_PRJ
  assign role ctx::VPLMProjectLeader.Company Name.dsu
  nothidden
  property Company Key value 00000000000000000001
  property preference_Currency value Dollar
  created 1/20/2021 11:28:30 PM
  modified 1/20/2021 11:28:33 PM
MQL<35>
```

从上图我们可以看到，用户有两个默认被指派的 **role**，“**Grant.Company Name.lz1_PRJ**”和“**ctx::VPLMProjectLeader.Company Name.dsu**”。

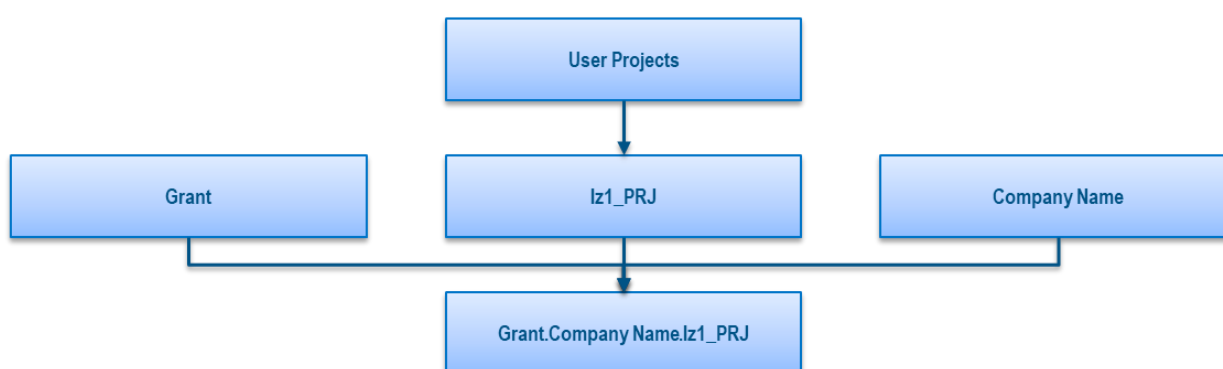
在解析这两个默认的 **role** 之前，我们先说一下系统的一个默认的机制，当我们创建一个新组织或者一个新的协作区后，系统内部会自动创建一个相应名字的 **role** 与我们创建的组织或者协作区对应。

我们先看“**ctx::VPLMProjectLeader.Company Name.dsu**”这个 **role**，从写法结构上，我们可以看出这个 **role** 包含了三部分内容：角色“**VPLMProjectLeader**”、组织“**Company Name**”和协作区“**dsu**”，我们通过 **MQL** 可以得到的信息为，当前的这个 **role** 同时继承于“**VPLMProjectLeader**”、“**Company Name**”和“**dsu**”这三个 **role**，这与前面我们创建用户填写的信息用户所属组织为“**Company Name**”，分配了协作区“**dsu**”的“**Leader**”角色

这三个关键信息是对应的（“Leader”角色对应后台的角色就是“VPLMProjectLeader”）。



我们再看“Grant.Company Name.lz1_PRJ”这个 role，我往上追踪了一下这个 role 的父 role，得到结果如下图：



这里需要着重点出的是“lz1_PRJ”，当前的 role 是用户对象一旦创建，会随着用户的创建而生成，继承于“User Projects”而生的一个 role。

4. 3DE 相关的一些权限概念与机制

用户的权限，从获取的途径来划分，我个人将其大致分为两种：固定权限和临时权限。固定权限是对象一旦创建，权限项是立即定义好生效的，在当前对象的属性和状态不发生变化的情况下，固有权项是不可增加和减少的；临时权限是对象创建后，本身所不具备的权限，是可以通过权限分配的方式给对象授予一定的操作权限，并且还可以通过反向取消的方式将用户所拥有的权限给去除掉。

固定权限的设置，我们是在 policy 上的各个 state 上面进行设定，与系统里面的人员角色想关联对应；

临时权限的设置，在 3DE 目前推荐的做法是通过 ownership 的设定来完成，下面我们来描述一下 ownership 的概念。

Ownership 在系统里面分成两种：Primary Ownership Vector（POV）和 Secondary Ownership Vector（SOV），在官方的有些文档中，也将 POV 叫做 Direct Ownership Vector（DOV），后面如果大家看到 POV 和 DOV，从概念上了解这是同一个东西即可。

首先，我们先来了解一下什么是 POV，在前面我们介绍过每个对象默认都会有 organization 和 project 两个基础属性信息，如下图：

```

MQL<29>print bus 55624.58181.47588.16388 !history;
BusinessObject Document testAccessDoc-1133525410 A
  lattice eService Production
  policy Document Release
  description
  created 1/20/2021 1:05:25 AM
  modified 1/20/2021 5:49:51 PM
  owner lz5
  organization Company Name
  project privateDocTest
  ownership businessobject 55624.58181.7056.50948 as all
  unlocked
  locking not enforced
  Title testAccessDoc
  
```

这两个属性信息合起来就是当前对象的 POV 信息，POV 简单来说，就是描述当前对象所属的组织和合作区。

这个 POV 信息是可以变更的，系统 OOTB 很多位置（比如：BookMark Editor）就提供了变更用户 POV 信息的功能，如下图：



当前，想进行 POV 信息的变更，也是需要当前用户对于当前的对象具有一定的操作权限才可以，在系统中，这个权限就是 changeowner 权限。

然后，我们先来了解一下什么是 SOV，系统里面的对象，很多都有 ownership 这个属性，如下图：

```
SQL>print bus 55624.58181.7056.50948 !history;
BusinessObject Workspace Vault testFolder 2996B056000026386007F22900000775
  lattice eService Production
  policy Workspace Vaults
  description testFolder
  created 1/20/2021 1:04:57 AM
  modified 1/20/2021 1:05:25 AM
  owner lz5
  organization Company Name
  project privateDocTest
  ownership businessobject 55624.58181.62558.28307 as all
  ownership - lz5_PRJ for Multiple Ownership For Object as read,modify,delete,checkout,checkin,lock,unlock,changeowner,
  promote,demote,revise,majorrevise,changename,changetype,fromconnect,toconnect,fromdisconnect,todisconnect,show,reserve,u
  nreserve,changesov
  unlocked
```

这个 ownership 主要是用来进行权限的临时赋予的操作，也可以用于对象之间的权限继承，这是在新版 3DE 中，目前主要的临时权限赋予机制，用户 ownership 的变更操作，需要当前用户对于当前的对象具有一定的操作权限才可以，在系统中，这个权限就是 changesov 权限。

提到临时权限赋予机制，就不得不提 grant 这种权限赋予方式，在 enovia 早期版本中，临时权限的赋予都是通过这种方式，原理是在具体的对象中，创建一张 grant 表，记录当前对象和相应用户对应的权限设定，如下图：

```
SQL>print bus Document testAccessDoc-1133525410 A;
BusinessObject Document testAccessDoc-1133525410 A
  lattice eService Production
  policy Document Release
  description
  created 1/20/2021 1:05:25 AM
  modified 1/26/2021 12:16:58 AM
  owner lz5
  organization Company Name
  project privateDocTest
  ownership businessobject 55624.58181.7056.50948 as all
  grantee hh7
  grantor creator
  granteeaccess read,show
  granteesignature false
  unlocked
  locking not enforced
```

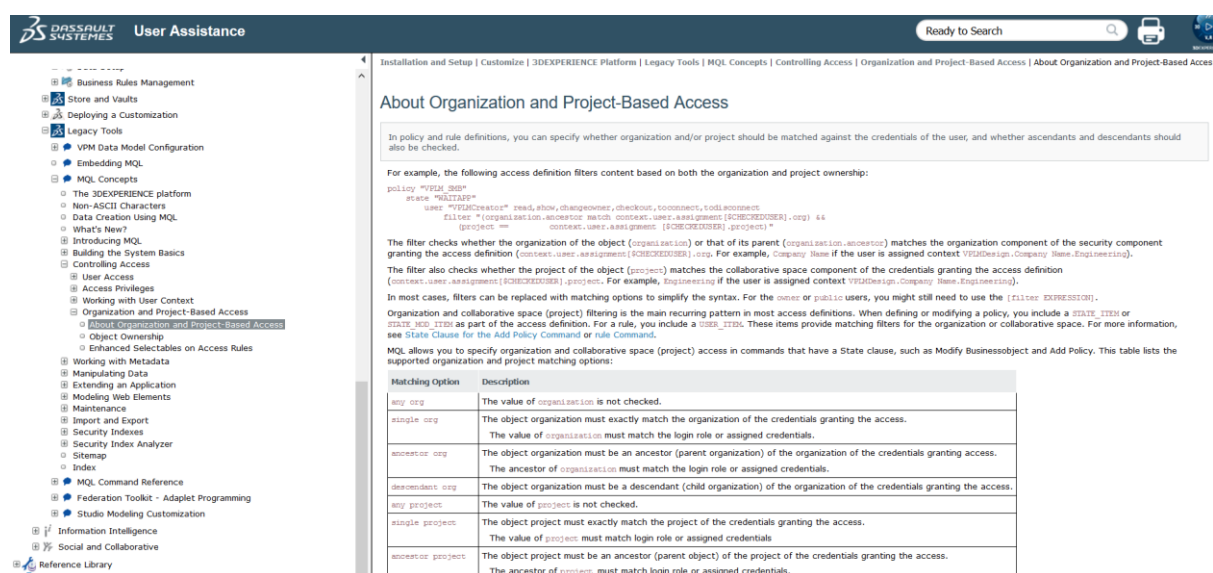
上面的信息表明，当前对象被 creator 这个超级管理员，给用户 hh7 赋予了对当前对象的 read, show 的权限，给用户赋予 grant 权限的操作，也需要赋予者在对象上具有一定的操作权限才可以，在系统中，这个权限是 grant 权限。

注意：grant 这个权限赋予机制虽然还存在于系统中，但是随着更强大的 ownership 的权限赋予机制的崛起，现在已经越来越少的使用了。

5. 固定权限设定示例解析

下面我们就通过 OOTB 的 Document 对象的权限设定的解析，来学习 3DE 的权限设定机制。

关于权限这块的资料，我们可以从 user guide 和 3DU 的“3DEXPERIENCE Customization Advanced”这两块获取资料，user guide 的位置如下图：



3DU 的“3DEXPERIENCE Customization Advanced”这门课程可以获取的文档资料为“3DEXPERIENCE Customization Advanced R2019x.pdf”。

首先，我们可以从资料上得到，协作区关于文档对象的权限设置机制如下图：

State of Maturity					
Private	In Work	Frozen	Released	Obsolete	
					"Public" Collaborative Space
					"Protected" Collaborative Space
					"Private" Collaborative Space

Legend	
	Personal Data
	Private Data
	Public Data

从上图我们可以得知，当文档处于“In Work”状态的时候，如果文档所属的协作区是 public 的，那么所有人可见，否则只有处于同一协作区的人员可见；当文档处于“Released”状态的时候，如果当前文档所属的协作区是 private 的，那么只有处于同一协作区的人员可见，否则就全体人员可见。

文档的默认 policy 为 “Document Release”，我们使用 MQL 命令 “print policy “Document Release” !history”，打印出当前 policy 的全部信息，我们先截取 state 为 “In Work” 下面的全部权限设定，如下图：

```
state IN WORK
minorrevisionable true
majorrevisionable true
versionable true
promote false
checkout history true
published false
public none branch FROZEN
login owner key VPLMstrictOrgOwnershipIsTRUE promote,demote single organization single project
login owner key VPLMstrictOrgOwnershipIsFALSE promote,demote single project filter (!expression[VPLMstrictOrgOwnership])
owner none branch FROZEN
login user VPLMCreator key CollaborativeAuthoring VPLMstrictOrgOwnershipIsTRUE2 modify,checkin,lock,changeName single organization single project context owner
login user VPLMCreator key ExtendedChangeAccessForCreator changeVault,changePolicy,changeType single organization single project context owner
login user 3DSRestrictedAuthor key ExtendedDisconnectAccess fromDisconnect single organization single project
login user VPLMCreator key ExtendedDisconnectAccessSingleOrg fromDisconnect,todisconnect single organization single project context owner
login user VPLMProjectAdministrator key ExtendedModifyAccessForPrjAdmin modify,checkin,lock,changeName single project
login user 3DSRestrictedLeader key ExtendedSingleCtxDeleteAccess delete single organization single project inclusive reserve rule DefinitionContentRestrictedInWorkState ruleuser 3DSRestrictedLeader rulekey "ExtendedSingleCtxDeleteAccess"
login user 3DSRestrictedAuthor key ExtendedSingleCtxDeleteAccess_OwningContent delete single organization single project context owner inclusive reserve rule DefinitionContentRestrictedInWorkState ruleuser 3DSRestrictedAuthor rulekey "ExtendedSingleCtxDeleteAccess_OwningContent"
login user VPLMCreator key ExtendedSingleCtxDeleteAccess_SingleOrgSinglePrj_OwningContent delete single organization single project context owner
login user 3DSRestrictedLeader key ExtendedSingleCtxModifyAccess modify,checkin,changeName,changeType single organization single project inclusive reserve rule DefinitionContentRestrictedInWorkState ruleuser 3DSRestrictedLeader rulekey "ExtendedSingleCtxModifyAccess"
login user 3DSRestrictedOwner key ExtendedSingleCtxModifyAccess modify,checkin,changeName,changeType single organization single project inclusive reserve rule DefinitionContentRestrictedInWorkState ruleuser 3DSRestrictedOwner rulekey "ExtendedSingleCtxModifyAccess"
login user 3DSRestrictedAuthor key ExtendedSingleCtxModifyAccess_OwningContent modify,checkin,changeName,changeType single organization single project context owner inclusive reserve rule DefinitionContentRestrictedInWorkState ruleuser 3DSRestrictedAuthor rulekey "ExtendedSingleCtxModifyAccess_OwningContent"
login user 3DSRestrictedLeader key ExtendedSingleCtxPromoteAccess promote,demote single organization single project inclusive reserve rule DefinitionContentRestrictedInWorkState ruleuser 3DSRestrictedLeader rulekey "ExtendedSingleCtxPromoteAccess"
login user 3DSRestrictedAuthor key ExtendedSingleCtxPromoteAccess_OwningContent promote,demote single organization single project context owner inclusive reserve rule DefinitionContentRestrictedInWorkState ruleuser 3DSRestrictedAuthor rulekey "ExtendedSingleCtxPromoteAccess_OwningContent"
login user VPLMLocalCrossOrgContextAccess key ExtendedSingleCtxPublicReadAccess_AnyOrgSinglePrj read,checkout,tocconnect,todisconnect,show single project public maturity
login user 3DSRestrictedAuthor key ExtendedSingleCtxReadAccess read,checkout,tocconnect,todisconnect,show single organization single project rule DefinitionContentRestrictedInWorkState ruleuser 3DSRestrictedAuthor rulekey "ExtendedSingleCtxReadAccess"
login user 3DSRestrictedOwner key ExtendedSingleCtxReadAccess read,checkout,tocconnect,todisconnect,show single organization single project rule DefinitionContentRestrictedInWorkState ruleuser 3DSRestrictedOwner rulekey "ExtendedSingleCtxReadAccess"
login user VPLMCreator key ExtendedSingleCtxReadAccess_AnyOrgSinglePrj read,checkout,tocconnect,todisconnect,show single project
login user VPLMProjectAdministrator key ExtendedSingleCtxReadAccess_AnyOrgSinglePrj read,checkout,tocconnect,todisconnect,show single project
login user 3DSRestrictedLeader key ExtendedSingleCtxReserveAccess lock,reserve single organization single project inclusive reserve rule DefinitionContentRestrictedInWorkState ruleuser 3DSRestrictedLeader rulekey "ExtendedSingleCtxReserveAccess"
```

这里我们可以看到，user 的设定有两种，login user 和 user，login user 是要获取当前用户的凭据里面的组织和协作区信息进行验证的，比方说，在协作区下的 reader 和 leader 两个角色，OOTB 的权限设置原则是：如果用户拥有 leader 角色，那只有当用户的凭据切换到 leader 角色，才可以将相应的组织和协作区下的文档进行发布操作，如果不将凭据切换到 leader 角色，那么就不具备将相应文档进行发布的权限；而如果用户拥有 reader 角色，那么不管用户是否将当前的凭据切换到 reader 角色，用户都具有访问相应的组织和协作区下面文档的权限。这二者的差异就是我们在 policy 进行 leader 权限设定的时候，加上了 login 前缀限制，而给 reader 权限设定的时候就是普通的 user 权限设定。

5.1. Policy 权限设置的一些新参数

Flag	Description
[any single ancestor descendant] organization	Specifies that the object's organization owner must be checked against the context's member organization.
[any single ancestor descendant] project	Specifies that the object's organization owner must be checked against the context's member project.
any owner	No check is performed on owner (default).
context owner	Checks that object is owned by context user.
any reserve	No check is performed on whether the object is reserved (default).
no reserve	Checks that the object is not reserved by anyone.
context reserve	Checks that the object is reserved by the current context user.
inclusive reserve	Checks that the object is either reserved by the current context user or by no one.

Flag	Description
any maturity	No check is performed on the maturity of the project owning the object (default).
public maturity	Checks that maturity on security context project is public.
protected maturity	Checks that maturity on security context project is protected.
private maturity	Checks that maturity on security context project is private.
notprivate maturity	Checks that maturity on security context project is either protected or public.
ppp maturity	Checks that maturity on security context project is either private, protected, or public.
[any oem goldpartner partner supplier customer contractor] category	Checks that the current context or current login security context is flagged with the corresponding property.
[filter localfilter] EXPR	A filter expression defined for the user access. Use localfilter instead of filter in policies and access rules to return only results to which the current user definitely has access. When you use localfilter, the expression is not evaluated by full-text search.
branch STATE_NAME	Indicates the state to move to when the signature is provided. You can include a filter with this keyword. For example: branch FROZEN filter TRUE

5.2. In Work 状态权限设置解析

我们将 “In Work” 状态下面，具有 read、show 权限设定的项提取出来，得到权限设置项如下：

```
login user VPLMLocalCrossOrgContextAccess key
ExtendedSingleCtxPublicReadAccess_AnyOrgSinglePrj
read,checkout,toconnect,todisconnect,show single project public maturity

login user VPLMCreator key ExtendedSingleCtxReadAccess_AnyOrgSinglePrj
read,checkout,toconnect,todisconnect,show single project

login user VPLMProjectAdministrator key ExtendedSingleCtxReadAccess_AnyOrgSinglePrj
read,checkout,toconnect,todisconnect,show single project

user VPLMSecuredCrossAccess key ExtendedReadAccessForPublicGuest
read,checkout,toconnect,todisconnect,show single organization single project public maturity

user VPLMLocalCrossOrgContextAccess key
ExtendedMultiCtxPrivateReadAccess_AnyOrgSinglePrj read,checkout,show single project
filter expression[ExtendViewerOnInWork] &&
expression[MultiContextualPrivateReadAccess]

user VPLMCreator key ExtendedMultiCtxReadAccess_AnyOrgSinglePrj read,checkout,show
single project filter expression[MultiContextualPrivateReadAccess]

user VPLMProjectAdministrator key ExtendedMultiCtxReadAccess_AnyOrgSinglePrj
read,checkout,show single project filter expression[MultiContextualPrivateReadAccess]
```



```
user VPLMLocalCrossPrjContextAccess key ExtendedPublicReadAccess
read,checkout,toconnect,todisconnect,show ancestor organization public maturity filter
expression[OrganizationPublicAccess]
```

我们再根据权限设置原则，只要用户具备相应的协作区的 **reader** 角色，那么不管当前凭据是否设定了 **reader** 角色，都对文档有可见权限这个机制，我们可以进一步把所有 **login user** 的权限设置项去掉，得到权限设置项如下：

```
user VPLMSecuredCrossAccess key ExtendedReadAccessForPublicGuest
read,checkout,toconnect,todisconnect,show single organization single project public maturity
```

```
user VPLMLocalCrossOrgContextAccess key
ExtendedMultiCtxPrivateReadAccess_AnyOrgSinglePrj read,checkout,show single project
filter expression[ExtendViewerOnInWork] &&
expression[MultiContextualPrivateReadAccess]
```

```
user VPLMCreator key ExtendedMultiCtxReadAccess_AnyOrgSinglePrj read,checkout,show
single project filter expression[MultiContextualPrivateReadAccess]
```

```
user VPLMProjectAdministrator key ExtendedMultiCtxReadAccess_AnyOrgSinglePrj
read,checkout,show single project filter expression[MultiContextualPrivateReadAccess]
```

```
user VPLMLocalCrossPrjContextAccess key ExtendedPublicReadAccess
read,checkout,toconnect,todisconnect,show ancestor organization public maturity filter
expression[OrganizationPublicAccess]
```

还是根据只要用户具备相应的协作区的 **reader** 角色，那么不管当前凭据是否设定了 **reader** 角色，都对文档有可见权限这个机制，我们把 **reader** 所对应的“**VPLMViewer**”的所有子 **role** 的权限都去掉，只保留“**VPLMViewer**”和其父角色的权限设置项，那么就得到权限设置项如下：

```
user VPLMLocalCrossOrgContextAccess key
ExtendedMultiCtxPrivateReadAccess_AnyOrgSinglePrj read,checkout,show single project
filter expression[ExtendViewerOnInWork] &&
expression[MultiContextualPrivateReadAccess]
```

```
user VPLMLocalCrossPrjContextAccess key ExtendedPublicReadAccess
read,checkout,toconnect,todisconnect,show ancestor organization public maturity filter
expression[OrganizationPublicAccess]
```

先通过 **MQL** 语句去查询 **filter** 里面的三个 **expression** “**ExtendViewerOnInWork**”、
“**MultiContextualPrivateReadAccess**”和“**OrganizationPublicAccess**”，三个表达式的 **value** 都为 **true**，另外我们也可以登录 **dashboard** 界面去查看，如下图：

Access Rules

Allows to define the rules on who can access what content

Read Operations				
Allow read access to in work content in protected or private collaborative spaces.	?	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	✓
Allow read access to any public content.	?	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	✓
Allow users read access to any content in any other collaborative space.	?	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	✓

这三个 **express** 实际上就对应这三个 **Read Operation** 的值，选中则代表表达式的值为 **true**，未选中则代表表达式的值为 **false**。

因为三个表达式的值都为 **true**，也就是说上面两个权限设定项都是起作用的，那么我们下面对这两项权限设定项进行深度解析。

第一项：从后面设定 **single project** 我们可以看出，只要用户设定的协作区和当前文档是同一协作区，就都具备了可见权限，这与我们前面从文档获取的权限设定情况相符：同一协作区的“**In Work**”状态文档对应具备同一协作区的 **reader** 权限的人员都是可见的这个权限设置是相符的；

第二项：从后面设定 **ancestor organization public maturity** 我们可以看出，如果文档是放在 **public** 的协作区里，只要用户所属组织是文档所属的组织（或者其组织的父组织），那么用户对该文档都是可见的，这也与我们前面从文档获取的权限设定情况相符：只要当前文档处于 **public** 的协作区里，同一组织下，不管用户具备那个协作区的 **reader** 权限，当前文档对于用户都是可见的；

5.3. Release 状态权限设置解析

我们将“**Release**”状态下面，与 **reader** 角色有继承关联的 **role**，并且具有 **read**、**show** 权限设定的项提取出来，重复上面的将 **login user** 的权限设置去掉，将非 **reader** 所对应的权限设置的 **role** 去掉，得到权限设置项如下：

```
user VPLMLocalCrossOrgContextAccess key  
ExtendedMultiCtxReadAccess_AnyOrgSinglePrj read,checkout,show single project filter  
expression[MultiContextualPrivateReadAccess]  
  
user VPLMLocalCrossPrjContextAccess key ExtendedPublicReadAccess  
read,checkout,toconnect,todisconnect,show ancestor organization notprivate maturity filter  
expression[OrganizationPublicAccess]
```

先通过 **MQL** 语句去查询 **filter** 里面的两个 **expression** “**MultiContextualPrivateReadAccess**”和“**OrganizationPublicAccess**”，两个表达式的 **value** 都为 **true**，也就是说两个权限设置项都是起作用的。

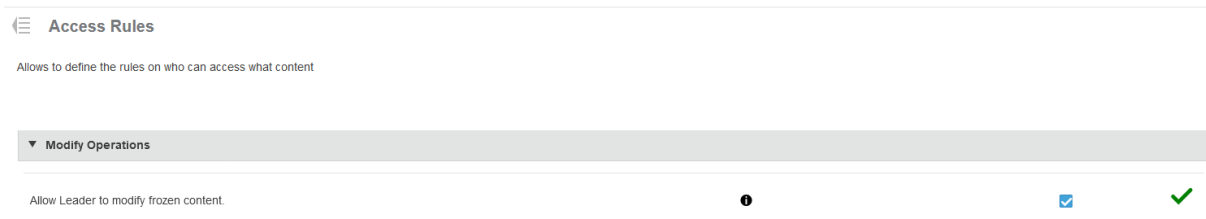
我们下面对这两项权限设定项进行深度解析。

第一项：从后面设定 **single project** 我们可以看出，只要用户设定的协作区和当前文档是同一协作区，就都具备了可见权限，这与我们前面从文档获取的权限设定情况相符：同一协作区的“**Release**”状态文档对应具备同一协作区的 **reader** 权限的人员都是可见的这个权限设置是相符的；

第二项：从后面设定 **ancestor organization notprivate maturity** 我们可以看出，如果文档是放在非 **private**（即 **public** 或者 **protected**）的协作区里，只要用户所属组织是文档所属的组织（或者其组织的父组织），那么用户对该文档都是可见的，这也与我们前面从文档获取的权限设定情况相符：只要当前文档处于非 **private**（即 **public** 或者 **protected**）的协作区里，同一组织下，不管用户具备那个协作区的 **reader** 权限，当前文档对于用户都是可见的；

5.4. FROZEN 状态权限设置解析

我们打开 **dashboard**，可以看到权限设置规则有这样的设置，如下图：



我们可以看到，在“**FROZEN**”状态，该条法则起作用的话，**Leader** 角色是具有修改协作区内其他用户的对象的权限的。

我们用 **MQL** 语句查询表达式“**AllowFrozenContentMod**”的值，当上面的规则生效的时候，值为 **true**，当上面的规则不生效的时候，值为 **false**。

我们创建数据进行测试，我在我的环境测试过程与结果如下：

1. 使用管理员账号使上面的规则生效；
2. 使用管理员账号给用户 **user1** 分配了一个 **private** 协作区 **cs001** 的 **Leader** 角色（**user1** 用户只有这一个凭据）；
3. 使用 **user1** 创建了一个文档对象 **Doc001**，并将其提升到“**FROZEN**”状态；
4. 使用管理员账号给用户 **user2** 分配了 **cs001** 的 **Reader** 角色；
5. **user2** 用户使用 **cs001** 的 **reader** 角色登录系统，可以搜索到 **Doc001** 对象，但是没有编辑权限；
6. 使用管理员账号给用户 **user2** 分配了 **cs001** 的 **Leader** 角色；
7. **user2** 用户使用 **cs001** 的 **reader** 角色登录系统，可以搜索到 **Doc001** 对象，可以编辑 **Doc001** 对象；

8. user2 用户使用 cs001 的 leader 角色登录系统，可以搜索到 Doc001 对象，可以编辑 Doc001 对象；
9. 使用管理员账号使上面的规则失效；
10. user2 用户使用 cs001 的 reader 角色登录系统，可以搜索到 Doc001 对象，但是没有编辑权限；
11. user2 用户使用 cs001 的 leader 角色登录系统，可以搜索到 Doc001 对象，但是没有编辑权限；

我们将“FROZEN”状态下面，与 Leader 角色有继承关联的 role，并且具有 modify 权限设置的项提取出来，得到权限设置项如下：

```
login user VPLMProjectLeader key TechnicalLeader_VPLMStrictOrgOwnershipIsFALSE
modify,checkin,lock,changename single project filter
(!expression[VPLMStrictOrgOwnership]) && expression[AllowFrozenContentMod]

login user VPLMProjectLeader key TechnicalLeader_VPLMStrictOrgOwnershipIsTRUE
modify,checkin,lock,changename single organization single project filter
expression[AllowFrozenContentMod]

user VPLMProjectLeader key ExtendedMultiCtxModifyAccess_AnyOrgSinglePrj
modify,checkin,lock,changename single project filter
(!expression[VPLMStrictOrgOwnership]) && expression[MultiContextualAuthoringAbility] &&
expression[AllowFrozenContentMod]

user VPLMProjectLeader key ExtendedMultiCtxModifyAccess_SingleOrgSinglePrj
modify,checkin,lock,changename single organization single project filter
expression[MultiContextualAuthoringAbility] && expression[AllowFrozenContentMod]
```

通过 MQL 查询这两个相关的 expression 的值，“VPLMStrictOrgOwnership”和“MultiContextualAuthoringAbility”，发现这两个表达式的值都为 true，则去掉因为这两个表达式而无法生效的权限设置项，过滤后的权限设置项为：

```
login user VPLMProjectLeader key TechnicalLeader_VPLMStrictOrgOwnershipIsTRUE
modify,checkin,lock,changename single organization single project filter
expression[AllowFrozenContentMod]

user VPLMProjectLeader key ExtendedMultiCtxModifyAccess_SingleOrgSinglePrj
modify,checkin,lock,changename single organization single project filter
expression[MultiContextualAuthoringAbility] && expression[AllowFrozenContentMod]
```

我们分析上面这两项权限设置项，可以得到结论如下：

首先，用户的权限设定可以通过表达式“AllowFrozenContentMod”的值来使上面的两条权限设定生效或者失效；其次，当上面两条权限生效时，如果用户所具有的凭据的组织和合作区，与当前对象所属的组织和合作区相同，并且该组织和合作区所具备的角色有 Leader 角色，无论用户当前使用的凭据是什么，都对当前的文档对象有修改权限。

这与我们上面的测试结果相同。

6. 临时赋权

为了能形象的说明临时赋权的机制，我们先创建了一组数据如下：

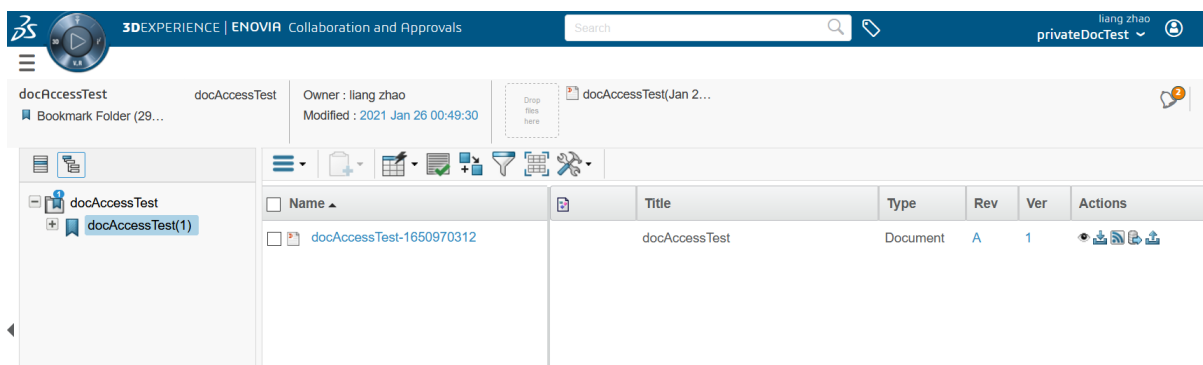
系统里面创建了两个用户 lz5 和 hh7，两个协作区 “privateDocTest” 和 “protectedCS”。

两个用户分别为两个不同协作区的 Leader 角色。

The screenshots show the 3DS user management interface. The top screenshot is for user 'lz5' and the bottom is for user 'hh7'. Both users are assigned the 'Leader' role in the 'privateDocTest' and 'protectedCS' collaborative spaces. The interface includes a form for user details, a list of roles, and a table for assigning roles to collaborative spaces.

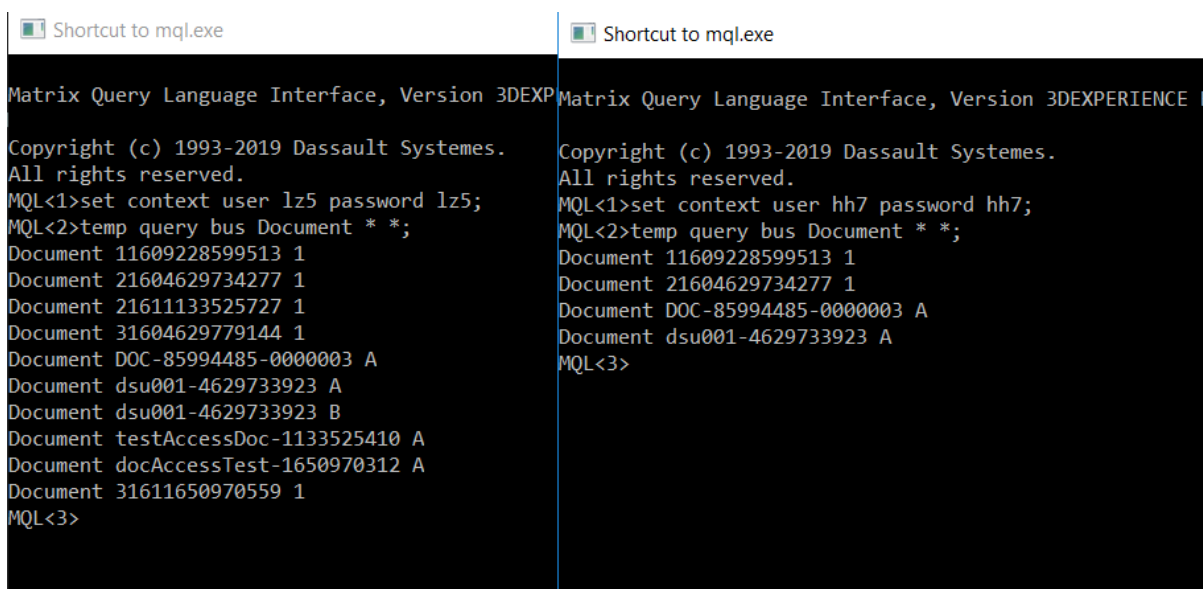
Filter Collaborative Spaces	3DSRestrictedAuthor	3DSRestrictedContributor	3DSRestrictedLeader	3DSRestrictedOwner	3DSRestrictedReader	Author	Contributor	Owner	Leader	Public Reader	Reader
Common Space	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Default	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
DSU	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
dsu	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
privateDocTest	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
protectedCS	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

用户 lz5 创建了一个名称为 “docAccessTest” 的 Bookmark Workspace，并且在当前的 workspace 下创建了一个同名的 “docAccessTest” 的文件夹，并在 “docAccessTest” 文件夹下，创建了一个名称为 “docAccessTest” 的文档对象。



根据 OOTB 的文档的可见权限原则定义，我们可以得出，当前的“docAccessTest”文档对于用户 hh7 是不可见的，我们通过搜索操作来验证当下的权限原则。

为了方便操作，我们采用 MQL 命令代替在浏览器页面进行查询操作，如下图：

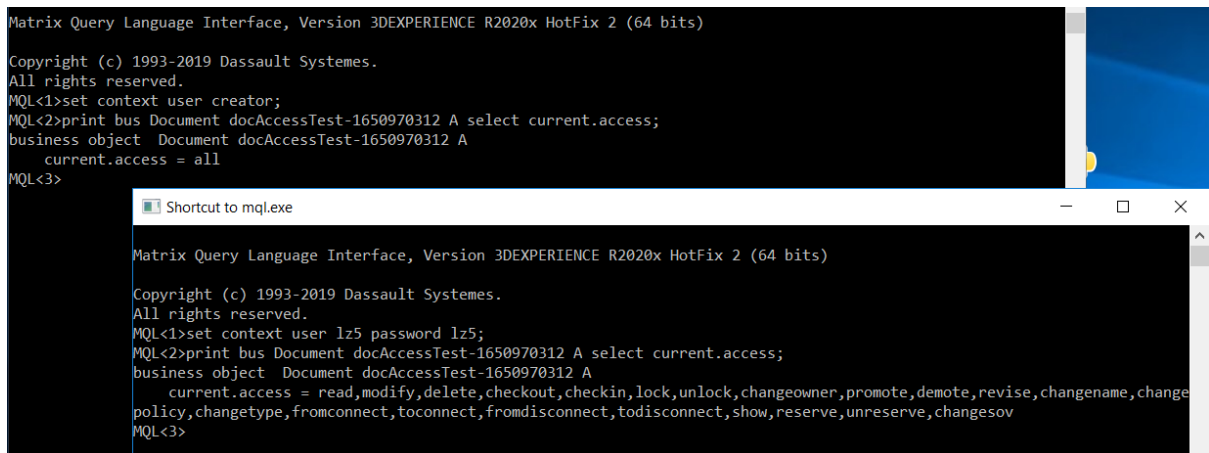


从上图我们可以看出，hh7 这个用户对于“testAccessDoc-1133525410”这个文档对象，没有可见权限。

下面我们就通过使用 grant 和 ownership 这两种设定方式给 hh7 这个用户赋予临时的 read、show 权限，让 hh7 具备对该文档的可见权限。

6.1. Grant 赋权模式

使用 grant 模式是需要进行 grant 操作的用户在当前对象上具备 grant 权限，我们可以通过 MQL 命令来查看用户在当前对象上是否具备 grant 权限，如下图：



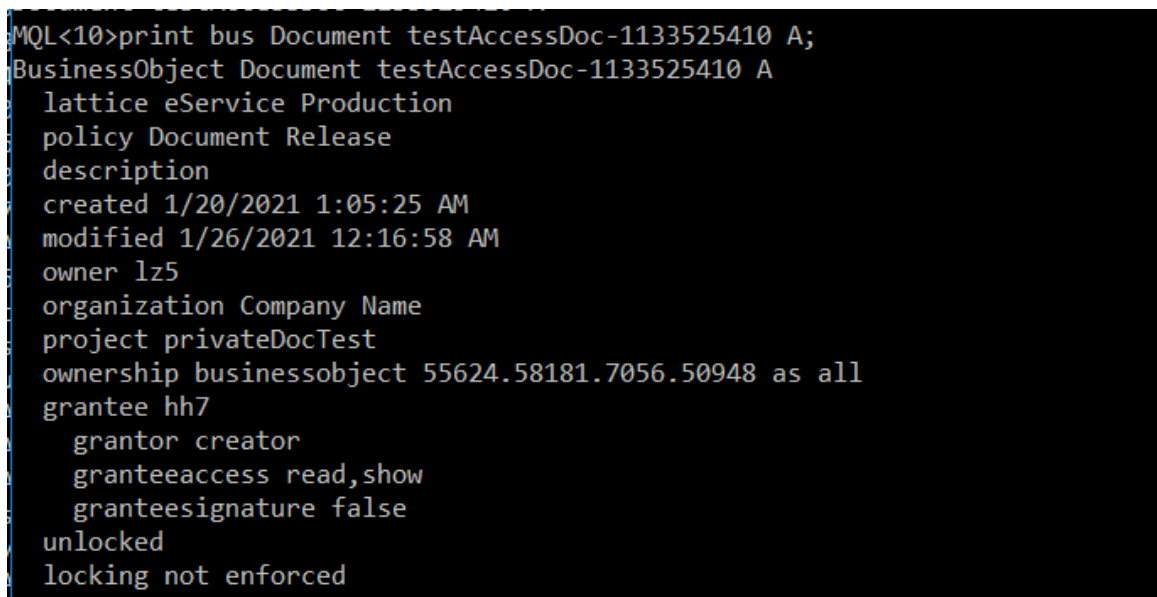
```
Matrix Query Language Interface, Version 3DEXPERIENCE R2020x HotFix 2 (64 bits)

Copyright (c) 1993-2019 Dassault Systemes.
All rights reserved.
MQL<1>set context user creator;
MQL<2>print bus Document docAccessTest-1650970312 A select current.access;
business object Document docAccessTest-1650970312 A
current.access = all
MQL<3>
```

我们从上图可以看到，**creator** 这个超级管理员在当前的文档对象上具备了所有权限，所以，我们使用 **creator** 用户来给 **hh7** 进行 **grant** 赋权操作。

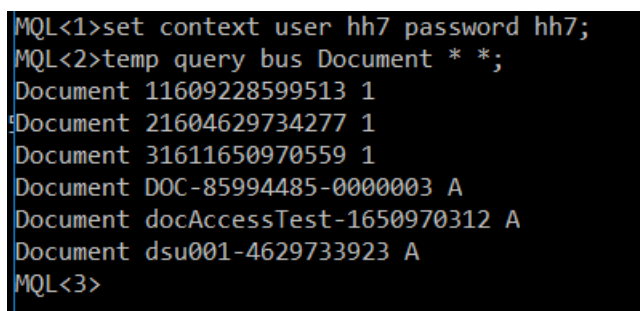
MQL 命令为：**mod bus Document docAccessTest-1650970312 A grant hh7 access read,show;**

命令执行成功后，我们使用 MQL 命令对当前这个文档对象内容进行打印，如下图：



```
MQL<10>print bus Document testAccessDoc-1133525410 A;
BusinessObject Document testAccessDoc-1133525410 A
lattice eService Production
policy Document Release
description
created 1/20/2021 1:05:25 AM
modified 1/26/2021 12:16:58 AM
owner lz5
organization Company Name
project privateDocTest
ownership businessobject 55624.58181.7056.50948 as all
grantee hh7
grantor creator
granteeaccess read,show
granteesignature false
unlocked
locking not enforced
```

从上图我们可以看到，当前对象对于 **hh7** 这个用户已经赋予了 **read**、**show** 权限，然后我们使用 **hh7** 用户登录 MQL 命令进行文档查询，查询结果如下图：



```
MQL<1>set context user hh7 password hh7;
MQL<2>temp query bus Document * *;
Document 11609228599513 1
Document 21604629734277 1
Document 31611650970559 1
Document DOC-85994485-0000003 A
Document docAccessTest-1650970312 A
Document dsu001-4629733923 A
MQL<3>
```

我们可以看到，赋予临时权限后，用户 **hh7** 具备了对文档 “docAccessTest-1650970312” 的可见权限了。

然后我们再使用 MQL 命令将临时赋权取消，具体 MQL 命令为：**mod bus Document docAccessTest-1650970312 A revoke grantee hh7;**

执行成功后，我们再使用用户 **hh7** 进行文档查询，我们可以看到，用户 **hh7** 又搜不到文档对象 “docAccessTest-1650970312” 了，也就是说取消临时赋权成功。

6.2. Ownership 赋权模式

Ownership 的权限赋予分三种模式，分别为：Explicit SOV，Inherited ownership 和 Inherited access。

6.2.1. Explicit SOV

这种赋权在系统里相当于给当前对象增加了余外的 POV 信息（组织和协作区所属）。

操作的 MQL 命令为：**mod bus T N R add ownership ORG PRJ as read,show.**

ORG 和 PRJ 可以使用通配符 “-”，上面的示例是只指给了 **read** 和 **show** 权限，我们可以指给更多的权限，如果是全部权限，那么可以直接指定 **all**。

一个简单示例：**mod bus Document docAccessTest-1650970312 A add ownership - protectedCS as read,show;**

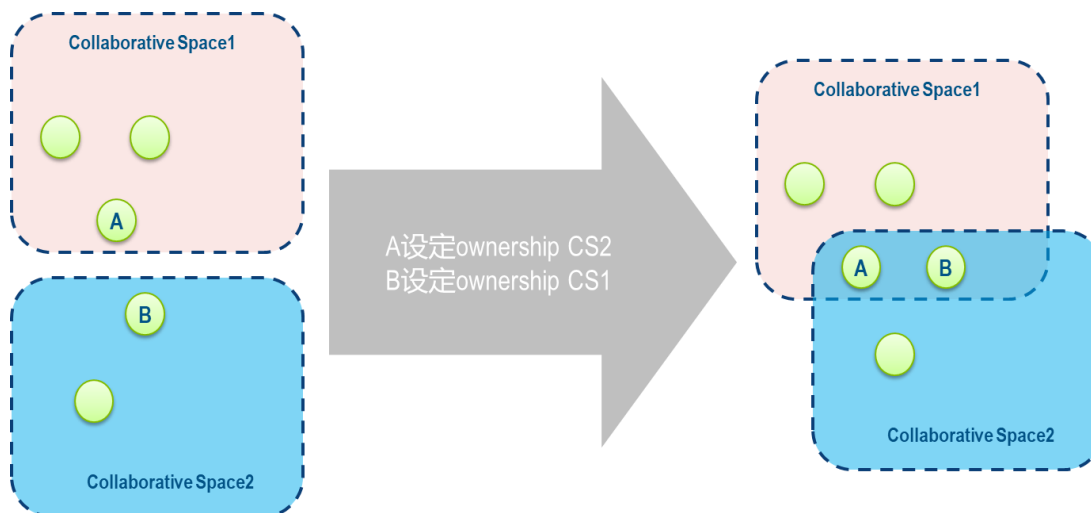
```
MQL<9>print bus Document docAccessTest-1650970312 A;
BusinessObject Document docAccessTest-1650970312 A
  lattice eService Production
  policy Document Release
  description
  created 1/26/2021 12:49:30 AM
  modified 1/27/2021 10:20:10 PM
  owner lz5
  organization Company Name
  project privateDocTest
  ownership businessobject 55624.58181.36875.49158 as all
  ownership - protectedCS as read,show
  unlocked
```

上面的示例我们可以看到用户多了一个 **ownership**，里面多表明了一个协作区

“**protectedCS**”，后面的标志权限为 **read** 和 **show**。这代表着当前的对象如果要检测对当前用户是否有 **read** 和 **show** 权限时，当前对象即属于 “**privateDocTest**”，也属于

“protectedCS”，只要用户在这两个随便哪一协作区的凭证具备对协作区内的对象 read 和 show 权限，那么用户就具备对当前对象的 read 和 show 权限。

下图是一个示例，我们可以理解设定 ownership 后，协作区就成为一个具有重合叠加的划分了。



上面的示例是将 PRJ 指定为一个协作区，我们还可以直接指定为一个人员，比如下图：

```
MQL<8>print bus Document dsu001-4629733923 B;
BusinessObject Document dsu001-4629733923 B
  lattice eService Production
  policy Document Release
  description
  created 11/5/2020 6:29:31 PM
  modified 11/5/2020 6:29:39 PM
  owner lz5
  organization Company Name
  project dsu
  ownership - lz5_PRJ for Multiple Ownership For Object as read,modify,checkout,checkin,lock,unlock,revise,majorrevise,
  changename,changetype,fromconnect,toconnect,fromdisconnect,todisconnect,show,reserve,unreserve
  ownership businessobject 55624.58181.45199.20865 as all
  unlocked
```

这就代表，不管当前对象的 POV 信息是什么，也不管 lz5 这个用户被指定了什么上下文凭证，对于当前对象，lz5 这个用户都具备 as 后面指定的一系列权限。

6.2.2. Inherited ownership

系统中设定 Ownership 是可以被继承的，也就是说如果一个对象继承了另外一个对象的 ownership，那么他也就相当于被设定了父对象的所拥有所有 ownership。Ownership 的继承，系统中最常见的一个示例是文档对文件夹权限的继承，这个在系统后台的实现逻辑，就是通过文档继承了文件夹的 ownership 来进行实现的。如下图：


```

MQLC10>print bus Document docAccessTest-1650970312 A;
BusinessObject Document docAccessTest-1650970312 A
  lattice eService Production
  policy Document Release
  description
  created 1/26/2021 12:49:30 AM
  modified 1/27/2021 10:20:10 PM
  owner lz5
  organization Company Name
  project privateDocTest
  ownership businessobject 55624.58181.36875.49158 as all
  ownership - protectedCS as read,show
  unlocked
  locking not enforced
  title docAccessTest

id = 55624.58181.47588.16388
MQLC10>print bus 55624.58181.36875.49158 {history;
BusinessObject Workspace Vault docAccessTest 299680560000294C600FD77900000127
  lattice eService Production
  policy Workspace Vaults
  description docAccessTest
  created 1/26/2021 12:49:10 AM
  modified 1/27/2021 10:14:30 PM
  owner lz5
  organization Company Name
  project privateDocTest
  ownership businessobject 55624.58181.41869.48915 as all
  ownership - lz5_PRJ for Multiple Ownership For Object as read,modify,delete,checkout,checkin,lock,unlock,changeowner,
  promote,demote,revise,majorrevise,changename,changetype,fromconnect,toconnect,fromdisconnect,todisconnect,show,reserve,u
  preserve,changeobj

```

我们可以看到文档对象“docAccessTest”具备了一个继承的 ownership “ownership businessobject 55624.58181.36875.49158 as all”，而 ID 为“55624.58181.36875.49158”从上图右侧，我们可以看到，就是文档对象“docAccessTest”所属的文件夹

“docAccessTest”，这句 ownership 代表的含义就是文档对象继承了文件夹所拥有的所有权限，在具体的操作实例我们已经知道，如果文件夹设置了一个具有读权限的用户，那么不管这个用户之前对文件夹下的文件是否具备读权限，在设置生效后，这个用户对这个文件夹下的文件，以及子文件夹下所有的文件都会具备读权限。

Ownership 的继承设定的 MQL 命令为：**mod bus B add ownership bus A as read,show;**

6.2.3. Inherited access

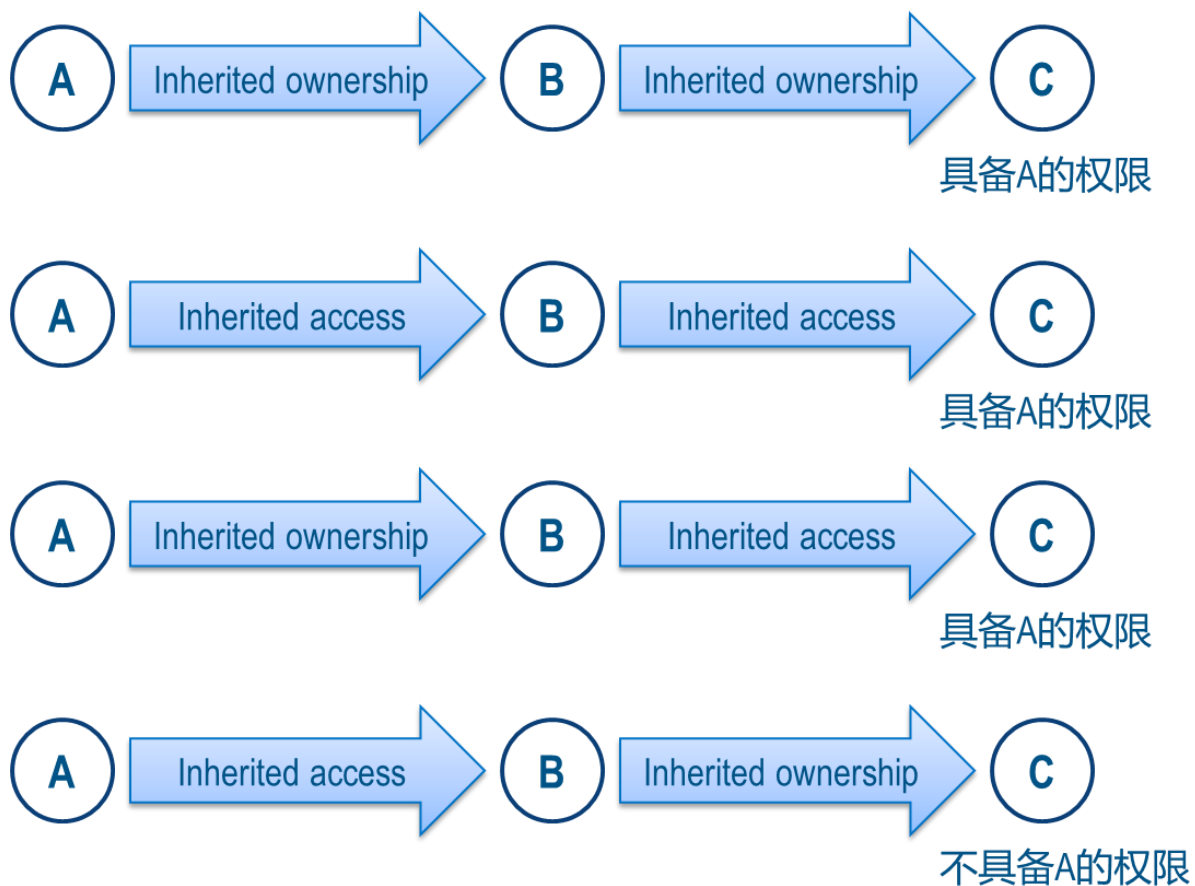
Inherited access 这种模式和 Inherited ownership 模式非常相似，但是二者最多的差别在于一个是 ownership 的继承，而另外一个直接的 access 的继承。

这里需要着重说明一下，ownership 本身并不是一个权限的描述，单纯从 ownership 上，我们也得不出当前用户对当前对象具备了什么权限。权限的判定是根据当前的 ownership，再去进行权限的判定的。

Ownership 的继承设定的 MQL 命令为：**mod bus B add access bus A as read,show;**

6.2.4. Inherited ownership VS Inherited access

我下面用几个简单的示例图来解释二者的相同与差异的地方。



6.2.5. 对象进行 clone 和 revise 操作后，临时赋权的变化

下面有一张表，表明了当对象进行 clone 和 revise 操作，这三种临时赋权模式设置方式。

Operation	POV/SOV	Regular BO (or root composer)	Composee
Clone	<i>POV (Org/Project)</i>	Login context Org/Project	Login context Org/Project
	<i>Explicit SOV (Org/ Project)</i>	Reset (null)	Reset (null)
	<i>Inherited SOV</i>	Reset (null)	Reset (null)
	<i>Inherited Access</i>	Reset (null)	Reset (null)
Revise (Major/Minor)	<i>POV (Org/Project)</i>	Copy	Copy
	<i>Explicit SOV (Org/ Project)</i>	Copy	Copy
	<i>Inherited SOV</i>	Reset (null)	Reset (null)
	<i>Inherited Access</i>	Reset (null)	Reset (null)