

CNGrid GOS 3.2 系统软件接口

WP 2.1

关键字：GOS，系统软件，接口

文档计划提交时间：无 文档实际提交时间：2008 年 5 月 8 日
文档类别：工作文档 文档版本号：0.9.6，生成于：2008 年 5 月 7 日
文档负责人：查礼，char@ict.ac.cn
文档编辑人及其联系方式：
查礼
中科院计算所
北京科学院南路六号
char@ict.ac.cn，010-6260-0969

文档发布范围：

公开 (Public)	对全社会公开	
项目 (Program)	对“高效能计算机及网格服务环境”重大项目公开	
课题 (Project)	对“中国国家网格软件研究与开发”课题公开	
工作包 (WP)	对特定工作包公开	X
任务 (Task)	对特定工作包中的特定任务公开	

文档修改历史：

版本号	编辑人及单位	编辑时间	备注 (修改了什么)
0.1	岳强、查礼 / 计算所	2007.06.04	确定文档结构与格式
0.2	邹永强/计算所	2007.06.08	完成核心中用户、资源和社区相关接口
0.3	王小宁/计算所	2007.06.09	完成全部的控制台命令，给出应用相关接口
0.4	汪浩、李沛旭/计算所	2007.06.09	添加了网程相关接口
0.5	程伯群，王小宁，周浩杰，邹永强等 / 计算所	2007.06.10	添加了批作业子系统相关接口，文档格式有所调整，修复了控制台界面相关的错误
0.6	周浩杰/计算所	2007.06.10	加入安全相关的接口
0.7	邹永强/计算所	2007.06.15	调整用户、资源、社区接口、授权与访问控制接口，去掉批作业相关接口
0.8	岳强/计算所	2007.06.18	加入异常相关的接口
0.8.1	王小宁/计算所	2007.06.19	修改命令行相关部分
0.8.2	岳强/计算所	2007.06.21	修改和补充异常相关的接口
0.8.3	邹永强/计算所	2007.06.21	各处细节错误的修改
0.8.4	邹永强/计算所	2007.06.30	各处细节的修改

0.8.5	岳强,王小宁/计算所	2007.07.02~04	各处细节的修改
0.9.3	邹永强、王小宁、李沛旭、熊飞、鲁小忆/计算所	2007.12.18	根据代码完成后设计复审的结果修改
0.9.4	邹永强	2008.1.14	根据核心测试与改进的结果微调
0.9.5	邹永强	2008.3.24	删除掉暂时没有实现的文件和安全相关的API, 以免引起误解。
0.9.6	邹永强	2008.5.7	发布前最后的检查与补充。
0.9.7	鲁小亿、熊飞、林健	2008.12.30	GOS3.2 版发包前检查及修改。

摘要: 本文档重点描述了 GOS3 的外部接口和内部接口。外部接口供 GOS 之上的用户使用, 重点讲述控制台界面和调用接口, 前者供最终用户使用, 后者提供 API 给 GOS 之上的应用开发人员使用。内部接口是 GOS 内部各模块之间的调用接口。

目录

CNGrid GOS 3.2 系统软件接口 WP 2.1	1
1 控制台界面.....	7
1.1 核心相关命令.....	7
1.1.1 用户相关命令.....	7
1.1.2 资源相关命令.....	8
1.1.3 社区相关命令.....	9
1.1.4 网程相关命令.....	10
1.1.5 异常相关命令.....	11
1.2 系统相关命令.....	11
1.2.1 文件系统相关命令（这里介绍的命令目前版本还不支持）	11
1.2.2 批作业子系统相关命令.....	12
1.2.3 CA 相关工具与命令	12
1.3 应用相关命令.....	12
1.4 其他.....	12
2 图形界面.....	13
3 调用接口（外部）	13
3.1 核心相关接口.....	13
3.1.1 用户相关接口.....	13
3.1.2 资源相关接口.....	18
3.1.3 社区相关接口.....	24
3.1.4 系统配置接口.....	32
3.1.5 网程相关接口.....	34
3.1.6 安全相关接口（在 GOS_3_2 版中不提供）	38
3.1.7 异常相关接口.....	44
3.2 系统相关接口.....	50
3.2.1 文件管理系统相关接口（这里介绍的接口目前版本还不支持）	50
3.2.2 动态部署服务.....	57
3.2.3 消息服务.....	58
3.2.4 CA 服务	59
3.2.5 批作业子系统.....	59
3.3 应用相关接口.....	59
3.3.1 通过 Resolver 机制使用资源.....	59
3.3.2 通过实现 Resovler 接口扩展解析过程.....	59
4 内部接口（模块之间）	60
4.1 命名系统.....	60
4.2 资源控制器.....	63
5 服务描述.....	65
5.1 服务说明.....	65
5.1.1 用户服务操作列表.....	66
5.1.2 资源服务操作列表.....	66
5.1.3 社区服务操作列表.....	66

5.1.4	Naming 服务操作列表.....	66
5.2	WSDL.....	67
5.3	部署描述.....	67
5.3.1	异常相关部署.....	67
6	参考文献.....	67

图目录

无

表目录

表 1-1 用户相关命令列表	8
表 1-2 资源相关命令列表	9
表 1-3 社区相关命令列表	10
表 1-4 网程相关命令列表	11
表 1-5 异常相关命令列表	11
表 1-6 文件相关命令列表	12
表 1-8 应用相关命令列表	12
表 1-9 其他命令列表	12
表 3-1 用户调用接口列表	14
表 3-2 资源调用接口列表	19
表 3-3 社区调用接口列表	24
表 3-4 系统配置接口列表	33
表 3-4 网程调用接口列表	35
表 3-5 安全调用接口列表	38
表 3-6 异常调用接口列表	47
表 3-7 文件调用接口列表	51
表 3-8 动态部署调用接口列表	58
表 3-9 ResolveClient 调用接口列表	59
表 4-1 Naming 内部接口列表	61
表 4-2 RController 内部接口列表	63
表 5-1 用户服务操作列表	66
表 5-2 社区服务操作列表	66
表 5-4 Naming 服务操作列表	67

GOS3 的接口可以分为外部接口、内部接口和服务描述。外部接口主要包括控制台界面、图形界面和调用接口。控制台界面和图形界面可以给最终用户使用，调用接口提供 API 给 GOS 之上的应用开发人员使用。内部接口是 GOS 内部各模块之间的调用接口。此外，本文档还给出部分模块提供的 Web 服务描述。

1 控制台界面

GOS3 提供了一个类似 unix 中 bash 的 shell 操作界面，即 GShell，简称 gsh。GShell 支持命令行操作，用于管理和使用 GOS。

本文描述控制台界面时，给出如下约定：

- 本文档中无特殊说明处，所指的命令都是在 Gshell 中运行的 GOS 命令（在启动 gshell 程序的时候使用了 -m 选项，则以 grun 作为第一个分词的命令认为是 GOS 命令，否则不在 local 模式下运行的命令认为是 GOS 命令。）
- 命令名字字符串区分大小写；通常核心相关的 GOS 命令以 G 开头，紧接着的第一个缩写词均小写，后面的每一个缩写词均首字母大写。
- 本文命令集的表格有三列，其中“参数描述”列中每一行是一个参数或者选项，选项以“-”开始，如果不是开关选项空格后可设置选项取值，带[]的是可选的选项。以下述表格为例：

命令名	功能描述	参数描述
GdelGroup	在当前社区删除指定用户组	用户组的名字 [-h] 输出命令使用格式

控制台命令可分为核心相关命令、系统相关命令、应用相关命令和其他命令。核心相关命令指操作 GOS 核心的命令，包括用户、资源、社区、网程相关命令。系统相关命令指 GOS 各个系统级服务提供的命令，包括文件系统相关命令和批作业子系统相关命令。应用相关命令主要指支持应用开发、运行等的命令。

全部的命令集在后文依次描述。

1.1 核心相关命令

1.1.1 用户相关命令

命令名	功能描述	参数描述
GuserInfo	查看当前社区的用户信息	[-h]输出命令使用格式 [-n name]指定查看的用户名 [-r GRID AGORA]指定察看范围为当前社区还是整个网格 [-l] 分别列出用户详细信息
GaddUser	在当前社区中添加一个用户	[-h]输出命令使用格式 [-l userName] 表示接入已存在的用户
GdelUser	在当前社区删除指定用户	[-h]输出命令使用格式 username 网格用户名

GchgUserAttr	修改当前社区指定用户的元信息	[-h]输出命令使用格式 [username] 网格用户名 [-n AttrName=NewValue]修改指定元信息的值, 如果不指定该项, 给出一个元信息列表
GchgUserProxy	修改当前社区指定用户的 proxy	[-h]输出命令使用格式 [username] 网格用户名 proxyfile 用户 proxy 文件路径
GchgPassword	修改当前社区指定用户的密码 (要求用户确认输入)	[-h]输出命令使用格式 [username] 网格用户名
gexit	退出网格系统, 这是一个内部命令	
gregistUser	申请注册网格用户 (通过交互完成注册过程) 这个命令目前版本还未实现	

表 1-1 用户相关命令列表

1.1.2 资源相关命令

命令名	功能描述	参数描述
GresInfo	查看当前社区的资源信息	[-h]输出命令使用格式 [-i guid]指定查看资源的 GUID [-n name]指定查看的资源别名 [-t type]指定查看的资源类别 [-o user]指定查看 owner 是 user 的资源 [o1 introducer]加资源的动作实施者 [-l] 分别列出资源详细信息 [-r GRID AGORA]指定查找范围为当前社区还是整个网格
GaddRes	在当前社区创建一个资源 (通过命令交互获取创建资源的基本信息)	[-h]输出命令使用格式
GdelRes	在当前社区删除指定资源 (执行的时候会给出指定名字的资源列表, 询问用户删除哪一个)	[-h]输出命令使用格式 [-i guid]指定修改社区的 GUID [-n name]指定资源名字 [-t type] 指定资源类型 [-o user] 指定资源 owner [-o1 introducer]指定资源引入者
GchgResAttr	修改当前社区指定资源的元信息	[-h]输出命令使用格式 [resname] 资源别名 [-i guid]指定修改资源的 GUID [-n AttrName=NewValue]修改指定元信息

		的值, 如果不指定该项, 给出一个元信息列表
GexportRes	共享资源到其他社区	<p>[-h]输出命令使用格式</p> <p>[resname] 资源别名</p> <p>agoralist 社区列表, 格式如下:</p> <pre> export list: agoraitem (; goraitem)* ; agoraitem: i:guid name '*' i:giud:s name:s '*':s </pre> <p>其中带引号的*代表所有社区, 社区后面跟有 s 项的是自我管理模式, 即进入其他社区后, 仍然是此资源的 owner 来设置其权限, 否则是目标社区的管理员设置权限。</p> <p>[-i guid] 指定资源的 GUID</p> <p>[-u] 是开关选项, 如果有-u 选项, 则表示收回共享权限。</p>
GimportRes	导入其他社区共享的资源到当前社区 (如果对于引入的资源 资源 owner 取消了其对当前社区的 export 权限, 此引入的资源将空悬)	<p>[-h]输出命令使用格式</p> <p>[resname] 资源别名</p> <p>[-f agoraname] 原社区名字</p> <p>[-i guid]指定资源的 guid</p>

表 1-2 资源相关命令列表

1.1.3 社区相关命令

命令名	功能描述	参数描述
GagoraInfo	查看系统软件中存在的社区的基本信息	<p>[-h]输出命令使用格式</p> <p>[-i guid]指定查看社区的 GUID</p> <p>[-n name]指定查看的社区名</p> <p>[-o user]指定查看管理员是 user 的社区</p> <p>[-l] 分别列出社区基本信息</p>
GaddAgora	创建一个社区	[-h]输出命令使用格式
GdelAgora	删除指定社区 (执行的时候会给出指定名字的社区列表, 询问用户删除哪一个)	<p>[-h]输出命令使用格式</p> <p>[agoraname] 社区名字</p> <p>[-o ownername] 指定社区的 owner</p> <p>[-i guid]指定社区的 GUID</p>
GchgAgoraAttr	修改社区的基本信息 (执行的时候会给出指定名字的	<p>[-h]输出命令使用格式</p> <p>[agoraname]社区名</p>

	社区列表，询问用户操作哪一个)	[-i guid]指定修改社区的 GUID [-n AttrName NewValue]修改指定元信息的值，如果不指定该项，给出一个元信息列表
GchgAgoraProxy	修改社区的 proxy 信息（执行的时候首先显示原 proxy 值，再要求用户设定新值）	[-h]输出命令使用格式 [aroganame]社区名 [-i guid]指定修改社区的 GUID proxyfile 更新的社区 proxy
GwAgora	查看当前所在的社区	[-h]输出命令使用格式
GchgAgora	改变用户的当前社区（执行的时候会给出指定名字的社区列表，询问用户进入哪一个）	[-h]输出命令使用格式 [aroganame] 社区名 [-i guid]指定进入社区的 GUID
GgroupInfo	查看当前社区的用户组信息	[-h]输出命令使用格式 [-n groupname]指定查看的用户组名 [-u username]指定查看 user 所在的用户组 [-l] 分别列出用户组详细信息
GaddGroup	在当前社区中添加一个用户组	[-h]输出命令使用格式 [groupname] 用户组的名字
GdelGroup	在当前社区删除指定用户组	[-h]输出命令使用格式 groupname 用户组的名字
GchgGroupAttr	在当前社区修改指定用户组的信息	[-h]输出命令使用格式 groupname 用户组的名字 [-n AttrName=NewValue]修改指定元信息的值，如果不指定该项，给出一个元信息列表
GlistGroupUser	在当前社区中列出指定用户组中的用户	[-h]输出命令使用格式 [groupname] 用户组的名字 [-l]列出详细信息
GaddGroupUser	把用户加入指定的用户组	[-h]输出命令使用格式 username 用户名 groupname 用户组名
GdelGroupUser	从指定的用户组中删除特定用户	[-h]输出命令使用格式 username 用户名 groupname 用户组名

表 1-3 社区相关命令列表

1.1.4 网程相关命令

命令名	功能描述	参数描述
Gps	查看当前用户在当前社区中	[-t] 指定查看网程类型

	的网程执行信息	[-h] 输出命令使用格式 [-l] 分别列出网程详细信息 [-s] 获取网程所在节点 IP 和 Port [-u] 指定创建网程的用户
Gkill	结束当前社区中的指定网程	[-h]输出命令使用格式 gripID 指定结束的网程号

表 1-4 网程相关命令列表

1.1.5 异常相关命令

命令名	功能描述	参数描述
xxd	将 xml 异常描述文件 (file) 或路径中定义的异常转换成 java 接口文件 (javafile)。默认名为 javafile="ns+sec" 该命令可以执行在本地系统的 shell 环境中，即可以不以网程形式运行	-f file -p path 异常描述文件或者其路径 [-o javafile] java 接口文件

表 1-5 异常相关命令列表

1.2 系统相关命令

1.2.1 文件系统相关命令（这里介绍的命令目前版本还不支持）

命令名	功能描述	参数描述
glistFile	查看网格文件空间中当前位置的文件基本信息	[-n name]指定查看的文件名或者路径名 [-l] 分别列出文件基本信息
gcdFile	改变网格文件空间的当前位置	要进入的目录名
gmount	把本地目录空间加载到网格文件空间的指定位置	本地被 mount 的目录位置 网格文件空间的 mount 点
gumount	gmount 的反操作	网格文件空间的 mount 点
gmkdir	在网格文件空间中创建一个文件夹	创建文件夹的名字
grmFile	在网格文件空间中删除一个文件	删除文件的名字 [-r]递归删除
gupFile	上传文件到网格文件空间的指定位置	上传的文件在本地的位置和名字 网格文件空间中的目标位置

gdownFile	从网格文件空间中下载特定文件到本地	下载的文件在网格文件空间的位置 下载到本地的目标位置 [-O]设置 cache（类似 localize 起的作用）
gchangeFile	修改网格文件空间中指定文件的元信息	文件在网格文件空间中的名字 [-n AttrName NewValue]修改指定元信息的值，如果不指定该项，给出一个元信息列表

表 1-6 文件相关命令列表

1.2.2 批作业子系统相关命令

参见文献[2]。

1.2.3 CA 相关工具与命令

暂无。

1.3 应用相关命令

本节的命令在 GOS 3.0 中不提供。

命令名	功能描述	参数描述
Ggosc	Java 程序预编译	[-h] 输出命令使用格式 [-m mode] 预编译模式，取值 key 或者 name，默认是 name classfiles Java 类文件列表（本地路径）
Ggosr	程序预解析	[-h] 输出命令使用格式 symbolFiles 预解析的符号表列表（本地路径）

表 1-7 应用相关命令列表

1.4 其他

命令名	功能描述	参数描述
Ghelp	查看核心提供的命令和执行格式	
namingManager.gsh	查看网格中的节点拓扑情况以及各个节点信息	

表 1-8 其他命令列表

2 图形界面

GOS3 的各个模块默认均提供 portal 形式的图形界面。核心功能由系统管理 Portal 提供，批作业子系统提供批作业管理 portal、科学计算网关、文件服务 portal 等提供。

限于篇幅，本文档对图形界面不再描述。

目前版本的 GOS3 核心没有提供任何 portal 形式的图形界面，但是 HPCG 部分已经提供相应的管理 portal 和使用 portal，具体参见文献[2]。

3 调用接口（外部）

3.1 核心相关接口

核心中用户、社区、资源都需要引用 GNode 数据结构。

GNode 和 GNodeInfo: 代表一个全局对象（用户、社区或资源）的元信息。GNode 有 guid 唯一标识一个全局对象。每个 GNode 对象有字段 originGNodeInfo，指向一个 GNodeInfo 对象。如果 GNode 对象的 guid 等于其 originGNodeInfo 的 guid，则此 GNode 为对象型 GNode，否则为引用型 GNode。可以有多个引用型 GNode 共享同一个 GNodeInfo 对象。

3.1.1 用户相关接口

所在包名：org.gos.core.user.client

所在类名：UserClient

关键数据结构：

UserInfo: 代表一个用户相关的真正信息

UserHandle: 代表一个用户的引用，含有用户的很多元信息类型的 Attribute，通过它可以访问用户的真正的信息。其中第一项是用户的 guid。此对象继承自 GNode。

UserAuthStruct: 代表一个已通过认证的用户的消息

● 调用接口列表

返回类型	接口名称	简要说明
------	------	------

UserHandle	add	增加用户
	remove	删除用户
UserInfo	update	修改用户信息
UserInfo	view	查看此用户的信息
UserHandle	readAttribute	读取用户的属性信息，即元信息
UserHandle	writeAttribute	修改用户的属性信息，即元信息
UserHandle[]	searchAttribute	根据属性搜索满足条件的用户信息
OperateContext	login	用户登录
	logout	用户退出网格

表 3-1 用户调用接口列表

- 功能说明：增加用户

接口描述：`UserHandle add(UserInfo info, String userName, String ownerAgoraID) throws GosException`

输入参数：

`Info` 用户相关信息

`username` 给定用户名，必须全网格唯一

`ownerAgoraID` 用户加入的目标社区的 `guid`，如果为 `null`，则接入当前社区

输出：

代表此用户的元信息

抛出异常：

用户名已存在

证书已过期

调用说明：

- 功能说明：删除用户

接口描述：`void remove(String userID) throws GosException`

输入参数：

`userID` 指代需要操作的用户

输出：

无

抛出异常：

权限不够

用户不存在

调用说明：

- 功能说明：修改用户信息

接口描述：`UserHandle update(String userID, UserInfo newInfo) throws GosException`

输入参数：

`userID` 指代需要操作的用户

`newInfo` 给出新的用户信息。

输出：

`UserInfo` 修改后的用户信息

抛出异常：

权限不够

用户不存在

调用说明：

- 功能说明：查看此用户的信息

接口描述：`UserInfo view(String userID) throws GosException`

输入参数：

`userID` 指代需要操作的用户

输出：

用户的真正信息。如果用户不存在，返回 `null`。

抛出异常：

权限不够

用户不存在

调用说明：

- 功能说明：读取用户的属性信息，即元信息

接口描述：`UserHandle readAttribute(String userID) throws GosException`

输入参数：

`userID` 指代需要操作的用户

输出：

一个 `UserHandle` 对象，其中包含了各种用户属性。

抛出异常：

权限不够

用户不存在

调用说明：

- 功能说明：修改用户的属性信息，即元信息

接口描述：`UserHandle writeAttribute(String userID, UserHandle newHandle) throws GosException`

输入参数：

`userID` 指代需要操作的用户

`newHandle` 给出新的 `UserHandle` 信息。其中多出来的属性将会加上，减少的属性将删除，已存在的属性将更新值。

输出：

`UserHandle` 对象，即修改后的用户的元信息

抛出异常：
权限不够
用户不存在

调用说明：

- 功能说明：根据属性搜索满足条件的用户信息

接口描述：UserHandle[] searchAttribute (String searchCondition , Object[] values) throws GosException

输入参数：

searchCondition 支持嵌套的表达式形式的搜索条件。表达式左侧是 GNode 的数据成员，右侧用?代表变量，变量的值在 values 中依次给出，表达式中间是运算符。

对于 GNode 的数据成员，有两种情况需要注意：

- GNode 中有一个可供扩充的 Map 型属性 attributes，其中以 key-value 对形式存储的可扩展的属性，对于这种属性的搜索以 attributes['属性名']的形式。
- 由于 GNode 和 GNodeInfo 都有 guid 字段，因此，我们用 gn.guid 标识 GNode 的 guid，而用 gi.guid 标识 GNodeInfo 的 guid。

GOS 中支持如下运算符：

- 支持的比较运算符包括：=, >, <, like
- 支持的逻辑运算符包括：and, or, not
- 支持()改变优先级

合法的 searchCondition 的例子 1：

```
type=? and acl=? and version>? and attributes['keyword']=? and
* attributes['serviceCategory']=? and
attributes['attr\"2']=?
```

注意：例子中 type, acl 都是 GNode 中定义的数据成员。其中也对 attributes 中的 keyword 和 serviceCategory 根据条件进行了判断。

合法的 searchCondition 的例子 2：

```
gn.guid=? or gi.guid=?
```

注意：由于 GNode 和 GNodeInfo 都有 guid 字段，因此，我们用 gn.guid 标识 GNode 的 guid，而用 gi.guid 标识 GNodeInfo 的 guid。

Values：与 condition 中的?对应的变量的值，此对象数组的每个元素依次替代 searchCondition 中的? 形成真正的搜索条件。当 searchCondition 和 values 为 null 时，表明没有任何搜索条件，此时得到所有的结果。

输出：

符合条件的用户的列表，是 `UserHandle` 列表。如果没有符合条件的，则返回的列表长度为 0。

抛出异常：

调用说明：

- 功能说明：用户登录

接口描述：`OperateContext login(String username, String password, String agoraName) throws GosException`

输入参数：

`Username` 用户名

`Password` 密码

`agoraName` 要登录的社区名字，如果为 `null`，则登录到用户的 `ownerAgora`。

输出：

成功登录则得到 `OperateContext` 对象，其中包含了用户相关的各种关键信息。

抛出异常：

用户不存在

密码错误

调用说明：

- 功能说明：用户登录（使用 `proxy`），并更新 `proxy`。

接口描述：`OperateContext login(byte[] proxy, String agoraName) throws GosException`

输入参数：

`proxy` 代理证书

`agoraName` 要登录的社区名字，如果为 `null`，则登录到用户的 `ownerAgora`。

输出：

成功登录则得到 `OperateContext` 对象。

抛出异常：

不是合法的证书格式

证书已过期

调用说明：

用户提供一个合法的 `proxy`，完成登录，并更新 `proxy` 证书。

- 功能说明：用户退出网络

接口描述：`void logout(OperateContext oc) throws GosException`

输入参数：

`oc`: 登录时获得的用户信息。

输出：

无
抛出异常:

调用说明:

用户退出网格后, 将无法使用网格, 直到下一次登录。

3.1.2 资源相关接口

所在包名: org.gos.core.resource.client

所在类名: ResourceClient

关键数据结构:

ResourceHandle: 代表一个资源的引用, 含有资源的很多 **Attribute**, 通过它可以访问资源的真正的信息。其中第一项是资源的 **guid**。此对象继承自 **GNode**。

OperateSession: 代表某个资源的某个具有相同性质的资源操作状态。一个 **OperateSession** 内资源的绑定关系可以保持稳定, 访问控制信息也不变。位于 **org.gos.core.rc.client**。

RuntimeHandle: 代表资源的本次操作的运行时实体, 位于 **org.gos.core.rc.client**。GOS 3.0 中支持同步和异步两种调用方式, 但都是用同样的接口。

Class **RuntimeHandle** {

Object getResult();//获取 **RuntimeHandle** 中的结果, 对于异步调用方式, 这步会导致客户端程序阻塞直到调用结果返回。

public Throwable getException();//获取调用过程中的异常信息。

}

OperateContext: 代表一次操作的上下文, 位于 **org.gos.core.rc**。其数据结构定义如下:

private UserAuthStruct user = null;//代表本次访问资源操作的用户身份

private AgoraHandle agora = null;// 代表本次访问资源操作所运行的社区

private String gripId = null;//代表本次访问资源操作所在的网程, 如果不是以网程方式运行, 或者在资源端运行则为 **null**。

private byte[] token = null;//本次访问资源操作所使用的访问控制的 **token**, 此 **token** 必须由所运行的社区签发。

● 调用接口列表

返回类型	接口名称	简要说明
ResourceHandle	add	创建资源
	remove	删除资源
ResourceHandle	readAttribute	读取资源的属性信息, 即元信息
ResourceHandle	writeAttribute	修改资源的属性信息, 即元信息
ResourceHandle[]	searchAttribute	根据属性搜索满足条件的资源信息
OperateSession	open	申请使用资源

RuntimeHandle	execute	执行对资源的操作
	close	释放资源

表 3-2 资源调用接口列表

- 功能说明：创建资源

接口描述: ResourceHandle add(String rControllerType, Object[] resInfo, String agoraID, String ownerID) throws GosException

输入参数:

rControllerType 处理此资源的 RController 的类全路径名。

resInfo 此资源相关信息，具体格式由具体的资源 RController 确定。

agoraID 资源接入的目标社区，如果为 null，则接入当前社区

ownerID 资源接入时指定的 owner，如果为 null，则为当前用户

输出:

代表此资源的元信息

抛出异常:

调用说明:

创建资源主要目的是将资源接入到网格，以便后续的操作和使用。

- 功能说明：添加资源（传入 OperateContext context）

接口描述: ResourceHandle add(OperateContext context, String rControllerType, Object[] resInfo, String agoraID, String ownerID) throws GosException

输入参数:

Context 添加资源时所使用的操作上下文。

rControllerType 处理此资源的 RController 的类全路径名。

resInfo 此资源相关信息，具体格式由具体的资源 RController 确定。

agoraID 资源接入的目标社区，如果为 null，则接入当前社区

ownerID 资源接入时指定的 owner，如果为 null，则为当前用户

输出:

代表此资源的元信息

抛出异常:

调用说明:

创建资源主要目的是将资源接入到网格，以便后续的操作和使用。GOS 中所有的操作必须有操作上下文才能进行，默认情况下 GOS 会从当前网程自动获得操作上下文。本操作显式的传入一个操作上下文以支持灵活的用户身份支持与访问控制策略。

- 功能说明：删除资源

接口描述: void remove(String resourceID) throws GosException

输入参数:

resourceID 指代需要操作的资源

输出:

无

抛出异常：
权限不够
资源不存在

调用说明：

- 功能说明：删除资源

接口描述：`void remove(OperateContext context, String resourceID) throws GosException`

输入参数：

`Context` 删除资源时所使用的操作上下文。

`resourceID` 指代需要操作的资源

输出：

无

抛出异常：

权限不够

资源不存在

调用说明：

`GOS` 中所有的操作必须有操作上下文（`OperateContext`）才能进行，默认情况下 `GOS` 会从当前网程自动获得操作上下文。本操作显式的传入一个操作上下文以支持灵活的用户身份支持与访问控制策略。

- 功能说明：读取资源的属性信息，即元信息

接口描述：`ResourceHandle readAttribute(String resourceID) throws GosException`

输入参数：

`resourceID` 指代需要操作的资源

输出：

一个 `ResourceHandle` 对象，其中包含了各种资源属性。

抛出异常：

权限不够

资源不存在

调用说明：

- 功能说明：修改资源的属性信息，即元信息

接口描述：`ResourceHandle writeAttribute(String resourceID, ResourceHandle newHandle) throws GosException`

输入参数：

`resourceID` 指代需要操作的资源

`newHandle` 给出新的 `ResourceHandle` 信息。其中多出来的属性将会加上，减少的属性将删除，已存在的属性将更新值。

输出：

`ResourceHandle` 对象，即修改后的资源的元信息

抛出异常：
权限不够
资源不存在

调用说明：

这个操作非常重要，因为系统中资源的很多重要信息是写在属性中的。比如：

- **Acl**
给出访问控制信息。Acl 为 rwx rwx rwx 模式，分别表明对 owner, group, others 的读、写、执行方面的访问控制信息。
- **export**
将 export 出去的社区的 id，多个社区以,分割，如果为*，则表示任何社区。如果原来已经有对此社区的 export 信息，则覆盖原有信息。其格式为: agora1(self), agora2(self), agora3(delegate), agora4(self)
Self 和 delegate 分别表示是自主管理模式还是托管模式。如果是自主管理模式，则目标社区 import 资源后，其 owner 仍然是资源现在的 owner，因此还是此 owner 来设置访问控制权限。托管模式资源被 import 到新的社区后其 owner 是目标社区的管理员，由他设置此资源在目标社区中的访问控制权限。

- 功能说明：根据属性搜索满足条件的资源信息

接口描述：`ResourceHandle[] searchAttribute(String searchCondition, Object[] values) throws GosException`

输入参数：

searchCondition 支持嵌套的表达式形式的搜索条件。表达式左侧是 GNode 的数据成员，右侧用?代表变量，变量的值在 values 中依次给出，表达式中间是运算符。

对于 GNode 的数据成员，有两种情况需要注意：

- GNode 中有一个可供扩充的 Map 型属性 attributes，其中以 key-value 对形式存储的可扩展的属性，对于这种属性的搜索以 attributes['属性名']的形式。
- 由于 GNode 和 GNodeInfo 都有 guid 字段，因此，我们用 gn.guid 标识 GNode 的 guid，而用 gi.guid 标识 GNodeInfo 的 guid。

GOS 中支持如下运算符：

- 支持的比较运算符包括：=, >, <, like
- 支持的逻辑运算符包括：and, or, not
- 支持()改变优先级

合法的 searchCondition 的例子 1：

```
type=? and acl=? and version>? and attributes['keyword']=? and
*
attributes['serviceCategory']=? and
attributes['attr\"2']=?
```

注意：例子中 type, acl 都是 GNode 中定义的数据成员。其中也对 attributes 中的 keyword 和 serviceCategory 根据条件进行了判断。

合法的 searchCondition 的例子 2:

gn.guid=? or gi.guid=?

注意: 由于 GNode 和 GNodeInfo 都有 guid 字段, 因此, 我们用 gn.guid 标识 GNode 的 guid, 而用 gi.guid 标识 GNodeInfo 的 guid。

Values: 与 condition 中的?对应的变量的值, 此对象数组的每个元素依次替代 searchCondition 中的? 形成真正的搜索条件。当 searchCondition 和 values 为 null 时, 表明没有任何搜索条件, 此时得到所有的结果。

输出:

符合条件的资源的列表, 是 ResourceHandle 列表。如果没有符合条件的, 则返回的列表长度为 0.

抛出异常:

调用说明:

- 功能说明: 申请使用资源

接口描述: OperateSession open(String resourceID, String option)

输入参数:

resourceID 指代需要操作的资源

Option 给出 open 时所需的参数, 具体如何取值由具体的资源决定

输出:

创建的 OperateSession 对象

抛出异常:

权限不够

资源不存在

调用说明:

- 功能说明: 申请使用资源

接口描述: OperateSession open(OperateContext context, String resourceID, String option)

输入参数:

Context 删除资源时所使用的操作上下文

resourceID 指代需要操作的资源

Option 给出 open 时所需的参数, 具体如何取值由具体的资源决定

输出:

创建的 OperateSession 对象。这个对象在 close 之前将一直可用。在一个 OperateSession 的生命周期中, OperateContext 也将保持不变。所以, 在一对 open 和 close 之间执行多个 execute 可以提高性能。但一对 open 和 close 之间的时间不宜过长, 因为这之间对用户、资源以及资源访问控制策略等的修改也不会生效, 而是沿用 open 那一刻的状态。

抛出异常：
权限不够
资源不存在

调用说明：

当要想以特定的用户身份在特定的社区中执行某个操作时，就在 `open` 中加入一个 `OperateContext` 对象。否则系统默认使用从当前网程中构造的 `OperateContext`。

● 功能说明：执行对资源的操作

接口描述：`RuntimeHandle Execute(OperateSession operateSession, String operationName, Object[] parameter, Boolean isSync)`

输入参数：

`operateSession` `open` 得到的 `OperateSession` 对象

`operationName` 操作名

`parameter` 参数列表

`isSync` 是否同步方式

输出：

返回运行实体的指代物（是一个 `RuntimeHandle` 对象，对于 `Web service`，可能代表了一个服务线程），通过 `RuntimeHandle` 的 `getResult()` 接口可以取得调用结果。同步时得到 `RuntimeHandle` 可立刻取得调用结果，异步调用时由 `RuntimeHandle` 的 `getResult()` 接口可能导致等待（`wait-by-necessary`）。

抛出异常：
权限不够
资源不存在

调用说明：

● 功能说明：释放资源

接口描述：`Void close(OperateSession operateSession)`

输入参数：

`operateSession` `open` 得到的 `OperateSession` 对象

输出：

无

抛出异常：

权限不够

资源不存在

调用说明：

3.1.3 社区相关接口

所在包名: org.gos.core.agora.client

所在类名: AgoraClient

关键数据结构:

AgoraInfo: 社区的真实信息

AgoraHandle: 代表一个社区的引用, 含有社区的很多 Attribute, 通过它可以访问社区的真实的信息。其中第一项是社区的 guid。此对象继承自 GNode。

GroupInfo: 组的信息。社区中的用户有分组以做更好的访问控制。

● 调用接口列表

返回类型	接口名称	简要说明
AgoraHandle	add	创建社区
	remove	删除社区
AgoraInfo	update	修改社区信息
AgoraInfo	view	读取此社区相关信息
AgoraHandle	readAttribute	读取社区的属性信息, 即元信息
AgoraHandle	writeAttribute	修改社区的属性信息, 即元信息
AgoraHandle[]	searchAttribute	根据属性搜索满足条件的社区信息
UserHandle[]	listUser	查看社区用户
ResourceHandle[]	listResource	查看社区资源
AgoraHandle[]	listAgora	查看全网格的社区
UserHandle	linkUser	接入用户到社区
	unlinkUser	使用户退出社区
ResourceHandle	linkResource	接入资源到社区
	unlinkResource	使资源退出社区
GroupInfo[]	listGroup	列出指定社区中的全部组
GroupInfo	addGroup	添加一个组
	removeGroup	删除一个组
GroupInfo	updateGroup	修改组信息
GroupInfo	viewGroup	查看组信息
	addGroupUser	将社区中用户加入到组
	removeGroupUser	删除组用户
String[]	listGroupUser	列出组用户
String[]	listUserGroup	列出用户所在的全部组

表 3-3 社区调用接口列表

● 功能说明: 创建社区

接口描述: AgoraHandle add(AgoraInfo info, String agoraName, String ownerID) throws GosException

输入参数:

info 此社区相关信息

agoraName 此社区的名字

ownerID 创建社区时指定的 owner, 如果为 null, 则为当前用户

输出:

代表此社区的元信息

抛出异常:

调用说明:

- 功能说明: 删除社区

接口描述: void remove(String agoraID) throws GosException

输入参数:

agoraID 指代需要操作的社区

输出:

无

抛出异常:

权限不够

社区不存在

调用说明:

- 功能说明: 修改社区信息

接口描述: AgoraInfo update(String agoraID, AgoraInfo newInfo) throws GosException

输入参数:

agoraID 指代需要操作的社区

newInfo 给出新的社区信息, 完全覆盖老的信息。

输出:

修改后的社区信息。

抛出异常:

权限不够

社区不存在

调用说明:

- 功能说明: 读取此社区的信息

接口描述: AgoraInfo view(String agoraID) throws GosException

输入参数:

agoraID 指代需要操作的社区

输出:

此社区的真实信息。如果社区不存在, 返回 null。

抛出异常:

权限不够

社区不存在

调用说明:

- 功能说明: 读取社区的属性信息, 即元信息

接口描述: `AgoraHandle readAttribute(String agoraID) throws GosException`

输入参数:

`agoraID` 指代需要操作的社区

输出:

此社区的元信息。

抛出异常:

权限不够

社区不存在

调用说明:

- 功能说明: 修改社区的属性信息, 即元信息

接口描述: `AgoraHandle writeAttribute(String agoraID, AgoraHandle newHandle) throws GosException`

输入参数:

`agoraID` 指代需要操作的社区

`newHandle` 给出新的 `AgoraHandle` 信息。其中多出来的属性将会加上, 减少的属性将删除, 已存在的属性将更新值。

输出:

修改后的社区的元信息。

抛出异常:

权限不够

社区不存在

调用说明:

- 功能说明: 根据属性搜索满足条件的社区信息

接口描述: `AgoraHandle[] searchAttribute(String searchCondition) throws GosException`

输入参数:

`searchCondition` 支持嵌套的表达式形式的搜索条件。表达式左侧是 `GNode` 的数据成员, 右侧用 `?` 代表变量, 变量的值在 `values` 中依次给出, 表达式中间是运算符。

对于 `GNode` 的数据成员, 有两种情况需要注意:

- `GNode` 中有一个可供扩充的 `Map` 型属性 `attributes`, 其中以 `key-value` 对形式存储的可扩展的属性, 对于这种属性的搜索以 `attributes['属性名']` 的形式。
- 由于 `GNode` 和 `GNodeInfo` 都有 `guid` 字段, 因此, 我们用 `gn.guid` 标识 `GNode` 的

guid, 而用 gi.guid 标识 GNodeInfo 的 guid。

GOS 中支持如下运算符:

- 支持的比较运算符包括: =, >, <, like
- 支持的逻辑运算符包括: and, or, not
- 支持()改变优先级

合法的 searchCondition 的例子 1:

```
type=? and acl=? and version>? and attributes['keyword']=? and
    *
    attributes['serviceCategory']=? and
attributes['attr\"2']=?
```

注意: 例子中 type, acl 都是 GNode 中定义的数据成员。其中也对 attributes 中的 keyword 和 serviceCategory 根据条件进行了判断。

合法的 searchCondition 的例子 2:

```
gn.guid=? or gi.guid=?
```

注意: 由于 GNode 和 GNodeInfo 都有 guid 字段, 因此, 我们用 gn.guid 标识 GNode 的 guid, 而用 gi.guid 标识 GNodeInfo 的 guid。

Values: 与 condition 中的?对应的变量的值, 此对象数组的每个元素依次替代 searchCondition 中的? 形成真正的搜索条件。当 searchCondition 和 values 为 null 时, 表明没有任何搜索条件, 此时得到所有的结果。

- 功能说明: 查看社区用户

接口描述: UserHandle[] listUser(String agoraID)

输入参数:

agoraID 指代需要操作的社区

输出:

社区内的用户列表, 是 UserHandle 列表。

抛出异常:

权限不够

资源不存在

调用说明:

- 功能说明: 查看社区资源

接口描述: ResourceHandle[] listResource(String agoraID)

输入参数:

agoraID 指代需要操作的社区

输出:

社区内的资源列表, 是 ResourceHandle 列表。

抛出异常：
权限不够
资源不存在

调用说明：

- 功能说明：查看全网格的社区

接口描述：AgoraHandle[] listAgora()

输入参数：

无

输出：

全网格的社区列表，是 AgoraHandle 列表。

抛出异常：
权限不够
资源不存在

调用说明：

- 功能说明：接入用户到社区

接口描述：UserHandle linkUser(String userID, String agoraID)

输入参数：

userID 指代需要操作的用户

agoraID 指代目标社区

输出：

注册产生的引用型 UserHandle 对象

抛出异常：

权限不够

用户不存在

调用说明：

- 功能说明：使用户退出社区

接口描述：void unlinkUser(String userID, String agoraID)

输入参数：

userID 指代需要操作的用户

agoraID 指代目标社区

输出：

无

抛出异常：

权限不够

用户不存在

调用说明:

- 功能说明: 接入资源到社区

接口描述: `ResourceHandle linkResource(String resourceID, String agoraID)`

输入参数:

`resourceID` 指代需要操作的资源

`agoraID` 指代目标社区

输出:

注册产生的引用型 `ResourceHandle`

抛出异常:

权限不够

资源不存在

调用说明:

如果资源是以 `selfManage` 模式 `export` 的, 则 `link` 后资源 `owner` 还是原来的 `owner`, 如果资源原来 `owner` 没有在本社区, 则 `link` 资源失败。如果资源不是以 `selfManage` 模式 `export` 的, 则 `link` 后资源 `owner` 是本社区管理员, 由社区管理员设置访问控制权限。

- 功能说明: 使资源退出社区

接口描述: `void unlinkResource (String resourceID, String agoraID)`

输入参数:

`resourceID` 指代需要操作的资源

`agoraID` 指代目标社区

输出:

无

抛出异常:

权限不够

资源不存在

调用说明:

- 功能说明: 列出指定社区中的全部组

接口描述: `GroupInfo[] listGroup (String agoraID)`

输入参数:

`agoraID` 社区 ID

输出:

`GroupInfo` 对象的列表

抛出异常:

权限不够

社区不存在

调用说明:

- 功能说明：添加一个组

接口描述：GroupInfo addGroup (GroupInfo info)

输入参数：

Info 待创建的组信息，其中包含了其目标社区

输出：

添加的组的信息

抛出异常：

权限不够

组已存在

调用说明：

- 功能说明：删除一个组

接口描述：void removeGroup (String agoraID, String groupID)

输入参数：

AgoraID 组所在的社区 id

groupID 待删除的组的 id

输出：

无

抛出异常：

权限不够

组不存在

调用说明：

- 功能说明：修改组信息

接口描述：GroupInfo updateGroup (String agoraID, String groupID, GroupInfo info)

输入参数：

AgoraID 组所在的社区 id

groupID 待删除的组的 id

info 将修改的目标信息，目标 group 需要被替换。

输出：

修改后的组信息

抛出异常：

权限不够

组不存在

调用说明：

- 功能说明：查看组信息

接口描述：GroupInfo viewGroup (String agoraID, String groupID)

输入参数：

AgoraID 组所在的社区 id

groupID 待处理的组的 id

输出:

GroupInfo 组的信息。如果组不存在, 返回 null。

抛出异常:

权限不够

组不存在

调用说明:

- 功能说明: 将社区中用户加入到组

接口描述: void addGroupUser (String userID, String agoraID, String groupID)

输入参数:

UserID 待加入的用户的 id, 此用户必须在社区中

AgoraID 组所在的社区 id

groupID 待处理的组的 id

输出:

无

抛出异常:

权限不够

组不存在

用户已在组里

调用说明:

- 功能说明: 删除组用户

接口描述: void removeGroupUser (String userID, String agoraID, String groupID)

输入参数:

UserID 待删除的用户的 id

AgoraID 组所在的社区 id

groupID 待处理的组的 id

输出:

无

抛出异常:

权限不够

组不存在

用户不在组里

调用说明:

- 功能说明: 列出组用户

接口描述: String[] listGroupUser (String agoraID, String groupID)

输入参数:

AgoraID 组所在的社区 id

groupID 待处理的组的 id

输出:

用户的 guid 的列表

抛出异常:

权限不够

组不存在

调用说明:

- 功能说明: 列出用户在给定社区中所属的全部组

接口描述: `String[] listUserGroup (String userID, String agoraID)`

输入参数:

userID 用户的 id

AgoraID 给定的社区 id

输出:

用户在所属全部的组的 guid 的列表

抛出异常:

权限不够

组不存在

调用说明:

3.1.4 系统配置接口

所在包名: `org.gos.core.systemcfg.client`

所在类名: `SystemCfgClient`

关键数据结构:

SiteInfo: 代表一个节点的基本信息

LocalConfig: 代表一个节点的基本配置信息。比如此节点的节点管理员信息, 默认用户、默认社区, 所加入的网格的信息等。整个 **LocalConfig** 对象由一个个的 key-value 对形成。

- 调用接口列表

返回类型	接口名称	简要说明
SiteInfo	getLocalSiteInfo	获取本节点的信息
SiteInfo[]	listNeighbourSites	列出本节点的邻居
SiteInfo[]	listSiteInfo	列出网格的全部节点
LocalConfig	loadCfg	获得当前节点的配置信息
SiteInfo	updateSiteInfo	修改某个节点的信息
SiteInfo	viewSiteInfo	查看本节点的信息
String	getNamingUrlInCfgFile	获取当前节点的 naming 服务的 url

表 3-4 系统配置接口列表

- 功能说明：获取本节点的信息

接口描述：`public SiteInfo getLocalSiteInfo() throws Exception`

输入参数：

无

输出：

本节点的信息

抛出异常：

调用说明：

- 功能说明：列出本节点的邻居

接口描述：`public List listNeighbourSites() throws Exception`

输入参数：

无

输出：

本节点的邻居的列表，其中每一项是一个 `SiteInfo`。

抛出异常：

调用说明：

- 功能说明：列出网格的全部节点

接口描述：`public List listSiteInfo () throws Exception`

输入参数：

无

输出：

网格的全部节点列表，其中每一项是一个 `SiteInfo`。

抛出异常：

调用说明：

- 功能说明：获得当前节点的配置信息

接口描述：`public LocalConfig loadConfig() throws Exception`

输入参数：

无

输出：

本节点的配置信息。

抛出异常：

调用说明:

- 功能说明: 查看本节点的信息

接口描述: `public SiteInfo viewSiteInfo(String siteID) throws Exception`

输入参数:

siteID: 待查看的网格节点的 guid

输出:

网格节点信息

抛出异常:

调用说明:

- 功能说明: 修改本节点的信息

接口描述: `public SiteInfo updateSiteInfo(String siteID, SiteInfo siteInfo) throws Exception`

输入参数:

siteID: 待更新的网格节点的 guid

siteInfo: 网格节点的新信息, 这个对象将覆盖原有 SiteInfo。

输出:

修改后的网格节点信息

抛出异常:

调用说明:

- 功能说明: 获取当前节点的 naming 服务的 url

接口描述: `public static String getNamingUrlInCfgFile() throws GosException`

输入参数:

无

输出:

当前节点的 naming 服务的 url

抛出异常:

调用说明:

由于整个系统是通过名字服务作为所有全局对象(用户、社区、资源)的存储系统,因此很多 API 都需要提供一个名字服务的 url。本操作即从配置文件获取本节点的名字服务的 url。

3.1.5 网程相关接口

所在包名: `org.gos.core.grip.client`

所在类名: `GripClient`

关键数据结构:

GripHandle 从该结构中可以获取网程的 ID, 和其对应的本地进程的 PID。

GripContext 获取或者设置网程上下文的时候, 以及查询网程运行状态的时候, 都会使用到该结构, 主要包括用户身份信息, 网程所运行的社区信息等网程上下文信息, 以及相应的本地进程的部分信息。

● 调用接口列表

返回类型	接口名称	简要说明
GripHandle	create	创建网程, 启动网格应用
String	getCurrentGripID	获得当前网程 gripId
GripContext	getCurrentGripContext	获得当前网程 Context
OperateContext	getCurrentThreadOperateContext	从 ThreadLocal 中获得当前线程 OperateContext
	setCurrentGripContext	设置当前网程 Context
	setThreadOperateContext	将当前线程 OperateContext 设置到 ThreadLocal 中
HashMap	gripStatus	获得当前节点所有网程信息
	kill	杀掉一个特定的网程, 同时释放网程资源空间中占有的资源。

表 3-5 网程调用接口列表

● 功能说明: 创建网程, 启动网格应用

接口描述:

GripHandle create(**String[]** parameters) **throws** Exception

输入参数: Java application 可执行代码, 或 Web application 打包 war 文件。

输出参数: 网程的 **GripHandle**。

抛出异常: 创建网程失败的异常。

GripHandle create(**UserAuthStruct** userAuthStruct, **AgoraHandle**

simpleAgoraHandle, **String[]** parameters) **throws** Exception

输入参数: 用户身份信息, 社区信息, Java application 可执行代码或 Web application 打包 war 文件。

输出参数: 网程的 **GripHandle**。

抛出异常: 创建网程失败的异常。

GripHandle create(**UserAuthStruct** userAuthStruct, **AgoraHandle** simpleAgoraHandle, **Map** property, **String[]** parameters) **throws** Exception

输入参数: 用户身份信息, 社区信息, 网程属性, Java application 可执行代码或 Web application 打包 war 文件。

输出参数: 网程的 **GripHandle**。

抛出异常: 创建网程失败的异常。

调用说明:

以上接口协调完成网程的创建以及网格应用的启动。主要针对 Java application 或 Web Application .war 文件创建的网程结构体和获得新的网程执行体; 在创建同时将获得的网程信息, 用户身份信息, 社区信息等发送到服务端保存, 包括该网程对应的网格用户的全局证书以及该用户在当前社区中的上下文; 最后在 GOS 中 Core 建立网格结构体与网格执行体之间的映射, 生成网程的全局 ID (两种可能的实现方法, 采用统一的 Naming 机制, 或者, 使用 URL+本地 pid (或者加上本地 tid) 的简单命名机制)。创建工作完成以后, 网格应用开始执行。

- 功能说明: 取得当前网程 gripID。

接口描述:

String getCurrentGripID() throws Exception

输入参数: 无

输出参数: 当前网程 gripID

抛出异常: 获取网程 gripID 失败, 抛出异常

调用说明:

以上口实现获取网程 gripID 的功能, 其中 getCurrentGripID 接口是提供给用户调用的, 而 getWebCurrentGripID 接口不对用户开放。在 getCurrentGripID 接口中调用了 getWebCurrentGripID 接口, 因此 getCurrentGripID 融合了获取两种类型网程 gripID: javaapp 型网程与 webapp 型网程。对 javaapp 型网程, 调用本地动态库方法, 获取当前进程的 pid, 利用此 pid 在本地日志文件中查找对应此 pid 的 grip, 若找到, 则返回 gripID, 否则, 使该线程睡眠一段时间; 对 webapp 型网程, 根据用户请求 request 对象获得相应 webapp 的网程 ID。

- 功能说明: 取得当前网程 Context。

接口描述: **GripContext** getCurrentGripContext() throws Exception

输入参数: 无

输出参数: 当前网程 Context

抛出异常: 获取网程 Context 失败, 抛出异常

调用说明:

该接口融合了获取 Javaapp 型与 Webapp 型网程的 Context。根据调用该接口时是网程类型的不同, 分别处理。若在 Webapp 环境中调用此接口, 则首先从线程变量 ThreadLocal 中取出本次调用的 HttpServletRequest 对象, 然后从 HttpServletRequest 对象取出对应 session, 最后从 session 中取出保存的用户认证信息 U 与社区信息 A, 将 U 设置到 gripContext 的 userAuthStruct 字段, A 设置到 simpleAgoraHandle 字段, 而该 Webapp 的初始用户信息保留在 iniUserAuthStruct 与 iniSimpleAgoraHandle 字段中; 若是在 Javaapp 中调用该接口: 首先调用接口取得当前网程 gripID, 以“读”模式调用服务端的服务去取得保存的 gripContext, 若 gripContext 不为空, 则经序列化后从服务端返回, 该返回的字符串需再经客户端反序列化。

- 功能说明: 取得当前线程级网程的 OperateContext。

接口描述: **OperateContext** getCurrentThreadOperateContext()

输入参数: 无

输出参数: 当前线程级网程 OperateContext

调用说明: 从 ThreadLocal 中获得与当前线程级网程相关的 OperateContext。

- 功能说明: 设置当前网程 Context

接口描述: **void** setCurrentGripContext(GripContext gripContext) **throws** Exception

输入参数: 网程 Context

输出参数: 无

抛出异常: 设置网程 gripcontext 失败, 抛出异常。

调用说明:

该接口主要目的有两个, 首先对外部用户查看网程信息提供了一个接口, 其次对内部模块提供修改网程数据结构的接口, 该接口向用户提供了更改当前网程 context 的功能, 以 write 模式调用部署在服务端的服务, 由此服务去更改保存的当前网程的 gripContext。

- 功能说明: 设置当前线程级网程 OperateContext

接口描述: **void** setThreadOperateContext(OperateContext gc)

输入参数: OperateContext

输出参数: 无

调用说明: 设置当前线程级网程的 OperateContext, 将 OperateContext 保存到 ThreadLocal 中。

- 功能说明: 获得当前节点所有网程信息

接口描述:

HashMap gripStatus() **throws** Exception

输入参数: 无

返回参数: 保存有网程状态信息的 HashMap。

抛出异常: 获取网程状态失败, 抛出异常。

HashMap gripStatus(**String** userOriginalID, **String** instructionType) **throws** Exception

输入参数: 创建网程的用户 ID, 网程的类型。

返回参数: 满足条件的、存有网程状态信息的 HashMap。

抛出异常: 获取网程状态失败, 抛出异常。

HashMap gripStatus(**String** userOriginalID, **String** instructionType, **String** siteIp, **String** sitePort) **throws** Exception

输入参数: 创建网程的用户 ID, 网程的类型、指定的网络节点 IP、指定的网络节点 port

返回参数: 满足条件的、存有网程状态信息的 HashMap。

抛出异常: 获取网程状态失败, 抛出异常。

接口说明:

第一个 gripStatus() 接口实际上是以默认参数 null 调用第三个 gripStatus(**String** userOriginalID, **String** instructionType, **String** siteIp, **String** sitePort) 接口实现的; 而第二个接口 gripStatus(**String** userOriginalID, **String** instructionType) 是以参数

siteIp、siteIp 为 null 调用第三个接口实现的。主要用于查看指定节点上、指定用户创建的、指定类型的的网程的信息，通过调用服务端相关服务实现。

- 功能说明：杀掉一个特定的网程，同时释放网程资源空间中占有的资源

接口描述：**void** kill(String gripID) **throws** Exception

输入参数：网程的 ID gripID, tomcat 管理员帐号、密码

输出参数：无

抛出异常：定义新的异常，表明用户无权杀死该网程。

调用说明：

该接口融合了对 javaapp 型与 webapp 型网程的操作,对 javaapp 网程杀掉本地对应进程,而对 webapp 网程则卸载掉服务端的 web 应用。两中类型都使用了内部接口 signal。另杀死网程的工作除了回收该网程结构体对应的内存数据结构以外,还需要回收资源空间中的资源,这个工作依赖于资源端是否部署有资源的 RController 支持最终的实现。统一的语义是只负责回收网格应用所占有的资源,不负责回收网格资源端的资源。

3.1.6 安全相关接口（在 GOS_3_2 版中不提供）

所在包名：org.gos.core.security.client

关键数据结构：

无

- 调用接口列表

返回类型	接口名称	简要说明
	editUser	注册 CA 用户/修改用户属性
List	findUser	查询 CA 注册用户
Certificate	generateCertificate	为某个用户生成证书
	revokeCert	注销证书
RevokeStatus	checkRevocationStatus	检查证书是否被注销
	revokeUser	注销用户
PubKeyCertInfo	PubKeyCertParse	解析身份证书
String	generatePorxy	生成代理证书
ProxyPubKeyCertInfo	ProxyPubKeyCertParse	解析代理证书
Int	ProxyPubKeyCertVerify	根据代理证书、证书链验证代理证书有效性
Int	ProxyPubKeyCertVerify	根据代理证书、证书链和 CRL 验证代理证书有效性
boolean	acDecision	根据资源 GNode 信息进行访问控制决策
Byte[]	getToken	获取一个已签名的授权令牌
boolean	verifyToken	验证授权令牌有效性
boolean	acEnforce	根据授权令牌进行访问控制实施
OperateContext	getOperateContext	获取当前操作上下文

表 3-6 安全调用接口列表

3.1.6.1 CA 服务

(此功能在 GOS 3.0 中 WP2 不再提供)

- 功能说明: 注册 CA 用户/修改用户属性

接口描述: void editUser (CAUser auser)

输入参数: auser - CAUser CA 注册用户类型, 包括 Username, Password, DN, CAName, Email, Status, TokenType 等属性。

输出: 无

抛出异常: 网络错误。
非授权调用。

调用说明: 修改 CA 注册用户, 若没有找到对应用户名, 则新注册一个用户。

- 功能说明: 查询 CA 注册用户

接口描述: List findUser (UserMatch userMatch)

输入参数: userMatch - UserMatch 查询用户类, 用来表示查询条件。

输出: 命中用户列表

抛出异常: 网络错误。
非授权调用。

调用说明:

- 功能说明: 为某个用户生成证书

接口描述: Certificate generateCertificate (String username, String pwd, String certificaterequest)

输入参数: 注册用户名, 密码, Base64 编码的证书签名请求

输出: X509 证书。

抛出异常: 网络错误。
非授权调用。

调用说明:

- 功能说明: 注销证书

接口描述: void revokeCert (String issuerDN, String serno, String reason)

输入参数: issuerDN, 签发者 DN, serno, 证书序列号, reason, 注销原因。

输出: 无

抛出异常: 网络错误。
非授权调用。
没有找到签发者或证书序列号。

调用说明:

- 功能说明: 检查证书是否被注销

接口描述: RevokeStatus checkRevocationStatus (String issuerDN, String serno)

输入参数: issuerDN, 签发者 DN, serno, 证书序列号

输出: 证书状态。

抛出异常：网络错误。
非授权调用。
没有找到签发者或证书序列号。

调用说明：

- 功能说明：注销用户

接口描述：void revokeUser (String Username, String reason)

输入参数：Username 注册用户名，reason 注销原因

输出：无

抛出异常：网络错误。
非授权调用。
找不到对应用户。

调用说明：

3.1.6.2 证书管理

(此功能在 GOS 3.0 中 WP2 不再提供)

- 功能说明：解析身份证证书

接口描述：PubKeyCertInfo PubKeyCertParse (String Cert) throws CertHandlerException

输入参数：cert-String 待解析身份证证书 (X.509V3 格式的 Base64 编码)

输出：PubKeyCertInfo

抛出异常：

调用说明：CertHandlerException

- 功能说明：生成代理证书

接口描述：String generatePorxy (String ProxyCert, String Cert, String Key, long time) throws CertHandlerException

输入参数：ProxyCert 未签名代理证书

Cert 用户公钥

Key 用户私钥

Time 代理证书有效期

输出：代理证书(字符串形式，可以保存到文件)

抛出异常： CertHandlerException

调用说明：

- 功能说明：解析代理证书

接口描述：ProxyPubKeyCertInfo ProxyPubKeyCertParse (String ProxyCert) throws CertHandlerException

输入参数：ProxyCert 未签名代理证书

输出：String 证书 DN

抛出异常： CertHandlerException

调用说明：

- 功能说明：根据代理证书、证书链验证代理证书有效性

接口描述：`Int ProxyPubKeyCertVerify(String ProxyCert, String[] CertChain) throws CertHandlerException`

输入参数：`ProxyCert` 未签名代理证书
`CertChain` 证书链

输出：

抛出异常：`CertHandlerException`

调用说明：

- 功能说明：根据代理证书、证书链和 CRL 验证代理证书有效性

接口描述：`Int ProxyPubKeyCertVerify(String ProxyCert, String[] CertChain, String CRL) throws CertHandlerException`

输入参数：`ProxyCert` 未签名代理证书
`CertChain` 证书链
`CRL` 已撤销证书列表

输出：

抛出异常：`CertHandlerException`

调用说明：

3.1.6.3 授权与访问控制

（此功能在 GOS 3.0 中 WP2 不再提供）

所在包名：`org.ict.gos.core.security.client`

所在类名：`SecurityClient`

这些功能由社区的授权权威提供。

- 功能说明：根据资源 `GNode` 信息进行访问控制决策

接口描述：`boolean acDecision (UserHandle userHd, AgoraHandle agoraHd, GNode gnode, String operationName)`

输入参数：

`userHd` 执行操作的用户
`agoraHd` 此次操作所在的社区
`gnode` 此次操作的资源对象
`operationName` 操作名

输出：`true or false` 访问控制通过则返回 `true`，否则返回 `false`。

抛出异常：

各种不应该出现的情况则报告异常

调用说明：

- 功能说明：签发授权令牌

接口描述：`byte[] getToken(UserHandle userHandle, AgoraHandle agoraHandle, GNode gnode, String operation)`

调用说明:

3.1.6.4通信安全

(此功能在 GOS 3.0 中 WP2 不再提供)

- 功能说明: 对 soap 消息进行签名

接口描述: SOAPMessage sign(MessageContext msgContext, GSSCredential cred, SOAPEnvelope unsignedEnvelope)

输入参数: msgContext 消息
cred 用于签名的证书
unsignedEnvelope 未签名的消息

输出: 签名后的 soap 消息

抛出异常:

调用说明:

- 功能说明: 验证消息签名

接口描述: boolean verifySign (MessageContext message, WSSecurityEngine engine)

输入参数: message 消息
Engine- WSSecurityEngine 用于签名的证书

输出: true or false

抛出异常:

调用说明:

3.1.6.5安全上下文

(此功能在 GOS 3.0 中 WP2 不再提供)

关键数据结构: OperateContext

此数据结构的获取依赖于资源的类型, 具有由处理此资源的 RController 提供。GOS 3.0 中, 由 ForAxis 类负责在服务实现内获取上下文, 由 ForJms 负责在消息中获取上下文。

org.gos.core.rc.axis 包中的 ForAxis 类

org.gos.core.rc.activemq 包中的 ForJms 类

- 功能说明: 获取当前上下文

接口描述: OperateContext getOperateContext();

输入参数:

无

输出:

代表当前环境的 OperateContext 对象

抛出异常:

无

调用说明:

3.1.7 异常相关接口

所在包名: org.ict.x2d
org.ict.x2d.*

3.1.7.1 x2D 文档类型定义 (DTD)

一个 x2D 文档类型定义 (DTD) 如下:

```
<?xml version="1.0"?>
<!-- x2D 一个 x2D 块信息的标记 -->
<!ELEMENT x2D (ds | sec | exception)*>
<!-- ds, sec 分别代表一个异常定义空间和一个分段 -->
<!ELEMENT ds (sec)*>
<!ATTLIST name CDATA #REQUIRED>

<!ELEMENT sec (trace | exception)*>
<!ATTLIST name CDATA #REQUIRED
begin (0-9)* #REQUIRED
end (0-9)* #REQUIRED >

<!ELEMENT trace CDATA>

<!-- exception 代表一个异常描述 -->
<!ELEMENT exception >
<!ATTLIST exception id (0-9)* #REQUIRED
desc CDATA #IMPLIED
type (0-7) #IMPLIED>
```

3.1.7.2 x2D 异常描述举例

```
1 <?xml version="1.0"?>
2 <x2D>
3 <ds name="com.ds1.ds2">
4 <sec name="sec0" begin="0" end="100">
5 <trace>com.ds1</trace>
6 <trace>com.x2d</trace>
7 <trace>java</trace>
8 <exception id="1" desc="d0" type="0"/>
```

```

9      <exception id="2" desc="d2" type="2"/>
10 </sec>
11 </ds>
12 </x2D>

```

3.1.7.3异常信息输出格式

对于用户/调用者来说，单一异常消息应该提供必要的异常描述信息，异常栈信息，标识结构，处理单元描述信息等；多级异常消息应该提供有序的异常标识和异常信息组合。

3.1.7.4关键数据结构

- 异常定义空间结构

```

public class x2D_Space{
    public x2D_Space parent;           //父空间
    public x2D_Section psec;          //父分段
    public String sp_name;            //空间名
}

public class x2D_Section{
    public x2D_Space sc_space;         //所在空间
    public String sc_name;            //分段名
    public int begin;                 //起始
    public int end;                   //结束
    public Vector sc_trace;           //过滤条件表
}

```

- 异常描述结构

```

public class x2Dexception {
    public int x_id;                   //异常 id
    public String x_desc;              //描述
    public int x_type;                //类型
}

```

```

public x2D_Section x_sec; //所属 section

public String x_dsnsec; //所属名字空间
}

```

- 异常消息结构

```

public class xxMessage {

    HashMap ms_stacktrace;

    HashMap ms_fault;

    HashMap ms_request;

    HashMap ms_response;

}

```

3.1.7.5调用接口列表

class/interface	主要构造函数/域/方法	说明
package org.ict.x2d		
xxd	public static void main(String[] argv);	Xxd 命令主函数，用于解析用户 xml 异常描述文件。
x2Dinit	(1) x2Dinit(); (2) x2Dinit(String[] dsnsec, boolean persistent);	用于初始化异常定义段及其异常描述
x2Drelease	x2Drelease(String[] dsnsec, boolean recursive);	用于释放异常定义段及其异常描述所占用的映射空间
x2Dexception	(1) x2Dexception(Error e); (2) x2Dexception(Exception e); (3) x2Dexception(int id); (4) x2Dexception(int id, String[] dsnsec); (5) x2Dexception(String xstr); (6) x2Dexception(x2Dexception e, String str);	用于构造 / 生成 x2Dexception。
package org.ict.x2d.deploy		
init_x2D	public void invoke(MessageContext msgContext);	提供外部初始化异常。
rel_x2D	public void invoke(MessageContext	提供外部释放异常。

	msgContext);	
package org.ict.x2d.message		
xxCollect	(1) xxCollect(Exception exp); (2) xxCollect(Exception exp, Object[] xmsg); (3) public Exception get();	用于接收远端异常/fault消息, 并通过 get 方法得到转换后的本地 x2Dexception 实例
xxDeliver	(1) xxDeliver(Exception exp); (2) xxDeliver(Exception exp, Object[] xmsg); (3) public Exception get();	用于传递远端 x2Dexception 消息, 并通过 get 方法得到转换后的异常/fault 实例
package org.ict.gos.common.exception		
Axis2GosEWrapper		兼容 Gos v2.0 异常
GosE2AxisWrapper		兼容 Gos v2.0 异常
GosException		兼容 Gos v2.0 异常

表 3-7 异常调用接口列表

3.1.7.6接口详细说明

- 功能说明: 初始化主空间及其保留段
接口描述 `public x2Dinit();`
输入参数: 无
输出: 无
抛出异常: 记录到指定的日志文件中。
调用说明: 调用主空间 (`InitMapping.rootNSpace`) 中保留段 (`InitMapping.xReserve`) 类的初始化方法 (`init`) 并完成其初始化。
- 功能说明: 初始化异常描述定义空间/段
接口描述: `public x2Dinit(String[] dsnsec, boolean persistent);`
输入参数:
`dsnsec`: 指定异常描述定义空间和分段名。对应与初始化类名 `dsnsec[0]` + “.” + `dsnsec[1]`。
`persistent`: 如果设置为 `true`, 则持久保存异常描述定义; 否则, 允许使用后释放所占用的异常定义空间。
输出: 无
抛出异常: 记录到指定的日志文件中。
调用说明: 一个异常定义描述为 `x2Dexception` 提供初始化异常信息。
- 功能说明: 释放异常描述定义空间
接口描述: `public x2Drelease(String[] dsnsec, boolean recursive);`
输入参数:
`dsnsec`: 指定异常描述定义空间和分段名。对应与初始化类名 `dsnsec[0]` + “.” + `dsnsec[1]`。

recursive: 如果设置为 **true**，则试图递归释放的所有从该段派生的异常描述扩展空间；否则，只释放该段所占用的异常定义空间。

输出：无

抛出异常：记录到指定的日志文件中。

调用说明：当所定义的异常描述段不再使用时调用。此外，`rel_x2D` 类提供默认的 `invoke` 方法。

- 功能说明：构造内部异常

接口描述一： `public x2Dexception(Error e);`

输入参数：**e:** 指定/捕获的 `java Error`。

输出：无

抛出异常：记录到指定的日志文件中。

调用说明：通常，用于在 `try-catch` 处理中对捕获到 `Error` 进行转换。

接口描述二： `public x2Dexception(Exception e);`

输入参数：**e:** 指定/捕获到 `java` 异常。

输出：无

抛出异常：记录到指定的日志文件中。

调用说明：通常，用于在 `try-catch` 处理中对捕获到 `Error` 进行转换。

接口描述三： `public x2Dexception(int id);`

输入参数：**id:** 指定在主空间中异常描述标识；

输出：无

抛出异常：记录到指定的日志文件中。

调用说明：用于生成/构造一个 `x2Dexception` 异常实例。

接口描述四： `public x2Dexception(int id, String[] dsnsec);`

输入参数：**id:** 指定在主空间中异常描述标识；

dsnsec: 指定该异常所在的空间名及其分段名，例如：`new String[]{"dsname", "secname"}`。

输出：无

抛出异常：记录到指定的日志文件中。

调用说明：用于生成/构造一个 `x2Dexception` 异常实例。

接口描述五： `public x2Dexception(String xstr);`

输入参数：**xstr:** 指定该异常的描述字符串。

输出：无

抛出异常：记录到指定的日志文件中。

调用说明：以 `xstr` 为描述生成/构造一个 `x2Dexception` 异常实例。

接口描述六： `public x2Dexception(x2Dexception e, String str);`

输入参数：**e:** 指定一个 `x2Dexception`；

str: 添加异常的描述字符串。

输出：无

抛出异常：记录到指定的日志文件中。

调用说明：为一个 `x2Dexception` 异常实例，添加进一步的描述信息。

- 功能说明：构造指定异常的 SOAP 消息

接口描述一： `public xxDeliver(Exception exp);`

输入参数：

`exp`：指定异常。

输出：无

抛出异常：记录到指定的日志文件中。

调用说明：当需要通过 SOAP 消息将本地异常信息传递给客户端调用程序时调用。

接口描述二： `public xxDeliver(Exception exp, Object[] xmsg);`

输入参数：

`exp`：指定 `x2Dexception` 异常。

`xmsg`：指定要传递的对象组。

输出：无

抛出异常：记录到指定的日志文件中。

调用说明：当需要通过 SOAP 消息将本地异常信息传递给客户端调用程序时调用。

接口描述三： `public Exception get();`

输入参数：无

输出：转换后的 SOAP 异常/fault 实例。

抛出异常：记录到指定的日志文件中。

调用说明：当需要抛出该 SOAP 消息时调用。

- 功能说明：接收远端 SOAP 异常/fault 消息

接口描述一： `public xxCollect(Exception exp);`

输入参数：

`exp`：指定异常。

输出：无

抛出异常：记录到指定的日志文件中。

调用说明：当需要把接收到 SOAP 异常/fault 消息转换成 `x2Dexception` 时调用。

接口描述二： `public xxCollect(Exception exp, Object[] xmsg);`

输入参数：

`exp`：指定 SOAP 异常/fault 实例。

`xmsg`：指定要传递的对象组。

输出：无

抛出异常：记录到指定的日志文件中。

调用说明：当需要把接收到 SOAP 异常/fault 消息转换成 `x2Dexception` 时调用。

接口描述三： `public Exception get();`

输入参数：无

输出：转换后的 `x2Dexception` 异常/fault 实例。

抛出异常：记录到指定的日志文件中。

调用说明：当需要抛出该异常时调用。

- 功能说明：GOS v2.0 类实现
为了兼容 GOS v2.0 的异常使用方法，x2d 提供了与 GOS v2.0 中相同的异常使用方式，主要是关于 GosException 类，具体使用方法参考相关 GOS v2.0 使用说明。

3.2 系统相关接口

3.2.1 文件管理系统相关接口(这里介绍的接口目前版本还不支持)

所在包名：org.gos.system.fms.client

所在类名：FmsClient

关键数据结构：

无

- 调用接口列表

返回类型	接口名称	简要说明
FMSFileType[]	list	查看当前目录下的文件/目录信息
FMSFileType	info	查看文件相关信息
boolean	exist	查看文件/目录是否存在
FMSFileType[]	search	按照关键字搜索文件
	rename	重命名文件空间内的文件
Quota	getQuota	获取用户配额信息
boolean	isFile	查看指定文件名是否是文件
boolean	isDir	查看指定文件名是否是目录
	mkdir	在文件空间内创建目录
	rmdir	在文件空间内删除目录
	create	在文件空间内创建文件
	delete	在文件空间内删除文件
	link	在文件空间内为文件/目录创建连接
	upload	上载本地文件到全局文件空间
	download	下载全局空间内文件到本地
	copy	拷贝文件空间中的文件/目录到指定位置
	move	移动文件空间中的文件/目录到指定位置
	createReplica	为文件空间中的文件创建只读副本
	localize	为文件空间中的文件创建只读副本到本地节点存储空间

	deleteReplica	删除文件副本
	auth	把文件空间的文件授权给用户/组/所有用户
	revoke	撤销对用户/组/所有用户的文件授权
FMSRight[]	authed	查询文件空间中文件的授权信息
StorageElement[]	listSE	列举系统中的存储服务
	addSE	为文件管理系统增加存储服务
	removeSE	从文件管理系统中删除存储服务
	modifySE	修改文件管理系统中的存储服务
StorageElement[]	getStorageElement	查询文件管理系统中的存储服务
	mount	挂载存储服务所在节点的目录到文件空间
	uMount	取消存储服务所在节点的目录到文件空间的挂载
RFTFile[]	rftList	查看用户传输中的文件
int	rftStatus	查看文件传输状态
boolean	rftDelete	取消文件传输

表 3-8 文件调用接口列表

3.2.1.1 元数据操作

- 功能说明：查看当前目录下的文件/目录信息

接口描述：FMSFileType[] list(String eName);

输入参数：eName-String 文件空间目录

输出：FMSFileType[]

抛出异常：权限异常/文件不存在

调用说明：

- 功能说明：查看文件相关信息

接口描述：FMSFileType info(String eName)

输入参数：eName-String 文件空间文件名

输出：FMSFileType

抛出异常：权限异常/文件不存在

调用说明：

- 功能说明：查看文件/目录是否存在

接口描述：boolean exist(String eName)

输入参数：eName-String 文件空间文件名

输出：true or false

抛出异常：无

调用说明：

- 功能说明：按照关键字搜索文件

接口描述：FMSFileType[] search(String keyWord);

输入参数：keyWord-String 关键字

输出：FMSFileType[]

抛出异常：无

调用说明：

- 功能说明：重命名文件空间内的文件

接口描述：void rename(String sourceENAME, String desENAME);

输入参数：sourceENAME-String 源文件名

desENAME-String 目标文件名

输出：null

抛出异常：无

调用说明：

- 功能说明：获取用户配额信息

接口描述：Quota getQuota();

输入参数：无

输出：Quota

抛出异常：无

调用说明：

- 功能说明：查看指定文件名是否是文件

接口描述：boolean isFile(String eName);

输入参数：eName-String 文件空间文件名

输出：true or false

抛出异常：无

调用说明：

- 功能说明：查看指定文件名是否是目录

接口描述：boolean isDir(String eName);

输入参数：eName-String 文件空间文件名

输出：true or false

抛出异常：无

调用说明：

3.2.1.2 文件操作

- 功能说明：在文件空间内创建目录

接口描述：void mkdir(String eName);

输入参数: eName-String 文件空间文件名

输出: null

抛出异常: 权限异常/父目录不存在

调用说明:

- 功能说明: 在文件空间内删除目录

接口描述: rmdir(String eName, boolean flag);

输入参数: eName-String 文件空间文件名

flag-boolean

如果为 true 则可以删除非空目录;

如果是 false 则只能删除空目录

输出: null

抛出异常: 权限异常/目录不存在

调用说明:

- 功能说明: 在文件空间内创建文件

接口描述: create(String eName);

输入参数: eName-String 文件空间文件名

输出: null

抛出异常: 权限异常/父目录不存在

调用说明:

- 功能说明: 在文件空间内删除文件

接口描述: delete(String eName);

输入参数: eName-String 文件空间文件名

输出: null

抛出异常: 权限异常/文件不存在

调用说明:

- 功能说明: 在文件空间内为文件/目录创建连接

接口描述: void link(String sourceENAME, String desENAME);

输入参数: sourceENAME-String 源文件名

desENAME-String 目标文件名

输出: null

抛出异常: 权限异常/源文件不存在/目标文件已经存在

调用说明:

- 功能说明: 上载本地文件到全局文件空间

接口描述: int upload(String localFile, String eName);

输入参数: eName-String 文件空间文件名

localFile-String 本地文件名

输出: int rft jobID

抛出异常：权限异常/父目录不存在
调用说明：

- 功能说明：下载全局空间内文件到本地
接口描述：int download(String eName,String localFile)
输入参数：eName-String 文件空间文件名
 localFile-String 本地文件名
输出：int rft jobID
抛出异常：权限异常
调用说明：

- 功能说明：拷贝文件空间中的文件/目录到指定位置
接口描述：void copy(String sourceEName, String desEName);
输入参数：sourceEName -String 源文件名
 desEName-String 目标文件名
输出：null
抛出异常：权限异常/源文件不存在/目标文件不存在
调用说明：

- 功能说明：移动文件空间中的文件/目录到指定位置
接口描述：void move(String sourceEName, String desEName);
输入参数：sourceEName -String 源文件名
 desEName-String 目标文件名
输出：null
抛出异常：权限异常/源文件不存在/目标文件不存在
调用说明：

3.2.1.3 副本管理

- 功能说明：为文件空间中的文件创建只读副本
接口描述：void createReplica(eName, StorageElement storageElement)
输入参数：eName-String 文件空间文件名
 storageElement- StorageElement 指定存储服务
输出：null
抛出异常：权限异常
调用说明：

- 功能说明：为文件空间中的文件创建只读副本到本地节点存储空间，并且下载到当前位置
接口描述：void localize(eName)
输入参数：eName-String 文件空间文件名

输出: null

抛出异常: 权限异常

调用说明:

- 功能说明: 删除文件副本

接口描述: void deleteReplica(String vName)

输入参数: vName -String 文件虚拟名

输出: null

抛出异常: 权限异常/副本文件不存在

调用说明:

3.2.1.4 权限管理

- 功能说明: 把文件空间内的文件授权给用户/组/所有用户

接口描述: void auth(String eName, int targetType, String target, String right);

输入参数: eName-String 文件空间文件名

targetType -int 授权客体类型 0 用户; 1 组; 2 其他用户

target-String 授权客体 DN

right- String 授权的权限

输出: null

抛出异常: 权限异常/文件不存在

调用说明:

- 功能说明: 撤销对用户/组/所有用户的文件授权

接口描述: void revoke(String eName, int targetType, String target);

输入参数: eName-String 文件空间文件名

targetType -int 授权客体类型 0 用户; 1 组; 2 其他用户

target-String 授权客体 DN

输出: null

抛出异常: 权限异常/文件不存在

调用说明:

- 功能说明: 取消对所有用户的文件授权

接口描述: void revoke(String eName);

输入参数: eName-String 文件空间文件名

输出: null

抛出异常: 权限异常/文件不存在

调用说明:

- 功能说明: 查询文件空间中文件的授权信息

接口描述: FMSRight[] authed(String eName);

输入参数: eName-String 文件空间文件名

输出: FMSRight[] FMSRight 是授权信息对象
抛出异常: 权限异常/文件不存在
调用说明:

3.2.1.5 数据源管理

- 功能说明: 列举系统中的存储服务

接口描述: StorageElement[] IstSE() ;

输入参数: 无

输出: StorageElement[]

抛出异常: 权限异常

调用说明:

- 功能说明: 为文件管理系统增加存储服务

接口描述: void addSE(StorageElement StorageElement) ;

输入参数: StorageElement- StorageElement 存储服务描述结构

输出: null

抛出异常: 权限异常

调用说明:

- 功能说明: 从文件管理系统中删除存储服务

接口描述: void removeSE(String storageElementID) ;

输入参数: storageElementID - String 服务 ID

输出: null

抛出异常: 权限异常

调用说明:

- 功能说明: 修改文件管理系统中的存储服务

接口描述: void ModifySE (StorageElement StorageElement) ;

输入参数: StorageElement- StorageElement 存储服务描述结构

输出: null

抛出异常: 权限异常

调用说明:

- 功能说明: 查询文件管理系统中的存储服务

接口描述: StorageElement[] getStorageElement (int type, String keyword)

输入参数: type-int 查询方式 0-根据 SEID; 1-根据 SE 名字; 2-根据 SE 地址

输出: StorageElement

抛出异常: 权限异常

调用说明:

- 功能说明: 挂载存储服务所在节点的目录到文件空间

接口描述: mount(String eName, StorageElement StorageElement, String DirName)

输入参数: eName-String 文件空间文件名
StorageElement- StorageElement 存储服务描述结构
DirName-String 存储服务节点所在节点目录名

输出: null

抛出异常: 文件空间权限异常/存储服务权限异常

调用说明:

- 功能说明: 取消存储服务所在节点的目录到文件空间的挂载

接口描述: uMount(String eName)

输入参数: eName-String 文件空间文件名

输出: null

抛出异常: 文件空间权限异常/存储服务权限异常

调用说明:

3.2.1.6 可靠数据传输

- 功能说明: 查看用户传输中的文件

接口描述: RFTFile[] rftList()

输入参数: null

输出: RFTFile[]

抛出异常: 无

调用说明:

- 功能说明: 查看文件传输状态

接口描述: int rftStatus(int JobID)

输入参数: JobID-int

输出: int 数据传输的比例; -1 为出错;

抛出异常: 无

调用说明:

- 功能说明: 取消文件传输

接口描述: boolean rftDelete(int JobID)

输入参数: JobID-int

输出: true or false

抛出异常: 无

调用说明:

3.2.2 动态部署服务

所在包名: org.gos.system.dynamicDeploy.client

所在类名: DdsClient

- 调用接口列表

返回类型	接口名称	简要说明
String	deploy	部署服务
	undeploy	卸载服务
	update	更新已部署服务
DeploymentDescription	info	查看此服务的信息

表 3-9 动态部署调用接口列表

- 功能说明：部署服务

接口描述：String deploy(DeploymentDescription dd)

输入参数：dd 描述要部署的服务的内容和部署参数

输出：新部署的服务的 URL

抛出异常：DynamicDeployException

调用说明：

- 功能说明：卸载服务

接口描述：void undeploy(String url)

输入参数：url 是要卸载的服务的 URL

输出：无

抛出异常：DynamicDeployException

调用说明：

- 功能说明：更新已部署服务

接口描述：void update(String url, DeploymentDescription dd)

输入参数：url 是要更新的服务的 URL

dd 描述用于更新服务的内容和部署参数

输出：无

抛出异常：DynamicDeployException

调用说明：

- 功能说明：查看服务信息

接口描述：DeploymentDescription info(String url)

输入参数：url 是要查看信息的服务的 URL

输出：相应服务的部署信息

抛出异常：DynamicDeployException

调用说明：

3.2.3 消息服务

GOS 中消息服务接口兼容 JMS (JSR-914) 标准。

3.2.4 CA 服务

参见 3.1.5 安全相关接口中 3.1.5.1 “CA 服务” 部分。

3.2.5 批作业子系统

参见文献[2]。

3.3 应用相关接口

3.3.1 通过 Resolver 机制使用资源

所在包名: org.gos.glinker.runtime

所在类名: ResolveClient

关键数据结构:

无

● 调用接口列表

返回类型	接口名称	简要说明
OperateSession	open	通过给定的资源描述（一般是资源名字）打开资源
RuntimeHandle	execute	执行资源的某个方法
void	close	关闭资源

表 3-10 ResolveClient 调用接口列表

这里接口的语法和语义唯一和 ResourceClient 提供的有所不同的是 open 的第一个参数，虽然都是 String 类型，但是在这个接口中，这个 string 可以是资源的名字，也可以是资源的 keyword。其他和 ResourceClient 提供的方法语法语义均保持一致，支持重新封装了一下。

3.3.2 通过实现 Resovler 接口扩展解析过程

所在包名: org.gos.glinker.preresolve

所在类名: Resolver

关键数据结构:

无

● 调用接口列表

返回类型	接口名称	简要说明
ResourceHandle	resolve	解析自定义名字的资源得到选择的结果
ResourceHandle[]	resolveGroup	解析自定义名字的资源得到全部的满足条件的结果

表 3-10 Resolver 调用接口列表

- 功能说明：解析自定义名字的资源得到选择的结果
 接口描述：ResourceHandle resolve (String rgroupname);
 输入参数：
 rgroupname, String 类型，给一组资源自定义的名字
 输出：
 ResourceHandle 对象，指代对应实体。如果没有符合条件的，则返回 null.
 抛出异常：
 NoResource 异常，当查找不到任何资源的时候给出该异常信息
 NoPrivilege 异常，用户对所有查找到的资源都没有权限操作
 调用说明：
 根据参数给定的名字在符号表文件中找到相应的解析条件，查找，并返回一个确定的实体，给出其 ResourceHandle，用于满足应用使用单个资源的需求。

- 功能说明：解析自定义名字的资源得到全部的满足条件的结果
 接口描述：ResourceHandle [] resolveGroup (String rgroupname);
 输入参数：
 rgroupname, String 类型，给一组资源自定义的名字
 输出：
 ResourceHandle 对象的数组，指代对应实体。如果没有符合条件的，则返回的数组长度为 0.
 抛出异常：
 NoResource 异常，当查找不到任何资源的时候给出该异常信息
 NoPrivilege 异常，用户对所有查找到的资源都没有权限操作
 调用说明：
 根据参数给定的名字在符号表文件中找到相应的解析条件，查找，并返回满足条件的所有资源，供应用自己选择，或者使用全部资源。
 目前，该接口只有 resolve 接口实现有效。

4 内部接口（模块之间）

4.1 命名系统

所在包名：org.gos.core.naming.client

所在类名：NamingClient

关键数据结构：

GNode 和 GNodeInfo。请参考 3.1 节对 GNode 和 GNodeInfo 的描述。

● 内部接口列表

返回类型	接口名称	简要说明
GNode	register	注册一项实体，形成 GNode
GNode	update	更新 GNode 信息
List	unregister	注销 GNode 信息
GNode	locate	定位获得特定 guid 的 GNode 信息
List	search	根据属性搜索满足条件的 GNode 信息

表 4-1 Naming 内部接口列表

● 功能说明：注册一项实体，形成 GNode

接口描述：GNode register(GNode gnode) throws GosException

输入参数：

Gnode 待注册的 Gnode，有部分信息，通过指定或者注册后产生 guid

输出：

注册完成并补足内容的 GNode 对象

抛出异常：

此 guid 已存在

调用说明：

注册 GNode，一旦注册则此 GNode 可以被全网格查找到。注册时产生相应的 GNode 的 guid，并产生一系列系统属性。注册时如果 GNode 已经提供 guid，则使用此 guid。

● 功能说明：更新 GNode 信息

接口描述：GNode update(GNode gnode) throws GosException

输入参数：

GNode 待修改的 GNode，此对象将覆盖老的 GNode 对象的值。

输出：

修改后的 GNode。

抛出异常：

此 guid 不存在

调用说明：

修改 GNode 的属性。GNode 的属性分为通用属性和用户自定义属性两种。通用属性对各种类型的 GNode 均适用，有相同结构，用户自定义属性允许用户扩展定义新的属性。通用属性中有一些是由命名系统维护，用户不允许修改。

● 功能说明：注销 GNode 信息

接口描述：List unregister(String guid) throws GosException

输入参数：

Guid 待注销的 Gnode 的 guid

输出：

此次注销引起的删除的全部 GNode 的列表。

抛出异常：

此 guid 不存在

调用说明:

注销某 GNode。一旦注销, 则系统中无法再找到此 GNode。前文提到对象型 GNode 和引用型 GNode。如果注销对象型 GNode, 则其对应的各个引用型 GNode 也会同时注销掉, 此时 List 中最后一个元素就是被删除的对象型 GNode。通过此 List, 此 API 的调用者可以在出现异常时针对此 List 做后续处理, 比如重新注册回去。

- 功能说明: 定位特定 guid 的 GNode 信息

接口描述: GNode locate(String guid) throws GosException

输入参数:

Guid 待定位的 Gnode 的 guid

输出:

此 GNode 对象

抛出异常:

此 guid 不存在

调用说明:

根据 id 找到对应资源的 GNode, 其中有元数据。

- 功能说明: 根据属性搜索满足条件的 GNode 信息

接口描述: List search (String searchCondition, Object[] values) throws GosException

输入参数:

searchCondition 以属性作为搜索条件, 搜索条件中用?代表需要替换的变量, 变量的值在 values 中依次给出。

searchCondition 的支持的语法如下:

支持的算子包括: =, >, <

支持的逻辑运算符包括: and, or, not

支持()改变优先级

合法的 searchCondition 的例子 1:

```
type=? and acl=? and version>? and attributes['keyword']=? and
    *
    attributes['serviceCategory']=? and
attributes['attr\"2']=?
```

注意: 例子中 type, acl 都是 GNode 中定义的数据成员。GNode 中有一个可供扩充的 Map 型属性 attributes, 其中以 key-value 对形式存储的可扩展的属性, 对于这种属性的搜索以 attributes['属性名']的形式。

合法的 searchCondition 的例子 2:

```
gn.guid=? or gi.guid=?
```

注意: 由于 GNode 和 GNodeInfo 都有 guid 字段, 因此, 我们用 gn.guid 标识 GNode 的 guid, 而用 gi.guid 标识 GNodeInfo 的 guid。

Values: 与 condition 中的?对应的变量的值, 此对象数组的每个元素依次替代

searchCondition 中的? 形成真正的搜索条件。

输出:

符合条件的 GNode 的列表。如果没有符合条件的, 则返回的列表长度为 0.

抛出异常:

调用说明:

4.2 资源控制器

所在包名: org.gos.core.rc.client

所在类名: RControllerClient

● 内部接口列表

返回类型	接口名称	简要说明
GNode	add	创建资源
void	remove	删除资源
OperateSession	open	申请使用资源
void	close	释放资源
RuntimeHandle	execute	执行对资源的操作
void	clean	销毁运行时实体

表 4-2 RController 内部接口列表

RController 提供如下操作 (RController 有客户端和服务端):

● 功能说明: 创建资源

接口描述: GNode add(OperateContext context, GNode gnode, Object[] resInfo) throws GosException

输入参数:

context 执行此次操作的上下文信息

gnode 用于存储资源元信息的数据结构

resInfo 此资源相关信息, 具体格式由具体的资源确定。

输出:

代表此资源的元信息

抛出异常:

调用说明:

将资源部署到平台, 返回部署好的资源的引用 (如果是外部资源, 则 deploy 得到一个内部访问点的引用)。这个 GNode 还是未注册的, 需要注册到平台以便为系统所感知

● 功能说明: 删除资源

接口描述: void remove(OperateContext context, GNode gnode) throws GosException

输入参数:

context 执行此次操作的上下文信息

gnode 需要操作的 GNode

输出:

无

抛出异常:

权限不够

社区不存在

调用说明:

- 功能说明: 申请使用资源

接口描述: `OperateSession open(OperateContext context, GNode gnode, String option) throws GosException`

输入参数:

context 执行此次操作的上下文信息

gnode 需要操作的资源

option 给出 `open` 时所需的参数, 具体如何取值由具体的资源决定

输出:

创建的 `OperateSession` 对象

抛出异常:

权限不够

资源不存在

调用说明:

申请使用资源。对于不需要 `open` 的 (比如服务), `client` 端就直接屏蔽掉。而像消息就需要 `open` 来建立连接。

- 功能说明: 释放资源

接口描述: `void close(OperateSession session) throws GosException`

输入参数:

session `open` 得到的 `OperateSession` 对象

输出:

无

抛出异常:

权限不够

资源不存在

调用说明:

`Open` 的逆操作, 释放资源。

- 功能说明: 执行对资源的操作

接口描述: `RuntimeHandle Execute(OperateSession session, String operationName, Object[] parameter, Boolean isSync) throws GosException`

输入参数:

session open 得到的 OperateSession 对象
operationName 操作名
parameter 参数列表
isSync 是否同步方式

输出:

返回运行实体的指代物 (是一个 RuntimeHandle 对象, 对于 Web service, 可能代表了一个服务线程), 通过 RuntimeHandle 的 getResult()接口可以取得调用结果。同步时得到 RuntimeHandle 可立刻取得调用结果, 异步调用时由 RuntimeHandle 的 getResult()接口可能导致等待 (wait-by-necessary)。

抛出异常:

权限不够
资源不存在

调用说明:

- 功能说明: 销毁运行时实体。

接口描述: Void clean(RuntimeHandle handle) throws GosException

输入参数:

handle 需要销毁的运行时实体

输出:

无

抛出异常:

调用说明:

销毁此 RuntimeHandle 及其所指代的运行时实体

5 服务描述

5.1 服务说明

以下部分只描述了核心相关的服务。系统服务的接口与其外部接口相同, 这里不再赘述。

针对用户, 资源, 社区和网程的操作分别提供一个服务, 其操作 就是其外部接口部分。另外, Naming 部分也有其服务接口, 但这些接口不对外公开。

由于实现中用户、资源、社区的库的接口中与属性操作相关的由 Naming 系统完成, create 和 delete 等由相应的 RController 完成, 因此在服务的操作列表中没有以上两类接口。

5.1.1 用户服务操作列表

操作名	操作说明
Add	添加用户
Remove	删除用户
update	修改用户信息
View	查看此用户的信息
login	用户登录
logout	用户退出网格

表 5-1 用户服务操作列表

5.1.2 资源服务操作列表

无

5.1.3 社区服务操作列表

接口名	操作说明
Add	添加社区
Remove	删除社区
update	修改社区信息
view	读取此社区相关信息
listGroups	列出指定社区中的全部组
addGroup	添加一个组
removeGroup	删除一个组
updateGroup	修改组信息
viewGroup	查看组信息
addGroupUser	将社区中用户加入到组
removeGroupUser	删除组用户
listGroupUser	列出组用户

表 5-2 社区服务操作列表

5.1.4 Naming 服务操作列表

接口名	操作说明
register	创建资源
update	删除资源
unregister	申请使用资源
locate	执行对资源的操作

表 5-3 Naming 服务操作列表

5.2 WSDL

限于篇幅，这里不再赘述。

5.3 部署描述

5.3.1 异常相关部署

初始化:

```
<requestFlow>
  <handler name="initialize_x2D_grip"
    type="java:gos3.init_x2D">
    <parameter name="path" value="c:\gos3"/>
    <parameter name="file" value="grip.x2d"/>
    <parameter name="persistent" value="true"/>
  </handler>
</requestFlow>
```

释放空间:

```
<responseFlow>
  <handler name="release_x2D_grip"
    type="java:gos3.rel_x2D">
    <parameter name="namespace" value="http://gos3"/>
    <parameter name="section" value="grip"/>
    <parameter name="recursive" value="true">
  </handler>
</responseFlow>
```

6 参考文献

- [1] 《CNGrid_GOS_v3.0_WP2.1_系统软件总体设计_v0.6.2》
- [2] 《CNGrid_GOS_v3.0_WP6.1_批作业子系统总体设计》