DCMD Task Cmd手册

**(V0.1)**

*变更记录*

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **状态** | **描述** | **作者** |
| 2013-12-9 | V0.1 | 初稿 | cwinux |

**目录**

[1 简述 4](#_Toc374481507)

[2 Task cmd脚本的执行环境 4](#_Toc374481508)

[3 Task cmd脚本的执行结果文件 5](#_Toc374481509)

[4 【服务】的上线软件包命名及存储 6](#_Toc374481510)

[5 服务的安装位置 7](#_Toc374481511)

[6 Dcmd对【服务】的要求 9](#_Toc374481512)

[7 【服务】的安装、更新 10](#_Toc374481513)

[8 【服务】的启动、关闭与健康性检查 10](#_Toc374481514)

[9 【服务】的配置文件 10](#_Toc374481515)

## 简述

Dcmd的Task通过Task cmd的脚本，实现任务的分布式执行。其虽然可通过用户自己编写的task cmd脚本做任何事情，但其设计的初衷是实现【服务】的自动、可控、可视的发布。

对于【服务】的发布，牵扯到如下的内容：

* 【服务】的上线软件包命名及存储
* 【服务】的安装位置
* Dcmd对【服务】的要求
* 【服务】的安装、更新
* 【服务】的启动、关闭与健康性检查
* 【服务】的配置文件

下面会对这些内容进行详细的说明，在说明之前，会首先说明Task cmd脚本的运行环境，在最后，会通过一个【服务】的install的task cmd脚本实例，说明task cmd的编写。

## Task cmd脚本的执行环境

Task任务的task cmd，是由dcmd的agent在服务器上执行的，而且是按照【服务】池子所指定的用户执行的。

Task cmd的脚本通过dcmd agent执行的时候，dcmd agent会将下面的变量通过环境变量的方式，传递给task cmd的脚本，供task cmd的脚本使用。

具体的变量说明如下：

* DCMD\_SVR\_POOL

服务所属的dcmd的服务池子名

* DCMD\_SVR\_NAME

服务的名字

* DCMD\_SVR\_VERSION

服务的版本号

* DCMD\_SVR\_REPO

服务软件包存储的Repository目录位置，如svn的位置

* DCMD\_SVR\_USER

服务安装、启动的OS的用户名

* DCMD\_SVR\_IP

服务所在服务器的IP地址

* DCMD\_SVR\_UPDATE\_ENV

是否需要更新服务的环境配置。1：是；0：不

* DCMD\_SVR\_UPDATE\_TAG

是否需要重新获取服务软件包。1：是；0：不

* DCMD\_SVR\_ENV\_V

服务的环境配置版本

* DCMD\_SVR\_OUT\_FILE

Task cmd执行结果的输出文件，此输出文件，描述task cmd执行的结果，内容如下：

* + - * process=进度

此为任意的字符串，必须在一行。

* + - * state=success/failure

输出state表示执行状态；success：成功；failure：失败。必须在一行。

* + - * errmsg=错误信息

若任务执行失败，则记录错误信息。错误信息可以多行。

* DCMD\_SVR\_PROCESS

是否输出任务执行的进度。1：是；0：不

* Task cmd的自定义参数变量

Task cmd可以定义0个或任意数量的task cmd参数，并在任务执行的时候指定参数的值。这些task cmd的参数，也是通过环境变量的方式，传递给task cmd脚本的，而且这些参数的命名规则如下：

* + DCMD\_TASK\_xxx

参数名为xxx的参数

* + DCMD\_TASK\_yyy

参数名为yyy的参数

## Task cmd脚本的执行结果文件

Task cmd执行成功与否，是通过task cmd执行输出结果文件指定的。

Task cmd的执行输出结果文件，是通过task cmd执行时的环境变量$DCMD\_SVR\_OUT\_FILE来指定的。

$DCMD\_SVR\_OUT\_FILE环境变量指定文件的内容如下：

* + - * process=进度

此为任意的字符串，必须在一行。

* + - * state=success/failure

输出state表示执行状态；success：成功；failure：失败。必须在一行。

* + - * errmsg=错误信息

若任务执行失败，则记录错误信息。错误信息可以多行。

## 【服务】的上线软件包命名及存储

要通过dcmd的任务实现【服务】的发布，必须有地方存储待发布的服务软件包，这就牵扯上线服务软件包的存储问题。

由于Dcmd任务的Task cmd可以有用户自由编写，因此，dcmd对软件包的存储的获取方式没有任何限制，可以通过svn、ftp、http、git、甚至共享存储的方式获取。同样对服务软件包的命名也没有限制。

但，基于dcmd的设计，无论采用哪种方式获取软件包，dcmd建议采用如下的方式进行服务软件包的命名以及组织服务的软件包的存储

1. 服务软件包的命名规范：服务名\_版本号.tar.gz

在dcmd中，服务的名字是唯一的，而且在dcmd创建task的时候，需要指定task所操作的服务名及服务的版本号。

因此，若以【服务名\_版本号.tar.gz】的方式命名服务的软件包，那么task的task cmd脚本，就可以明确的知道要上线的服务的软件包的名字(dcmd会将服务的名字、服务的版本号，通过环境变量的方式传递给task cmd脚本)，而无需额外的指定。

通过【tar.gz】的后缀，明确了服务的软件包在上线的时候，如何进行解包。

1. 服务软件包的存储

Dcmd的服务池子，可以设置服务软件包存储的Repository，Repository指定了软件包存储的根目录，如svn中代码存储的位置。

在执行task的task cmd脚本时，会将dcmd会将此位置传递给task cmd的脚本，因此，dcmd的task cmd可以通过Repository及软件包的名字，计算出软件包存储的网络位置。

为了简单的目的，dcmd建议采用如下的结构来存储服务的不同版本的软件包：

--软件包存储根目录

|---package

| |---服务名字1

| | |--服务名子1\_版本号1.tar.gz

| | | ……

| | |--服务名子1\_版本号k.tar.gz

| |---服务名字n

| |--服务名字n\_版本号1.tar.gz

| | ……

| |--服务名字n\_版本号k.tar.gz

|---conf

|---服务名字1

|---具体服务池子1

| |---env1 //服务池子1的env1的配置文件目录

| |---env2 //服务池子1的env2的配置文件目录

|---具体服务池子2

通过以上的存储目录结构，dcmd task的task cmd，根据传入的Repository、服务名、服务版本号，就可以自动计算出服务软件包的存储位置。

同时，也可以计算出服务配置文件的目录。

## 服务的安装位置

Dcmd自身对于服务的服务器安装位置没有任何特殊要求。但为了实现所有的服务都采用一个task cmd脚本实现发布，建议服务在线上服务器的安装位置及目录结构如下：

---Base Path

|---服务名1

| |--package：服务软件包的存储目录，存储最多N个最新的服务软件包。

| |--running：服务运行位置目录，为服务软件包解压形成的根目录。

| | |--bin：服务运行的bin目录，内部有start.sh、stop.sh、check.sh

| | |--conf:配置文件目录，内部有服务所需的所有配置文件

| | |--log：运行服务的日志目录

| | |--服务的其他各种目录

| |--version：当前running目录下运行服务的版本号记录文件

| |--back：上线时running代码的备份目录。

|---服务名2

|--package：服务软件包的存储目录，存储最多N个最新的服务软件包。

|--running：服务运行位置目录，为服务软件包解压形成的根目录。

| |--bin：服务运行的bin目录，内部有start.sh、stop.sh、check.sh、

| | install.sh

| |--conf:配置文件目录，内部有服务所需的所有配置文件

| |--服务的其他各种目录

|--version：当前running目录下运行服务的版本号记录文件

|--back：上线时running代码的备份目录。

通过这样组织的目录结构，可以通过一个脚本实现所有符合此要求的服务的上线。当前，各个使用者可以根据自己的规范，定义自己服务的线上结构，只要编写自己的task cmd脚本就可以了。

在上线的时候，task cmd的脚本处理流程如下：

1. 版本检查

检查version文件内的版本号，若与当前的版本不一致或DCMD\_SVR\_UPDATE\_TAG=1,则继续，否则不替换线上版本，安装成功。

1. 下载要上线版本并解压到package的目录下

下载需要上线的版本到package目录下，并解压的【具体服务名\_版本】的子目录下，若失败，则上线失败，否则继续。

1. 从线上下载配置文件

配置文件的目录，在【Repository/conf/服务名/具体服务池子/env版本】的目录下，整个下载下来就可以了。放到解压package的conf目录下。

1. 停止线上服务

执行running/bin/stop.sh，若失败则上线失败，否则继续。

1. 删除version文件

将【具体服务名】目录下的version文件删除，若失败则上线失败；否则继续。

1. 将running下的代码，move到back目录

为了保留running下的数据，会将running下的所有文件，move到back的目录下,以便install.sh程序使用。若失败则上线失败，否则继续。

1. 将解压的代码，从package目录下，move到running目录下

若失败则上线失败，否则继续。

1. 执行running/bin目录下的install.sh

cd到running/bin目录，执行install.sh，传入两个参数：

第一个：running的目录

第二个：back的目录

若失败则上线失败，否则继续。

1. 执行running/bin目录下的start.sh

在running/bin目录下，执行start.sh，只传入running的目录。

若失败则上线失败，否则继续

1. 删除back目录下的备份代码

删除back下的代码，若失败则上线失败，否则继续。

1. 创建version文件，并写入版本号

创建version文件并写入当前running的版本号，若失败则上线失败，否则继续。

1. 上线成功

## Dcmd对【服务】的要求

同样，Dcmd自身对于服务没有任何特殊要求，但为了实现由一套dcmd task-cmd脚本实现对所有【服务】的管理，建议【服务】遵循如下的约束：

1. 【服务名\_版本号.tar.gz】软件包加压后的目录结构

…/解压根目录

|--bin：服务运行的bin目录，内部有start.sh、stop.sh、check.sh

| install.sh

|--conf:配置文件目录，内部有服务所需的所有配置文件

|--服务的其他各种目录

也就是说，在解压形成的根目录中，需要有bin目录及conf的目录。

Bin目录下是用来存放【服务】的启动、停止、检查及安装脚本的。

Conf目录是用来存放【服务】的配置文件的。

1. 对于bin下的脚本，说明如下：

所有的task cmd的环境变量都会传递给以下的脚本，除此之外，还会有额外的参数通过脚本执行变量传入。

* + - * 1. start.sh： 服务的启动脚本

只传入一个参数：running的目录

* + - * 1. stop.sh：服务的停止脚本

只传入一个参数：running的目录

* + - * 1. check.sh：服务的check脚本

只传入一个参数：running的目录

* + - * 1. install.sh:

传入两个参数：

第一个：running的目录

第二个：back的目录

## 【服务】的安装、更新

服务的安装、与更新，通过install的脚本来完成的。

Install.sh脚本由各个服务自己编写，其不但可以看到task cmd的所有【DCMD\_】开头的环境变量，而且还会给install.sh命令传入两个参数：

第一个：服务的running目录

第二个：服务的back目录，若为空，则表示是首次安装

对于大部分服务来说，install.sh可以什么都不做，只是对于那些有运行数据文件的服务，牵扯数据文件的拷贝等相关操作。

具体install.sh的编写，有各自的服务根据自己的需要实现。

## 【服务】的启动、关闭与健康性检查

服务的启动、关闭与监控性检查，分别有bin目录下的start.sh、stop.sh、check.sh。这些脚本由各个服务自己编写，其不但可以看到task cmd的所有【DCMD\_】开头的环境变量，而且还会传入一个参数：running的目录。

具体这些脚本的编写，有各自的服务根据自己的需要实现。

## 【服务】的配置文件

由于服务运行在不同的服务池子，其配置文件可能不同；而且不同版本的配置文件也可能不一样。

为了服务不同版本带来的配置文件的问题，解决此问题，每个服务池子都可以设置配置文件的版本，此版本有DCMD\_SVR\_ENV\_V的环境变量指定。

为了解决不同服务池子配置不一样的问题，在Repository中，服务的配置文件是按照服务池子的方式存储的。

一个服务池子的指定配置版本的目录为：

*Repository/conf/服务名/具体服务池子/env版本*

具体如下：

|---conf

|---服务名字1

|---具体服务池子1

| |---env1 //服务池子1的env1的配置文件目录

| |---env2 //服务池子1的env2的配置文件目录

|---具体服务池子2