# 贡献一份精心整理的RBAC规范接口及其解说

这是花费了我半天时间从国标GBT 25062-2010 RBAC上整理得到的IRBACService接口。不要感觉惊奇，GBT 25062-2010 RBAC是和美国的NIST RBAC标准完全一样的。如果您希望阅读GBT 25062-2010 RBAC国标文档的话可以从这里下载<http://git.oschina.net/anycmd/anycmd/tree/master/Docs>。

国家标准文档是用z语言书写的，一下子读不懂没关系，不要灰心，把丢掉的知识拾起来就可以了。咱们都是在同样的教育体系下接收教育的，咱们的知识结构是被众多金字塔尖的人精心设计和建造的，只要你静下心来没有你阅读不懂的人类知识人类文明。

各种标准文档上书写着的是无数人的智慧与聪明才智。那是他们的价值、荣誉与生命，希望你撇开相互的偏见去获取你想要的知识。理论与生产没有矛盾。

为了帮助大家阅读，我斗胆预先帮助大家预热一下一些重要概念。

**主体：**指对客体有认识和实践能力的对象。比如人、系统、服务提供者、服务消费者等。

**客体：**可被主体感知或想象到的任何事物。如文件、打印机、终端、数据库记录等。

**对象/资源：**资源是需要进行访问控制的系统资源，例如文件、打印机、终端、数据库记录等。

**资源类型：**基于人类发明的分类法对资源按照性质、特点、用途等作为区分的标准而做的第一次分类。

**权限：**对受保护的对象执行某个操作的许可。

**操作：**一个过程，这个过程通常有输入与输出，这个过程可能影响系统的状态也可能不影响系统的状态，是否影响系统的状态有赖于你的领域边界。首先基于资源类型定义一类资源的操作列表，通常这就够了，但也有可能会需要针对特定对象实例定义操作列表。

你也可以直接查看[IRBACService.cs](http://git.oschina.net/anycmd/anycmd/blob/master/Framework/Anycmd/IRBACService.cs)代码文件，它上面承载的是和下面的表格承载的同样的信息。

所有的RBAC标准接口都放进了一个类是希望把RBAC看成一个整体，不作根据anycmd的理解而施加的分类。接口的使用者可以根据自己的理解再对这套接口进行分类。比如接口的使用者可以把下面所有方法分类成User类、Role类、Privilege类等。

|  |  |
| --- | --- |
| **方法签名** | **说明** |
| void AddUser(IAccountCreateInput input) | 该命令创建一个新用户。当要待创建用户尚不存在于Account集合（Account表）中时，该命令可用。命令执行后，Account集合（Account表）被更新，新创建的用户不拥有任何的会话。 |
| void DeleteUser(Guid accountID) | 该命令从RBAC数据库中删除一个已经存在的用户。该命令可用当且仅当被删除的用户是Account数据集（Account表）  的一个成员（记录）。Accounts数据集（Account表）将被更新、PrivilegeBigram数据集（PrivilegeBigram表）将被更新，  PrivilegeBigram集合中主体为当前被删除的账户的成员将被删除。如果一个正处在会话中的用户被删除，anycmd  的实现会等待该会话结束后删除，但账户管理员是可以看到被删除的账户是否正在会话的且有权的管理员是可以强行终止给定的账户的会话的。 |
| void AddRole(IRoleCreateInput input) | 该命令创建一个新的角色。该命令可用当且仅当要创建的角色尚且不存在于RoleSet数据集中。RoleSet数据集  将被更新。初始时，新创建的角色没有分配任何用户和权限。 |
| void DeleteRole(Guid roleID) | 该命令从RBAC数uk中删除一个角色。该命令可用当且仅当被删除的角色是RoleSet数据集的成员。如果被删除的角色在某些会话中尚且是激活的，则Anycmd会从会话中删除这个角色然后允许会话继续执行。 |
| void AssignUser  (Guid accountID, Guid roleID) | 该命令给用户分配角色。该命令可用当且仅当该用户是Account数据集（Account表）的成员（记录），该角色是RoleSet  数据集的成员，并且该角色尚未分配给该用户。数据集（表）PrivilegeBigram将被更新。  1 该用户是Account数据集（表）的成员（记录）；  2 该角色是RoleSet数据集的成员；  3 该用户还没有被分配该角色；  4 所有SSD约束在执行完命令后仍然被满足。 |
| void DeassignUser  (Guid accountID, Guid roleID) | 该命令删除一个角色role到用户account的分配。该命令可用当且仅当account是Account数据集的成员，role  是RoleSet数据集的成员，并且角色role已经分配给了用户account。 |
| void GrantPermission  (Guid functionID, Guid roleID) | 该命令给一个角色分配对一个对象执行某个操作的权限。该命令可用当且仅当给定的（操作，对象）代表了  一项权限并且该角色是RoleSet数据集的成员。 |
| void RevokePermission  (Guid functionID, Guid roleID); | 该命令从分配给角色的权限集中撤销对某个对象执行某个操作的权限。该命令可用当且仅当（操作、对象）代表一项权限，并且该权限已经分配给了该角色。 |
| IUserSession CreateSession  (  Guid sessionID, AccountState worker,  List<PrivilegeBigramState> privileges  ); | 该函数创建一个新的会话，以指定的用户作为会话拥有者，以指定的角色集作为激活角色集。该函数可用  当且仅当：  1 该用户是Account数据集（表）的成员（记录）；  2 该会话的激活角色集是该用户分配的角色集的子集。  3 该会话的激活角色集满足所有的DSD约束。 |
| void DeleteSession(Guid sessionID) | 该函数删除一个会话。该函数可用当且仅当会话标识符是UserSession数据集（表）的成员（记录）。 |
| List<RoleState> SessionRoles  (Guid sessionID) | 该函数返回给定会话的激活角色。该函数可用当且仅当该会话标识符是UserSession数据集（表）的成员（记录）。 |
| List<PrivilegeBigramState>  SessionPermissions(Guid sessionID) | 该函数返回给定会话的权限，即该会话的激活角色拥有的权限。该函数可用当且仅当会话标识符是  UserSession数据集（表）的成员（记录）。 |
| void AddActiveRole  (  Guid accountID,  Guid roleID,  Guid sessionID  ) | 该函数为给定的用户会话增加一个激活角色。该函数可用当且仅当：  1 该用户是Account数据集（表）的成员（记录）；  2 该角色是RoleSet数据集的成员；  3 会话标识符是UserSession数据集（表）的成员（记录）；  4 该角色已经分配给了该用户；  5 该用户拥有该会话；  6 在给该会话增加新的激活角色后，所有的DSD约束都被满足。 |
| void DropActiveRole  (Guid accountID, Guid roleID) | 该函数从给定用户会话中删除一个激活角色。该函数可用当且仅当该用户是Account数据集（表）的成员（记录），  会话标识是UserSession数据集（表）的成员（记录），该用户是该会话的拥有者并且该角色是该会话的一个激活角色。 |
| bool CheckAccess  (  Guid sessionID,  Guid functionID,  IManagedObject obj  ); | 该函数决定一个给定的会话的主体是否允许对给定的对象执行某个给定的操作并返回一个布尔值。该函数可用当且仅当  会话标识符是UserSession数据集（表）的成员（记录），该对象是它对应类型的对象的数据集的成员，该操作是FunctionSet  数据集的成员。会话的主体可以对该对象执行该操作当且仅当会话的某个激活角色拥有对应的权限。 |
| List<AccountState> AssignedUsers  (Guid roleID) | 该函数返回被分配给了某个指定角色的用户。该函数可用当且仅当该角色是RoleSet数据集的成员。 |
| List<RoleState> AssignedRoles  (Guid accountID) | 该函数返回分配给了一个给定用户的角色。该函数可用当且仅当该用户是Account数据集（表）的成员（记录）。 |
| List<AccountState> AuthorizedUsers  (Guid roleID) | 该函数返回拥有给定角色的授权用户。该函数可用当且仅当给定角色是RoleSet的成员。 |
| List<RoleState> AuthorizedRoles  (Guid accountID) | 该函数返回给定用户的授权角色。该函数可用当且仅当该用户是Account数据集（表）的成员（记录）。 |
| List<PrivilegeBigramState>  RolePermissions(Guid roleID) | 该函数返回分配给一个给定角色的权限。该函数可用当且仅当该角色是RoleSet数据集的成员。 |
| List<PrivilegeBigramState>  UserPermissions(Guid accountID) | 该函数返回一个给定用户的权限。该函数可用当且仅当该用户是Account数据集（表）的成员（记录）。 |
| List<FunctionState>  RoleOperationsOnObject  (Guid roleID, IManagedObject obj) | 该函数返回一个给定角色被允许的对给定对象执行的操作。该函数可用当且仅当该角色是RoleSet数据集的成员，该对象是obj对应类型数据集的成员。 |
| List<FunctionState>  UserOperationsOnObject  (Guid accountID, IManagedObject obj) | 该函数返回给定用户被允许的针对给定角色执行的操作。该函数可用当且仅当该用户是Account数据集（表）的成员（记录），该对象是obj对应类型数据集的成员。 |
| void AddInheritance  (  Guid subjectRoleID,  Guid objectRoleID  ) | 该命令在两个已经存在的角色r\_asc/客体和r\_desc/主体之间建立直接继承关系。r\_asc>>r\_desc。该命令可用当且仅当：  1 r\_asc/客体和r\_desc/主体都是RoleSet数据集的成员；  2 r\_asc/客体不是r\_desc/主体的直接祖先，  3 r\_desc/主体不继承r\_asc/客体（不免产生回路）。  4 SSD约束在该命令执行后扔是满足的。 |
| void DeleteInheritance  (  Guid subjectRoleID,  Guid objectRoleID  ) | 该命令删除已经存在的直接继承关系r\_asc>>r\_desc。该命令可用当且仅当r\_asc/客体和r\_desc/主体都是RoleSet数据集的成员，r\_asc/客体是r\_desc/主体的直接祖先。在执行完该命令以后，新的继承关系是新的直接继承关系的自反传递包。 |
| void AddAscendant  (  Guid childRoleID,  IRoleCreateInput parentRoleCreateInput  ) | 该命令创建一个新角色r\_asc/客体/父角色，并作为现存角色r\_desc/主体/子角色 的直接祖先插入到角色层次中去。该命令可用当且仅当r\_asc/客体/父角色 不是RoleSet数据集的成员，r\_desc/子角色 是RoleSet数据集的成员。 |
| void AddDescendant  (  Guid parentRoleID,  IRoleCreateInput childRoleCreateInput  ) | 该命令创建一个新的角色作为现存角色 r\_asc/客体/父角色 的直接后代插入到角色层次中。该命令可用当且仅当r\_desc/主体不是RoleSet数据集的成员，r\_asc/客体/父角色 是RoleSet数据集的成员。 |
| void CreateSSDSet  (ISSDSetCreateInput input) | 该命令创建一个命名的SSD角色集合，并设定相应的阀值。该命令可用当且仅当：  1 SSD角色集的标识和名称还没有被使用；  2 SSD角色集中的角色都是RoleSet数据集的成员；  3 SSDCard是一个大于或等于2的自然数，同事还要小于或等于SSD角色集的基数（基数：有限集的元素个数）；  4 新的SSD角色集的约束当前是被满足的。 |
| void DeleteSSDSet(Guid ssdSetID) | 该命令删除一个SSD角色集。该命令可用当且仅当该SSD角色集存在。 |
| void AddSSDRoleMember  (Guid ssdSetID, Guid roleID) | 该命令为SSD角色集增加一个角色，该SSD角色集关联的阀值不发生改变。该命令可用当且仅当：  1 该SSD角色集存在；  2 该角色是RoleSet数据集的成员，并且尚不属于该SSD角色集；  3 该命令执行之后，SSD约束仍然是满足的。 |
| void DeleteSSDRoleMember  (Guid ssdRoleID) | 该命令从SSD角色集中删除一个角色，其关联的阀值不发生改变。该命令有效当且仅当：  1 SSD角色集已经存在；  2 要删除的角色是该SSD角色集的成员；  3 该SSD角色集关联的阀值小于该角色集的成员数。  <remarks>  注意：修改后的SSD约束当前应该是被满足的。 |
| void SetSSDCardinality  (Guid ssdSetID, int cardinality) | 该命令设定与给定的SSD角色集关联的阀值。该命令可用当且仅当：  1 该SSD角色集存在；  2 新的阀值是一个大于或等于2的自然数，它要小于或等于SSD角色集的基（基：有限集元素的个数）；  3 新的SSD约束当前应该是被满足的。 |
| List<SSDRoleState> SSDRoleSets() | 该函数返回所有的SSD角色集。 |
| List<RoleState> SSDRoleSetRoles  (Guid ssdSetID) | 该函数返回与一个指定SSD角色集合相关联的角色的集合 |
| int SSDRoleSetCardinality  (Guid ssdSetID) | 该函数返回与给定SSD角色集关联的阀值。该函数可用当且仅当该角色集存在。 |
| void CreateDSDSet  (IDSDSetCreateInput input) | 创建一个动态责任分离角色集并设定相应的阀值。该DSD约束要求DSD角色集中任何【阀值】个或者更多角色不能都在某个用户会话过程中被激活。  该命令可用当且仅当：  1 DSD角色集的标识和名称还没有被使用；  2 DSD角色集中的角色都是RoleSet数据集的成员；  3 SSDCard是一个大于或等于2的自然数；  4 新的DSD角色集的约束当前是被满足的。 |
| void DeleteDSDSet(Guid dsdSetID) | 该命令删除一个DSD角色集。该命令可用当且仅当该DSD角色集存在。 |
| void AddDSDRoleMember  (Guid dsdSetID, Guid roleID) | 该命令为一个给定的DSD角色集增加一个角色，DSD角色集关联的阀值不发生改变。该命令有效当且仅当：  1 该DSD角色集存在；  2 该角色是RoleSet数据集的成员，并且尚不属于该DSD角色集；  3 该命令执行成功之后，DSD约束仍然是满足的。SSDCard要小于或等于DSD角色集的基（基：有限集元素的个数） |
| void DeleteDSDRoleMember  (Guid dsdRoleID) | 该命令从DSD角色集中删除一个角色，其关联的阀值不发生改变。该命令有效当且仅当：  1 DSD角色集已经存在；  2 要删除的角色是该DSD角色集的成员；  3 该DSD角色集关联的阀值小于该角色集的成员数。  注意：修改后的DSD约束当前应该是被满足的。 |
| void SetDSDCardinality  (Guid dsdSetID, int cardinality) | 该命令设定与给定的DSD角色集关联的阀值。该命令可用当且仅当：  1 该DSD角色集存在；  2 新的阀值是一个大于或等于2的自然数，它要小于或等于DSD角色集的基（基：有限集元素的个数）；  3 新的DSD约束当前应该是被满足的。 |
| List<DSDRoleState> DSDRoleSets() | 该函数返回所有的DSD角色集。 |
| List<RoleState> DSDRoleSetRoles  (Guid dsdSetID) | 该函数返回给定的DSD角色集中的角色。该函数可用当且仅当该角色集存在。 |
| int DSDRoleSetCardinality  (Guid dsdSetID) | 该函数返回与给定DSD角色集关联的阀值。该函数可用当且仅当该角色集存在。 |

Anycmd会完整的支持来自国籍标准的IRBACService接口，熟悉了这套接口也就学会了使用Anycmd。同时Anycmd也会提供一套稳定的、风格一致的、在AC领域近乎功能完备的、流畅的api。这些会在发布1.0版本的时候完成。

**anycmd是一个.net平台的完全开源的，完整支持RBAC的，将会支持xacml的通用的权限框架、中间件、系统。**

**如果您感兴趣的话现在可以先观察Anycmd的源码，期待您为Anycmd.AC提供帮助确保她走在正确的道路上。**

**Anycmd的开源地址在**[**http://git.oschina.net/anycmd/anycmd**](http://git.oschina.net/anycmd/anycmd)