管理组件升级（更新802版本的改动点）

1. 升级机制
2. 升级操作步骤：获取节点信息----->分发软件包----->升级前检查----->升级----->回退（可选）----->提交工程，每个步骤执行的操作在各个组件升级包的ModuleCfg.xml中设定，升级工具通过ssh链接基础架构虚拟机（需要保证升级工具执行机可以连接基础架构虚拟机的22端口），远程执行相应的命令进行升级操作。
3. 步骤及描述

获取节点信息：获取节点版本信息，如果STATE文件不存在，则创建和初始化各个组件的升级状态记录文件STATE\_versionNum.xml；如果STATE文件存在，就加载当前STATE文件。

分发软件包：把组件升级包上传到组件升级目录/opt/组件名/patch下（Tools的为/opt/ToolsUpgrade/patch），执行ModuleCfg.xml中的命令解压升级包，更新STATE文件中步骤记录。

升级前预检查：升级前深层次对升级包中所有压缩包进行解压，并对当前环境是否满足升级条件的常规检查，一般包含：升级包的完整性、类型检查，磁盘空间检查，版本检查不同组件间有细微差异，具体查看ModuleCfg.xml中子任务描述。

升级：组件实际的升级操作，一般包括：升级前服务状态检查、磁盘空间检查、停止服务、备份业务数据、备份数据库、升级数据库、升级软件、更新rpm包、刷新配置文件，启动业务、升级后状态检查、版本号刷新不同组件间有细微差异，具体查看ModuleCfg.xml中子任务描述。

回退：当升级出现失败时，需要进行的回退操作，包括数据的回退、软件回退、版本号回退-----当前不支持此操作，回退场景需要使用快照恢复。

提交：提交升级工程，会清理组件升级目录下载此次升级任务中产生的文件和目录，包括升级包、备份文件。

1. 升级目录主要文件和结构说明
2. 除去Tools之外，各个组件在各组的安装目录下都会有一个patch目录，里面会有lib和bin目录，是在安装组件的时候就放好的预埋脚本，主要是提供给获取节点信息的时候的命令调用，可以创建和初始化STATE文件由ModuleCfg.xml中指定的命令生成。从802版本开始，所有组件的STATE文件由ModuleCfg.xml中的getnodeinfo命令生成，不依赖预埋脚本。
3. STATE文件：
4. 获取节点信息时在每个组件的升级目录下生成的文件，一般命名为STATE\_versionNum.xml
5. 用于记录组件升级源版本、目标版本、日志路径、升级的每一步步骤执行时间和结果
6. 每一次刷新UpdateTool升级工具页面都会触发获取节点的信息操作，重新检测加载STATE文件，如果STATE文件被删除，会重新生成
7. 当出现UpdateTool升级工具显示的信息不对时，需要检查STATE文件中的信息是否正确完整，如果不正确，可以删除此文件后重新刷新升级工具触发获取节点信息操作，重新生成STATE文件
8. 802版本新增机制：使用UpdateTool升级时（线下升级方式，SDK不涉及），如果是使用了DB的组件（LiteAS、ITA、HDC），LiteAS会在升级工具中判断当前节点是否为IP值更小的节点，是的话会在getnodeinfo命令中创建STATE文件的命令中添加参数isPrimary=“Y”，ITA和HDC组件会在升级前检查步骤会判断当前升级节点是不是主备节点中IP值更小的节点，是的话会向STATE文件的state模块写入一个参数isPrimary=“Y”，在升级步骤中的备份数据库以及升级数据库的操作中只有存在此标记且不是使用SDK升级方式的节点才会执行。
9. ModuleCfg.xml文件
10. path模块：记录可以使用此升级包的源版本，用于判断当前软件包是否可用
11. remotePkgPath模块：设定服务器上组件的升级路径，和需要创建的STATE文件名和路径
12. Workflow模块：指定整个升级工作要做的全部步骤，每个task指定一个步骤，设定了每个步骤的执行命令，其中各个step为该步骤下的子任务，存在超时时间，按顺序执行以8.0.2版本的ITA的ModuleCfg.xml为例
13. 升级目录主要结构说明：（ITA为例）

/opt/ITA/patch：升级目录主目录

--bin：预埋升级脚本入口（在802版本已废弃）

--upgradedecmd.py：预埋升级脚本主入口，主要用于获取节点信息的方法调用

--lib：预埋升级脚本需要调用的对象和方法（在802版本已废弃）

--commonobj.py：提供通用方法，比如获取版本号

--configobj.py：提供对ModuleCfg.xml文件的操作，比如读取版本路径列表

--logobj.py：提供日志打印方法

--pathobj.py：设定路径变量

--stateobj.py：提供对STATE文件的操作方法，比如记录每个升级步骤的时间和结果

--xmlop.py：提供对读写xml文件的通用方法

--backup：升级前备份的数据和配置文件，在执行了升级步骤的“备份数据”子任务后生成

--STATE\_verNum.xml：记录整个升级工程的状态，包括升级源版本和目标版本、升级步骤时间和结果、日志路径、节点信息等

--FusionAccess-HDC-Ptch-verNum：组件升级包

--ModuleCfg.xml：包含此升级包可支持的升级路径和各个升级步骤的操作指令

--patchfile：从分发软件包步骤开始，后续的升级过程使用的方法逻辑都在这个目录下

--bin：升级命令入口文件

--upgradecmd.sh：升级指令总入口，调用了同目录下的upgradecmd.py，是ModuleCfg.xml中分发软件包、升级前检查、升级、回退、提交操作中调用的脚本

--upgradecmd.py：升级指令逻辑入口 ，包含每个升级步骤，根据main方法指定的参数判断执行的操作，每个操作的子步骤由一个字典taskFlowList定义，通过taskFlowCtl来控制字典中的子步骤执行流程和超时时间，字典中每个子步骤包括了：子步骤方法名、参数列表、子步骤描述、超时时间、步骤失败错误码

--lib：提供升级过程中需要使用的对象和方法，可以根据upgradecmd.py中的方法调用的对象名在这个目录下快速找到使用的的方法，比如upgradecmd.py中数据备份方法，内部调用的就是backup.py的backupData方法

包含的对象有：

--backup.py：数据和rpm包备份/恢复/更新相关逻辑

--check.py：升级包、磁盘空间、版本号等相关检查逻辑

--cleanup.py：回退和提交工程的清理临时文件逻辑

--commonobj.py：通用方法

--logobj.py：日志打印方法

--myconfigobj.py：读取、更新STATE文件或其他过程xml文件的方法类

--nodeinfo.py：节点类型判断

--service.py：组件服务状态管理，包括检查、启动、停止

--stateobj.py：State文件加载和操作类

--upgradeobj.py：升级操作类、例如解压软件包、备份数据库、升级数据库、备份业务数据、升级软件、执行各版本路径升级脚本等等

--xmlop.py：xml操作公共方法

--binary：升级过程使用到的二进制文件，比如用于连接数据库的gsql客户端

--config：升级过程中需要使用的组件信息

--script：升级过程中会被调用的备份、恢复脚本

--FusionAccess-HDC-Patch：升级所需的软件、升级脚本

--hdc-patch：升级使用的目标版本软件包、升级每一个版本需要执行的脚本，比如从R6 C20SPC100升级到6.5.1，就需要依次执行V100R006C20SPC101、V100R006C20SPC102、V100R006C20SPC103、6.5.RC1、6.5.RC2、6.5.RC3、6.5.0、6.5.0.SPC001、6.5.1中的S开头的升级脚本和待执行的sql文件

--scripts：包含升级到各个版本需要执行的脚本目录，主要是对配置文件、数据库的修改操作；部分组件（ITA、HDC、LiteAS）在这个目录下还有一些对DB数据操作的脚本

1. 升级日志查看方式：
2. 每个组件的升级日志路径

GaussDB：/var/FusionAccess/GaussDB/upgrade/patch.log；GaussDB升级进程主要日志

ITA：/var/FusionAccess/ITA/upgrade/patch.log；/var/FusionAccess/ITA/upgrade；/var/FusionAccess/ITA/Upgrade\_ITA\_DB.log；ITA升级进程主要日志

ITA处理配置文件的日志，包括dbupgrade.log和itaupgrade.log，分别对应FusionAccess-HDC-Patch/hdc-patch/scripts下面各个版本号目录下执行的升级脚本和sql文件打印的日志 ITA处理ITA数据库的日志

HDC：/var/FusionAccess/HDC/patch.log；/var/FusionAccess/HDC/DB\_Upgrade.log；/var/FusionAccess/HDC/upgrade.log；HDC升级进程主要日志

WI/UNS：/var/FusionAccess/WI/patch/patch.log；/var/FusionAccess/WI/patch/upgrade.log；WI/UNS升级进程主要日志

LIC：/var/FusionAccess/License/upgrade/patch.log；/var/FusionAccess/License/upgrade/upgrade.log；License升级进程主要日志 License处理配置文件日志

vAG：/var/FusionAccess/VNCGate/logs/patch.log；vAG升级进程主要日志

Tools：/var/FusionAccess/InstallTools/patch.log（8.0.2版本前）；/var/FusionAccess/ToolsUpgrade/patch.log（8.0.2版本后）；vLB、HA、BackupTool、InstallTool升级主要日志

1. 每个组件的升级进程主要日志会打印整个组件升级的每个步骤，每个步骤会的日志会清晰的打印步骤开始和结束的标志，可以快速地找到升级步骤的日志
2. 每个步骤的日志主要是由各个子步骤日志组成（获取节点信息和分发软件包不知道操作简单，没有子步骤），每个子步骤开始执行的时间都会打印子步骤的超时时间，关键字start to timeout，标志着当前步骤开始执行，并开始计时，超时时间和upgradecmd.py中的taskFlowList中设定

注：从8.0.2版本开始，每个子步骤开始时会打印类似“---------Backup Data----------”的日志，可根据关键字“----------”查看当前子步骤，每个关键方法调用开始和结束，会打印类似“>>>Begin to Backup data ...”、“Backup data success”的日志，问题定位时可以根据相关关键字确认具体的操作

1. 问题定位时优先查看升级进程主要日志，根据日志打印的错误信息，结合升级逻辑文件去查找执行的脚本或命令，如果是执行数据库相关操作，需要查看处理数据库的日志信息，（部分操作执行的命令会在日志打印出来，脚本一般会打印路径），再手动去检查问题所在