【V1.3_2】外部需求梳理

esp-slides项目对外界依赖的需求整理

依赖系统	类别	业务需求	希望支持的能力
LC	功能	在资源的查询过程中，希望能过滤掉某些特定的状态，比如状态不为CREATING的状态，目前LC关于资源的查询只有eq与in操作，无法满足现在的业务需求	希望LC对于资源的查询能提供 不等 不在（not in） 等操作
LC	功能	在资源的查询过程中，希望过滤条件，能支持逻辑运算，目前默认为and逻辑，希望能够有or的逻辑，来满足未来的需求	希望LC对于资源的逻辑运算能支持or
LC	性能	关于资源的查询，之前出现查询非常慢，查询速度不应该随着数据量的增加而指数增加，会严重影响用户体验 这是之前在集成上查询的一个结果： http://esp-lifecycle.pre1.web.nd/v0.6/coursewares/actions/query?words=&limit=(0,15)&include=TI,LC,CG,CR&relation=chapters/a10f58dc-e6ab-4d16-9699-c1fab3e154d0/ASSOCIATE&coverage=Org/nd/OWNER&coverage=Org/491036498581/OWNER,time 4516ms 这是属于相对比较快的情况，甚至有时候都链超时了，数据都没返回.	希望查询能快速响应，最好在单个1秒内能把响应数据返回（在数据量到达一定程度后）
LC	功能	当前情况，资源拷贝的时候，原始资源的关系没有拷贝过来，还需要业务方多做许多处理（先查关系，再继续与拷贝后的资源进行关系整合，）同理覆盖范围也是	希望LC在拷贝接口上能够有参数指明是否需要拷贝关系参数，以及是否需要保留原有的覆盖范围能力
LC	功能	调用LC的拷贝接口的时候，加入拷贝的资源在ND库，我指定to拷贝到个人库，经过LC开发人员确认，相应的CS文件还是存在于公共库的，这就导致了逻辑与物理不一致.	希望LC在处理资源拷贝的时候，CS文件也需要跟着变动.
LC	功能	在目前定义关于资源/{uuid}.pkg包的描述中都需要有一个metadata.json文件来描述该资源的元数据信息，目前规范已经出来（@王永弟）. 目前做法则是由编辑器来执行创建metadata.json这些逻辑.	希望能将这个能力移到LC中进行统一管理，在创建/更新元数据的时候，在{uuid}.pkg中能够创建/更新metadata.json文件 这是一个完整的一个逻辑，不适合拆分掉各个系统去做.
LC	功能	在编辑器中有这样的一个需求：从备课生产上打下来的ndp包，要导入到预生产中（跨环境执行ndp包），在ndp包中有对于外部资源依赖的路径： _ref/prepub_content_edu/esp/... 那么我需要判断改包是从哪个环境打下来的，我需要判断prepub_content_edu与当前的cs实例名是否属于同一个环境（生产/预生产/集成）等，但是理论上编辑器并不管理cs实例名的，因此也不知道cs实例名是否属于同一个环境	因为LC是管理cs实例名的，因此希望能提供一个接口判断两个实例名是否属于同一个环境，并且属于哪个环境(生产/预生产/集成)等等.
LC	功能 性能	下个版本，编辑器将要继续统计某个章节下的所有资源有多少数量，以及某个教材下的所有资源有多少数量，目前预计的策略则是采用调用原来的接口，但这对性能必定会带来严重的考验，并且不需要里面的具体数据，只需要数量即可.	希望LC能提供关于资源查询的统计接口（既不需要将数据返回，但查询性能要比查数据的快很多），能够返回查询到多少记录，总共有多少记录.
LC	建议	关于覆盖范围的校验 当前关于校本库，备课采用了Org/{node_id}/OWNER，按照原先的设计采用Org的，则id是为组织ID，采用了node_id则是为了满足业务上的需求，当使用node_id的时候是否是需要采用Org呢？	建议LC在传入覆盖范围的时候，针对org_id或者user_id做校验，如果不符合则进行报错处理.
CS	功能 性能	目前备课当中上传一个ndp文件，把ndp包里面的文件往CS上传，这个ndp包有些目录都是一些小文件，但是个数很多，目前都是通过调用CS上传单文件的方式一个一个上传，文件虽小，但是连接请求次数比较多，造成网络资源的浪费.	希望CS能够提供一个批量上传的接口，并且批量告诉CS这些文件分别存在哪个目录.
CS	功能 性能	目前关于秒传不支持覆盖文件，从已有目录的一个文件秒传到目标目录，如果目标目录文件存在，则会报错，并且秒传接口没有支持可以覆盖参数，临时解决方案有： 1.如果发现已存在文件，先读取流，再调用上传接口进行覆盖（当前采用这种策略） 2.先删除文件，再进行秒传	第一种策略的弊端在于，如果需要秒传的文件比较大，那么就会多浪费了下载上传这样的网络IO 第二种策略的弊端在于，如果第二次秒传失败，那么原有文件就不存在了，则还需要记录原始文件状态，再去回收站中还原数据. 希望CS能把二者能力进行结合，提供参数，是否重复进行覆盖，由业务方来决定是否覆盖

CS	功能 性能	关于跨实例copy目录，目前用户课件PPT的拷贝，有可能源PPT与目标PPT可能存在于两个实例，CS不支持进行跨实例的拷贝，目前通过MD5的值进行跨实例递归秒传	虽然通过递归妙传的方式避免了文件下载上传的问题，但是从一个目录到另外一个目录并且通过文件妙传的方式，也是一个一个文件妙传，还是消耗了大量的连接请求。也一定程度上影响了性能. 希望CS能够支持跨实例的目录拷贝
CS	功能 性能	目前编辑器需要大量的与CS的文件进行交互，编辑器系统中存在着大量的这样逻辑： 1.判断一个文件目录是否存在 2.如果存在，则进行覆盖文件 3.如果不存在，则进行上传文件 这里就会调用两个请求. 1.根据路径获取dentry.（这是一个patch请求，在rest上并不会被执行缓存） 2.上传文件接口	CS在上传的时候能否提供参数支持，如果设定了某个参数，如果文件存在则进行直接继续覆盖