## 1、SVN服务概况

SVN服务搭建在云服务器Dibon之上，SVN通信建立在SSH隧道之上。

SVN服务的用户名和密码与Dibon的Linux用户名和密码一致，用户可通过SSH协议登录Dibon完成密码修改。

云服务器Dibon的基本信息如下：

|  |  |
| --- | --- |
| **主机地址** | dibon.cetlab.org |
| **SSH端口** | 13022 |

## 2、用户账户申请

云服务器Dibon的用户名采用CET Lab ID。CET Lab ID的格式为“名.姓”，其中名和姓是可以根据自己的喜好任意确定的，但必须全部由小写英文字母构成。以下是几位用户的CET Lab ID示例：

|  |  |
| --- | --- |
| **姓名** | **CET Lab ID** |
| 严珩铭 | chardle.yim |
| 黄冠 | guan.h |
| 叶欢 | luoluo.ye |
| 周少雄 | shaoxiong.zhou |

使用CET Lab ID的注意事项：

【1】CET Lab ID一经确定，原则上不允许再次更换；

【2】因工作需要，不同用户能够读、写的电子资料有所不同，即不同CET Lab ID的权限有所不同。用户不得将自己的CET Lab ID借给他人使用，也不得将CET Lab ID对应的登录密码外泄。

用户确定了自己的CET Lab ID之后，将其提交Dibon管理员，管理员将以此创建Linux用户并设置初始密码。用户可通过SSH登录Dibon进行密码修改。

## 3、客户端安装和配置

请按照以下步骤操作，完成SVN客户端的安装和配置。

### 3.1、安装TortoiseSVN

去TortoiseSVN的官方网站（http://tortoisesvn.net）下载最新版的TortoiseSVN并安装。

### 3.2、配置Subversion

注：本配置方法仅适用于Windows用户。

进入Subversion配置文件所在的目录（若Windows用户名为Administrator，则目录为“C:\Documents and Settings\Administrator\Application Data\Subversion”），在该目录下，找到文件“config”，用文本编辑器打开，找到内容为“[tunnels]”的一行，在该行下新增一行，写入如下内容：

|  |
| --- |
| dibonssh = "C:\\Program Files\\TortoiseSVN\\bin\\TortoisePlink.exe" -P 13022 -l xxx -pw ??? |

其中：

C:\Program Files\TortoiseSVN是TortoiseSVN的安装目录，请根据实际情况调整；

xxx是登录Dibon的用户名（即CET Lab ID）；

???是登录Dibon的密码。

可见，配置文件中明文记录了登录Dibon的密码，所以SVN客户端一定要配置在私人计算机上，切不可在公共计算机上使用SVN。

## 4、客户端使用方法

SVN的三项基本操作是checkout（检出）、update（更新）和commit（提交）。

本文档将以项目svntest为例进行讲解，该项目是公用的测试仓库，每个用户都可以随意读取和修改svntest项目的内容，以便学习和试验SVN的用法。

### 4.1、Checkout（检出）

项目svntest的检出地址为：“svn+dibonssh://dibon.cetlab.org/data/svn/svntest/”。对于其他项目，请将检出地址中的svntest替换为项目名称。

### 4.2、Update（更新）

更新是指将SVN上的新版本作拷贝更新到本地的工作副本。

可以直接使用右键“SVN Update”，TortoiseSVN 会首先找到版本库的HEAD版本进行更新下最新版本的项目。

若想更新其他版本库的内容，可右键“TortoiseSVN” -->“Update to revision...”，选中“HEAD revision...” 则是检出最新的修改版本，若想取得之前的版本，则可以选中下面的“Revision”，并在右边填写要检出的版本号，若忘记具体版本号，可点击“Show log”进行查看，然后填写版本号点击ok完成所需更新。若填写不存在的版本号则错误提示并检出最新版本库。

更新完成后项目上有红色“!” 号标识的文件或者文件夹就是本地与svn上内容不一致的文件，而有蓝色 “?” 标识的文件或文件夹是指SVN上不存在此文件。或者，也可以先使用 “TortoiseSVN” -->“Check for modifications” 查看哪些文件在本地或是服务器上已经有了改动。

### 4.3、Commit（提交）

将对本地的工作副本的修改发送给版本库，称为提交修改。建议在提交之前先确保自己的工作副本是最新的，以免提交的程序与其他成员提交的程序发生冲突而报错。

若本地的工作副本版本并不是最新的，而右键“SVN Commit”进行提交，则会出现错误提示“You have to update your working copy first”提示更新版本，点击OK -->Update ,则可以检出此文件的各个版本进行对比查看。

若已经确保所要提交的工作副本是最新的，便可以在项目文件夹下点击右键或者所修改的文件下点击右键，选择“SVN Commit”进行提交程序。这两者的区别在于：第一个可以一次提交所有文件的修改，而第二个只是提交所选的文件。点击“SVN Commit”之后，弹出更改日志的对话框.输入你所作的修改，然后点击OK，就会开始向远程服务器提交修改。建议只提交有作修改的文件，而不要整个项目一起提交，既可避免和别人发生冲突也可节省时间。

若想删除文件/文件夹，则在本地删除后，右击“SVN Commit”，选择确定要删除的文件/文件夹，OK确认提交。

若想新增文件/文件夹，空白处右击“SVN Commit”，选择要提交的新增的文件/文件夹，或者可以直接点击该文件/文件夹，右键“TortoiseSVN” -->“Add”，然后右击“SVN Commit” ，OK确认提交。

若在本地工作副本上对文件/文件夹既有删除新增也有修改都可以同时进行，只需右击“SVN Commit”，选择所要对远程SVN上执行的操作即可。

若想移动文件/文件夹，右键“TortoiseSVN” -->“Repo Browser”，选择需要移动的文件/文件夹进行拖拉移动，填写修改日志，点击OK确认提交。

## 5、SVN使用原则

SVN的使用应遵守以下原则：

### 5.1、负责而谨慎地提交自己的代码（先更新后提交）

SVN更新的原则是要随时更新，随时提交。当完成了一个小功能，能够通过编译并且并且自己测试之后，谨慎地提交。 如果提交过程中产生了冲突，则需要同之前的开发人员联系，两个人一起协商解决冲突，解决冲突之后，需要两人一起测试保证解决冲突之后，程序不会影响其他功能。

如果提交过程中产生了更新，则也是需要重新编译并且完成自己的一些必要测试，再进行提交。

### 5.2、保持原子性的提交

每次提交的间歇尽可能地短，以一个小时，两个小时的开发工作为宜。如在更改UI界面的时候，可以每完成一个UI界面的修改或者设计，就提交一次。在开发功能模块的时候，可以每完成一个小细节功能的测试，就提交一次，在修改bug的时候，每修改掉一个bug并且确认修改了这个bug，也就提交一次。我们提倡多提交，也就能多为代码添加上保险。

### 5.3、不要提交自动生成的文件

一些开发工具（如Visual Studio）在使用过程中会产生很多自动文件，如.suo等配置文件、Debug、Release、Obj等编译文件，以及其他的一些自动生成、同编译代码无关的文件，这些文件在提交的时候不应该选中，如果不小心提交了，需要使用Delete命令从仓库中删除。这个可以使用SVN过滤功能，在设置里面设置ignore lists。

### 5.4、不要提交不能通过编译的代码

代码在提交之前，首先要确认自己能够在本地编译。如果在代码中使用了第三方类库，要考虑到项目组成员中有些成员可能没有安装相应的第三方类库或者没有放入GAC（针对.Net Framework）中，项目经理在准备项目工作区域的时候，需要考虑到这样的情况，确保开发小组成员在签出代码之后能够在统一的环境中进行编译。

### 5.5、不要提交自己不明白的代码

代码在提交入SVN之后，你的代码将被项目成员所分享。如果提交了你不明白的代码，你看不懂，别人也看不懂，如果在以后出现了问题将会成为项目质量的隐患。因此在引入任何第三方代码之前，确保你对这个代码有一个很清晰的了解。

### 5.6、提前宣布自己的工作计划

在自己准备开始进行某项功能的修改之前，先给工作小组的成员谈谈自己的修改计划，让大家都能了解你的思想，了解你即将对软件作出的修改，这样能尽可能的减少在开发过程中可能出现的冲突，提高开发效率。同时你也能够在和成员的交流中发现自己之前设计的不足，完善你的设计。

### 5.7、对提交的信息采用明晰的标注

+) 表示增加了功能

\*) 表示对某些功能进行了更改

-) 表示删除了文件，或者对某些功能进行了裁剪、删除、屏蔽

b) 表示修正了某个具体的bug