cidner.volume.manager.VolumeManager.initialize_connection过程分析：

执行者：

cinder.volume.drivers.netapp.dataontap.iscsi_cmode.NetAppCmodeISCSIDriver：

cinder.volume.drivers.netapp.dataontap.block_base.NetAppBlockStorageLibrary

cinder.volume.drivers.netapp.dataontap.block_cmode.NetAppBlockStorageCmodeLibrary

上述类型（NetAppBlockStorageCmodeLibrary）封装了一下东西：

1.NetApp的客户端类：zapi_client

cinder.volume.drivers.netapp.dataontap.client.client_cmode.Client：

客户端类执行操作：

get_lun_list()方法返回当前SVM下所有的LUN（疑问：有的LUN没被查到？？）

get_igroup_by_initiators(initiator_list)方法返回group信息

create_igroup(igroup_name, initiator_group_type,host_os_type) 创建一个组，并返回group名

add_igroup_initiator(igroup_name, initiator) 将指定initiator加入group

map_lun(path, igroup_name, lun_id=lun_id) Volume Path映射到Group，返回lun_id

get_lun_map(self, path) 获取path的所有LUN映射？

get_iscsi_target_details() 获取target信息

get_iscsi_service_details() 获取iscsi的iqn

2.内存中的LUN缓存：lun_table

lun_table是一个格式如{vol_id:NetAppLun}的字典

NetAppLun表示后端中一个LUN信息如下：

handle：honghe_io_1_iscsi_1:/vol/vol_09052017_153410_4/volume-ce979ab0-7f70-48a6-a176-87bfc87a18a2

metadata：

{

'Volume': 'vol_09052017_153410_4',

'UUID': 'ec1ae69b-a704-4eac-9ec9-e12dae79e11c',

'Qtree': None,

'Vserver': 'honghe_io_1_iscsi_1',

'SpaceReserved': 'true',

'OsType': 'linux',

'Path': '/vol/vol_09052017_153410_4/volume-ce979ab0-7f70-48a6-a176-87bfc87a18a2'

}

name : volume-ce979ab0-7f70-48a6-a176-87bfc87a18a2

size : 2147483648

每次cinder-volume启动，lun_table会被初始化！

执行流程：

cidner.volume.manager.VolumeManager.initialize_connection -> cinder.volume.drivers.netapp.dataontap.block_cmode.NetAppBlockStorageCmodeLibrary.initialize_connection_iscsi

A.获取lun_id : lun_id = self._map_lun(name, [initiator_name], 'iscsi', None)

1.在内存中根据ID在内存中查找NetAppLun以获取Volume的后端地址，如：/vol/vol_09052017_153410_4/volume-ce979ab0-7f70-48a6-a176-87bfc87a18a2

在这个过程中如果从lun_table中获取失败，那么驱动会扫描后端所有LUN，并加入lun_table!还没有，则直接失败！

2.获取、创建的组 : igroup_name, ig_host_os, ig_type = self._get_or_create_igroup(initiator_list, initiator_type, self.host_type)

_get_or_create_igroup方法中initiator_list虽然是个List，但是实质上只封装了nova-compute的initiator名称！

zapi_client类的get_igroup_by_initiators方法，只需要initiator名就可以返回需要的group信息。

group信息如下：

{

'initiator-group-os-type': 'linux',

'initiator-group-name': 'openstack-abed682b-5e98-43f7-956d-bd0c9fea94f8',

'initiator-group-type': 'iscsi'

}

注意：

如果initiator属于多个group，这个函数会返回多个group（当然正常情况下只有一个，并且名字以openstack开头）！

Cinder只会取返回值中的第一个group进行后续操作！

if igroups:

```
igroup = igroups[0]
igroup_name = igroup['initiator-group-name']
host_os_type = igroup['initiator-group-os-type']
initiator_group_type = igroup['initiator-group-type']
```

如果没有找到initiator对应的组，_create_igroup_add_initiators方法会被调用：

igroup_name = self._create_igroup_add_initiators(initiator_group_type, host_os_type, initiator_list)

_create_igroup_add_initiators会指定一个group名，然后调用客户端在后端创建组，最后将传入的initiator_list加入组（注意：这里_create_igroup_add_initiators只创了一个，但是允许将多个initiator加入组）。

```
def _create_igroup_add_initiators(self, initiator_group_type, host_os_type, initiator_list):
    """Creates igroup and adds initiators."""
    igroup_name = na_utils.OPENSTACK_PREFIX + six.text_type(uuid.uuid4())
    self.zapi_client.create_igroup(igroup_name, initiator_group_type,host_os_type)
    for initiator in initiator_list:
        self.zapi_client.add_igroup_initiator(igroup_name, initiator)
    return igroup_name
```

3.将对应的Volume Path映射到Group，并返回lun_id（0,1,2,3）

zapi_client.map_lun(path, igroup_name, lun_id=lun_id)

从目前分析的情况来看，Cinder不会指定lun_id,因此Group下的Lun ID是0，从0开始增长（不排除出现某个中间的id已经被dettach了，那么下一个就是中间的这个ID）。

同时,如果一个Vol被dettach了，Volume Path会从Group中被移除。

对于上述方法如果失败,有如下异常处理：

(_igroup, lun_id) = self._find_mapped_lun_igroup(path,initiator_list)

该异常处理：a)获取path的所有LUN映射；b)获取initiator_list（就一个）的所有group；c)在group中查找第一个满足条件的映射。

B.获取target信息 : target_list = self.zapi_client.get_iscsi_target_details()

信息如下：

```
{
'tpgroup-tag': '1033',
'interface-enabled': 'true',
'port': '3260',
'address': '172.24.3.150'
},
{
'tpgroup-tag': '1034',
'interface-enabled': 'true',
'port': '3260',
'address': '172.24.3.151'
}
```

C.从获取的target_list中选取一个 ：preferred_target = self._get_preferred_target_from_list(target_list)
D.获取target的iqn ：iqn = self.zapi_client.get_iscsi_service_details()
F.将上面获取的信息包装返回值，返回给Nova

实验：
1.通过Cinder创建一个卷，创建好之后Vol不属于任何iGroup，将卷加入iGroup1，将卷attach到虚机（对应为iGroup2），attach之后iGroup1中的映射依然存在！
2.通过Cinder刚创建出来的数据卷没有provider_location,执行过一次attach后这个值才会生成！？
3.通过Cinder创建一个卷，创建好之后Vol不属于任何iGroup，在后端将卷加入宿主机的Group，执行rescan命令，/dev/dish/by_path目录下可以看到连接文件！此时在OpenStack前台执行连接，可以成功且LUN_ID不变!在后端将卷移出group，执行rescan，析/dev/dish/by_path目录下的文件依然存在！！此时在OpenStack前台执行分离可以成功！之后再查看/dev/dish/by_path文件消失！

iscsiadm命令：

1.发现：

iscsiadm -m discoverydb -t sendtargets -p 172.24.3.150:3260 --discover

2.登录：

iscsiadm --mode node --targetname iqn.1992-08.com.netapp:sn.527720331fe011e7bb1500a098c364d6:vs.5 --portal 172.24.3.150:3260 --login

3.登出：

iscsiadm --mode node --targetname iqn.1992-08.com.netapp:sn.527720331fe011e7bb1500a098c364d6:vs.5 --portal 172.24.3.150:3260 --logout

4.重新扫描：

iscsiadm -m node -T iqn.1992-08.com.netapp:sn.527720331fe011e7bb1500a098c364d6:vs.5 -p 172.24.3.150:3260 --rescan