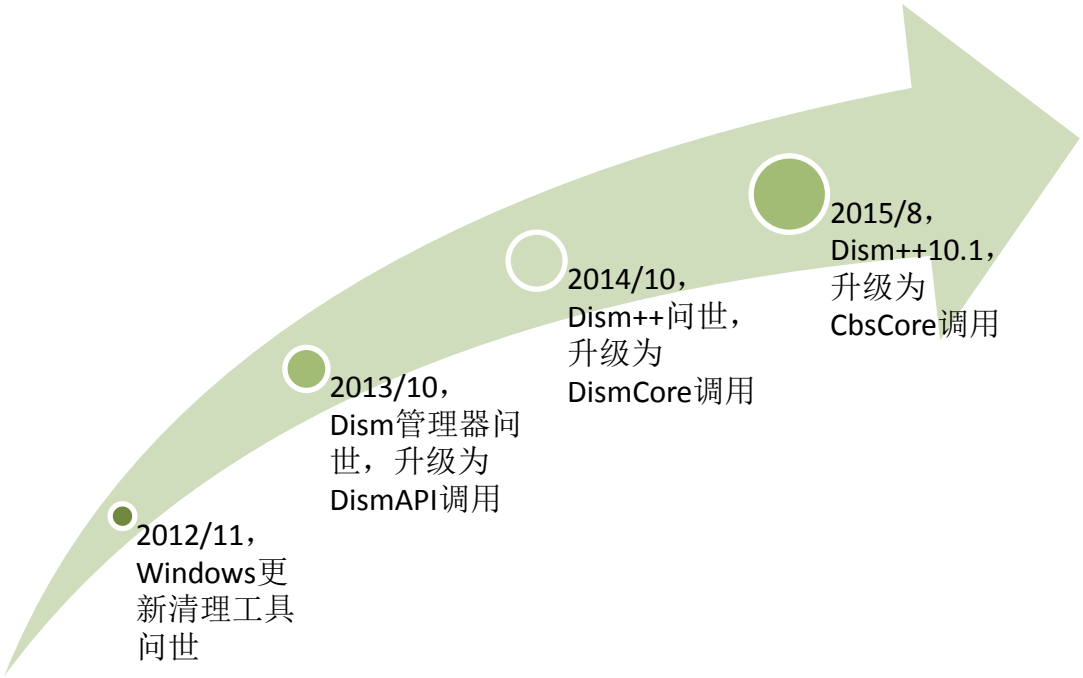
我们是一线战友！

——初雨团队

全球第一款基于 CBS（Component Based Servicing Reference）Dism GUI 实现

1. 关于 Dism++

欢迎到 www.chuyu.me 或者发送邮箱到 mingkuang@live.com 反馈你的意见。
也可以加入软件交流群，大家一起交流。QQ 群：200783396（敲门砖 临安初雨 一夜落红）



Dism++的发展史

1.1.什么是 Dism++？

Dism++是一款系统实用工具，主打清理+制定！

Dism++独有更新清理、过期驱动清理、Appx 清理等功能可以让系统不在慢性增大，而 CompactOS 功能通过哈夫曼算法（一种不牺牲性能，却明显减少体积是压缩算法），让系统盘直接增加 33%以上的可用空间。

Dism++还支持添加删除更新、调整 Windows 功能、设置服务状态、WIM/ESD 文件处理（ESD 解密、ESD 转 ISO、系统备份还原等）……

Dism++可以说是一个 Dism 的 GUI 版，但是 Dism++并不依赖 Dism 相关组件。Dism++直接基于更加底层的 CBS 核心，因此无需任何 Dism 组件即可兼容所有高低版本系统，甚至是 Vista！ 你无需为不同 Dism 版本之间的兼容性而困扰~

PS:为什么我们主打清理 + 制定呢？那是因为 Dism++自身的清理功能很多都是从一些制定管理功能上演变的。比如更新清理，其实就是添加删除更新的友好化。驱动清理则是驱动管理的友好化。换句话说如果不知道那些东西是如何管理的，又怎么知道怎么清理呢？因此对于一般用户，单纯使用清理功能即可。而高级用户则可以考虑使用高级管理功能。

1.2.Dism++运行要求

最小运行平台：Windows Vista、Windows 2008 或者更高

推荐运行平台：Windows 7、Windows 2008 R2 或者更高（虚拟内存 4G 或者更高）

温馨提示：如果运行平台为 Windows Vista 或者 Windows 2008 时，Dism++的某些功能将无法使用，比如 CompactOS。

温馨提示：关闭虚拟内存时，WIM 转 ESD 功能可能会因为内存不足而崩溃。

温馨提示：在没有 WOW64 支持的环境中（比如纯 64 PE），将导致 Dism++无法离线处理 32 位系统。

操作系统具体特性支持情况	
功能最低要求	功能名称
Windows Vista、Windows 2008 Dism++目前没有添加 Vista 更新数据库，因此 Windows Update 无法扫描 Vista/2008	1：更新清理以及添加删除更新 2：打开关闭功能 3：创建保存修改 WIM、ESD 以及 ESD 解密 4：系统热备份以及热还原 5：服务管理 6：BCD 引导修复 7：Windows Update 8：系统优化功能 9：从 ISO 释放/还原系统
Windows 7、Windows 2008 R2 “WIMBoot/Compact”至少需要 Windows 7 Host	10：驱动清理以及驱动管理 11：WimBoot OS 12：Compact OS
Windows 8、Windows 2012 Dism++目前没有添加 Win8 更新数据库，因此 Windows Update 无法扫描 Win8/2012	13：删除功能 14：Appx 管理以及过期 Appx 清理 15：健康扫描以及恢复
Windows 8.1、Windows 2012 R2	16：完整的更新清理支持（即 ResetBase）
Windows 10、Windows 2016 至少需要 Windows 8 Host 才能正常处理 Windows 10	17：可选功能

1.3. Dism++文件列表

温馨提示：Dism++在空间回收中已经内置了对自身的清理规则，如果你想不了解具体细节，请直接使用 空间回收 —— Dism++非必需文件

文件名称	功能说明
Dism++x64.exe	64 位系统的 UI，在 64 位系统中，启动此程序呈现 UI。32 位系统用户可以考虑删除此文件。
Dism++x86.exe	32 位系统的 UI，如果你在 64 位系统中启动此程序，则自动转向到 Dism++x64.exe。64 位系统用户可以考虑删除此文件。
Config\amd64\bcdboot.exe	提供引导修复功能，原版系统自带此文件，删除没有影响。原版系统用户以及 32 位系统用户可以删除此文件。
Config\x86\bcdboot.exe	提供引导修复功能，原版系统自带此文件，删除没有影响。原版系统用户以及 64 位系统用户可以删除此文件。
Config\amd64\CBSHost.dll	Dism++API 支持模块，删除后 64 位系统将无法使用 Dism++。32 位用户可以删除此文件。
Config\x86\CBSHost.dll	Dism++API 支持模块，删除后 32 位系统将无法使用 Dism++，64 位系统无法脱机处理 32 位系统。不需要脱机处理 32 位系统的 64 位用户可以考虑删除。
Config\amd64\DuiLib.dll	Duilib 支持库，为 Dism++x64.exe 提供 UI 支持。32 位用户可以删除此文件。
Config\x86\DuiLib.dll	Duilib 支持库，为 Dism++x86.exe 提供 UI 支持。64 位用户可以删除此文件。
Config\amd64\wimgapi.dll	WIM 文件操作支持模块，Win10 用户、32 位用户或者不需要任何 WIM 相关功能的用户，那么可以考虑删除。
Config\x86\wimgapi.dll	WIM 文件操作支持模块，Win10 用户、64 位用户或者不需要任何 WIM 相关功能的用户，那么可以考虑删除。
Config\amd64\wofadk.sys Config\x86\wofadk.sys	提供 Compact 功能相关支持，不需要脱机处理的 Win10 用户可以考虑删除此文件。强烈建议不要删除这些文件。
Config\Plugins	Dism++插件支持，不需要插件的用户可以删除此文件。
Config\Languages	Dism++的语言文件，以中国为例，只保留 zh-Hans.xml 即可。
Config\Data.xml	此文件保存了清理规则，ESD 解密 KEY 等。此文件不允许删除。
Config\Update.xml	Dism++更新元数据，用于判断是否存在新版本，以及防止用户新版本降级为老版本。推荐保留，删除后将重新下载。
Config\winapp1.ini	此文件保存了 CCleaner 的清理规则，删除此文件后 Dism++将无法使用 CCleaner 的规则。可以考虑删除。
Config\default.ui.zip	此文件保存了 Dism++的 UI。此文件不允许删除。

文件名称	功能说明
Config\wsusscn2.cab	Windows Update 数据库文件，用于扫描更新。可以考虑删除。
Config\include\Dism++.h	C Script 脚本支持头文件，缺少此文件后 C Script 脚本将无法使用。
Config\amd64\CScript.dll	64 位 C Script 脚本解析引擎，缺少此文件后 C Script 脚本将无法使用，32 位用户可以考虑删除此文件。
Config\x86\CScript.dl	32 位 C Script 脚本解析引擎，缺少此文件后 C Script 脚本将无法使用，64 位用户可以考虑删除此文件。
Config\amd64\NCleaner.dll	64 位 NCleaner 清理引擎，缺少此文件后某些高级清理功能将无法使用，32 位用户可以考虑删除此文件。
Config\x86\NCleaner.dl	32 位 NCleaner 清理引擎，缺少此文件后某些高级清理功能将无法使用，64 位用户可以考虑删除此文件。

如果你发现某些文件不在此列表中，可能是某些老版本 Dism++遗留的文件或者是 Dism++的临时文件，直接删除即可，不影响程序功能。

1.4. Dism++组件移植情况

由于 Dism++不依赖微软 Dism，因此所有功能必须自己移植完成。以下是 Dism++的移植情况：

操作系统	组件名称	功能说明	移植状态
NT 6.1 新增	CbsProvider.dll	提供添加删除更新支持	移植完成，已应用于更新管理、Windows Update、可选功能、以及更新清理
	DmiProvider.dll	提供驱动管理支持	移植完成，已应用于驱动管理、以及过期驱动清理
	IntlProvider.dll	国际化命令支持，调整语言等	移植完成，已应用于区域设置
	MsiProvider.dll	Msi/msp 信息获取	移植完成，已应用于 Windows Update 的 Office 更新扫描
	SmiProvider.dll	Windows 离线服务支持	None
	TransmogProvider.dll	比如家庭版升级旗舰版等	移植完成，但未应用
	UnattendProvider.dll	无人应答支持	None
	WimProvider.dll	挂载 wim 文件等	移植完成，已应用于添加映像、挂载映像、Imagex

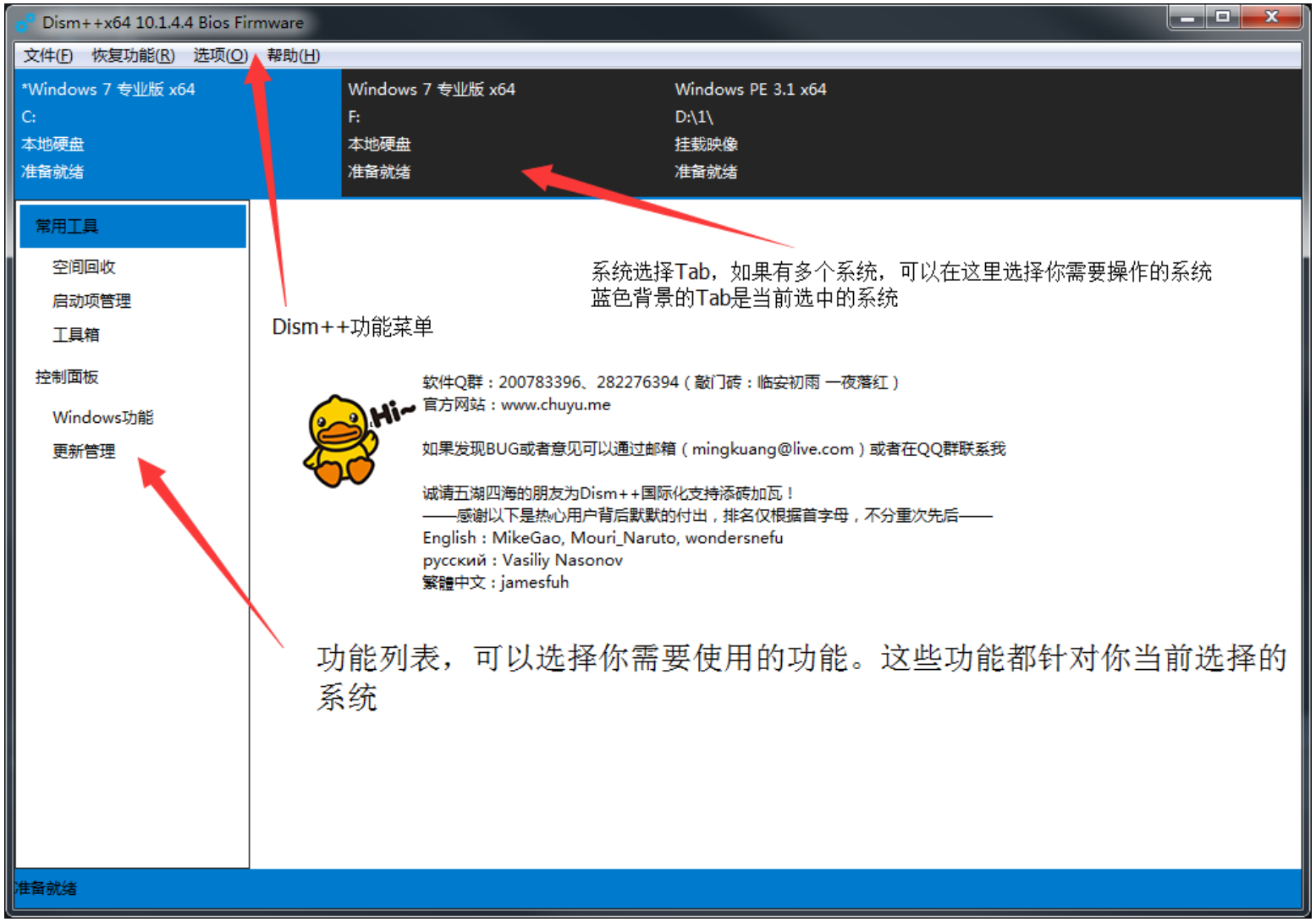
	PEProvider.dll	调整 PE 相关设置	移植完成，PE 管理
NT 6.2 新增	AppxProvider.dll	添加删除 Appx	移植完成，已应用于过期 Appx 清理
	AssocProvider.dll	文件关联支持	移植完成，已应用于文件关联
	IBSProvider.dll	Windows 预部署	移植完成，已应用于预部署
	VhdProvider.dll	用于挂载 VHD/VHDX	None
NT 10.0 新增	FfuProvider.dll	用于释放 FFU 映像	None
	OfflineSetupProvider.dll	离线安装支持程序	None
	ProvProvider.dll	提供 ppkg 映像释放支持。	None

2. 快速入门

程序的核心功能是清理更新，所以你可能需要在安装更新后才能感受到程序的价值。

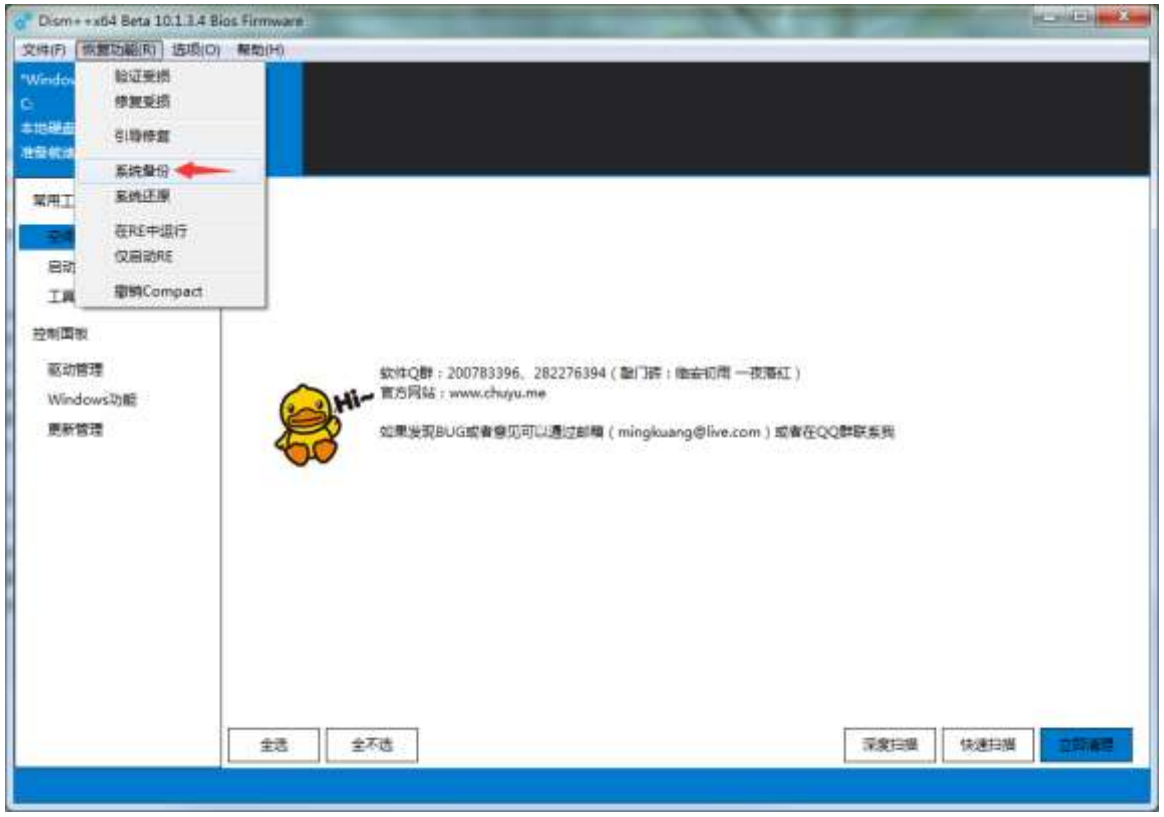
2.1. UI 说明

具体功能布局请参考下图。


Dism++x64 10.1.4.4 Bios Firmware
文件(F) 恢复功能(R) 选项(O) 帮助(H)
*Windows 7 专业版 x64
C:
本地硬盘
准备就绪
Windows 7 专业版 x64
F:
本地硬盘
准备就绪
Windows PE 3.1 x64
D:\1\
挂载映像
准备就绪
常用工具
空间回收
启动项管理
工具箱
控制面板
Windows功能
更新管理
Dism++功能菜单
系统选择Tab，如果有多个系统，可以在这里选择你需要操作的系统
蓝色背景的Tab是当前选中的系统
Hi~
软件Q群：200783396、282276394（敲门砖：临安初雨 一夜落红）
官方网站：www.chuyu.me
如果发现BUG或者意见可以通过邮箱（mingkuang@live.com）或者在QQ群联系我
诚请五湖四海的朋友为Dism++国际化支持添砖加瓦！
——感谢以下是热心用户背后默默的付出，排名仅根据首字母，不分重次先后——
English：MikeGao, Mouri_Naruto, wondersnefu
русский：Vasiliy Nasonov
繁體中文：jamesfuh
功能列表，可以选择你需要使用的功能。这些功能都针对你当前选择的系统
准备就绪

2.2.有备无患，将当前系统备份为 WIM

俗话说有“备”无患，现在 Dism++ 能够直接将当前系统备份为 WIM、ESD，无需进入 PE。使用方法很简单，选中当前系统，点击恢复功能 —— 系统备份即可。



温馨提示：系统热备份需要系统的 VSS 服务支持，当你关闭了该服务或者安装了硬盘过滤驱动导致 VSS 服务使用时，此功能也将无法使用。这时建议你使用 在 RE 中运行或者启动 PE 进行离线备份。

最后点击浏览，设置保存路径即可。如果文件已经存在，那么自动增量到现有文件。

名称	功能解释
卷影复制	使用微软 VSS 服务进行复制，这个可以模式下可以复制被占用的文件。保存当前 Host 系统时无法进行更改。
可启动	即将 WIM 文件设置为可启动，以便于制作 PE。如果你仅仅是为了备份系统，那么此选项对你无用，可以直接无视。

温馨提示：保存为 极限压缩 ESD CPU 的计算量非常巨大，会导致备份时间大大拉长。另外备份过程中推荐关闭杀毒软件。

温馨提示：设置 ——详细设置 ——排除列表中还能设置 WIM 保存时排除名单，加快备份速度。具体请参阅配置豁免列表。

2.3. 系统还原

当你遇到问题时，可以使用 2.1 创建的备份，还原到备份时的状态。此外 Dism++还支持添加自身到 BCD 引导中（具体参考选项——详细设置——添加引导菜单功能），计算机无法启动时可以通过此功能启动 Dism++。

2.3.1. 当系统还能启动时

如果你的系统还能启动，那么可以使用 Dism++的热还原功能，方法很简单，启动 Dism++后点击 恢复功能——系统还原即可。然后选择你当时备份的 WIM/ESD/SWM/ISO 文件。

温馨提示：此功能需要 RE 环境完好。

当然热还原会产生 Windows.old 文件夹，如果你不喜欢这种方式，你可以点击 恢复功能—— 在 RE 中运行。然后重启，你在 RE 中在点击恢复功能——系统还原即可。

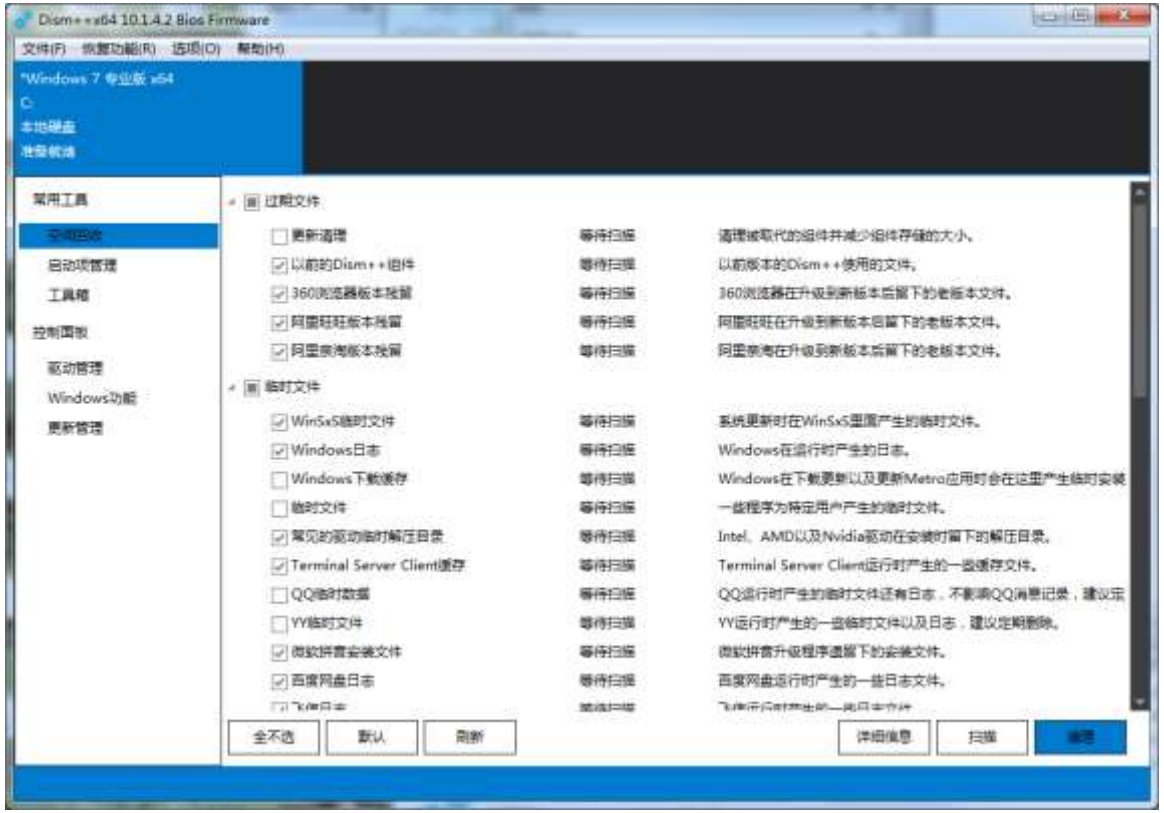
你还能根据你的情况，勾选格式化分区、以及添加引导。

2.3.2. 系统已经无法启动

Dism++有个添加引导菜单功能（选项——详细设置 中），如果你事先添加了引导菜单，那么系统彻底挂了也能通过 Dism++引导菜单重新恢复系统。如果你什么也没做，那么你只能制作 PE 启动盘，然后在 PE 中启动 Dism++，点击 恢复功能——系统还原。

2.4. 使用空间回收垃圾清理

建议你不要过于频繁的清理垃圾，尤其是 SSD！一般建议一个月清理一次即可。



程序启动后，点击空间回收，即可看到此界面。然后选中你需要清理的项目，在点击扫描（预计可以释放的空间）或者直接点击清理（不预估大小，立即删除）。

温馨提示：风险项目使用橙色标识，另外某些项目存在某些副作用，选中后程序将弹出警告框，请务必仔细确认！

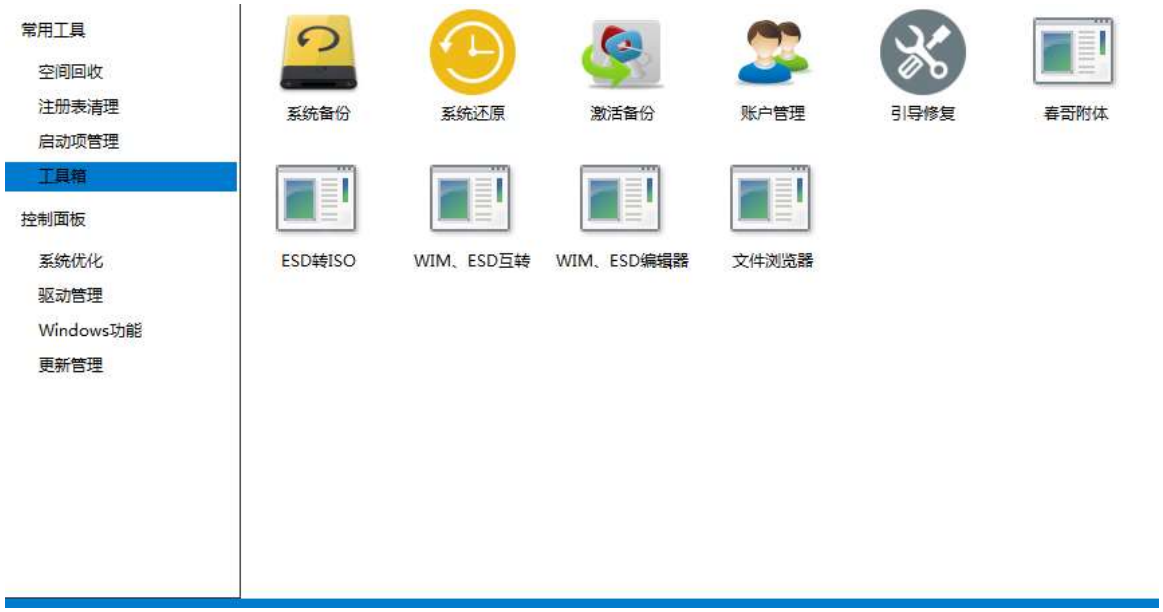
3. Dism++隐藏设置

此处将公开 Dism++未在 UI 中呈现的设置，你可以修改 Config\Config.ini，让这些设置生效。

启用方式	功能
[Dism++] NotLoadWofadk=1	添加后 Dism++将不会在启动时加载 Wof 驱动，默认情况下 Dism++会在 Win8 以上系统自动加载 Wof 驱动。如果你的环境比较特殊，可以开启此选项。开启后，WIMBoot 以及 Compact 相关功能可能会受限。
[WUA] DisableInstallUpdate=1	添加后 Dism++将跳过 Windows Update 中的更新安装过程，转变为仅下载更新，如果单纯需要收集更新文件的用户可以启用此选项。
注册表添加值 HKEY_LOCAL_MACHINE\SOFTWARE\Dism++\Hide	隐藏某些挂载映像，某些 PE 可能存在内置挂载点。启动 Dism++时可能会让用户比较郁闷。你可以在你的 PE 中设置此键。 [HKEY_LOCAL_MACHINE\SOFTWARE\Dism++\Hide] "D:\\7"=dword:00000000 导入后 Dism++将默认不显示 D:\7 这个挂载映像

4. 工具箱

用于展示 Dism++内置小工具，主要用于改进 Dism++的体验或者实现 Dism++本身不支持的功能。



4.1. 系统备份

此功能是等同于 2.2 有备无患。

4.2. 系统还原

此功能是等同于 2.3 系统还原。

4.3.激活备份

用于备份 Windows 以及 Office 的激活信息。备份时无需填写 Key，Dism++会备份 Key 信息，还原时将自动还原 Key 信息。

4.4.账号管理

用于启用、禁用 Windows 账号，还支持直接删除登录密码哦！

温馨提示：暂不支持 Windows 域账号以及微软在线账号密码删除。

4.5.引导修复

此功能是等同于 4.2.3 引导修复。

4.6.春哥附体

可以让其他程序无视权限运行。另外此功能还支持命令行：
Dism++x86.exe /All /Plugin:{4d420a2e-ea11-450a-b8a0-ab8ca7772043} /GodMode cmd
Dism++x86.exe /All /Plugin:{4d420a2e-ea11-450a-b8a0-ab8ca7772043} /GodMode regedit.exe
Dism++x86.exe /All /Plugin:{4d420a2e-ea11-450a-b8a0-ab8ca7772043} /GodMode “\”C:\Windows\notepad.exe\” \”C:\123.txt\””

4.7. ESD 转 ISO

将微软官方的 ESD 转换为 ISO 映像。

温馨提示：非微软发 ESD 不支持转换，将直接返回 格式错误 。

4.8. WIM、ESD 转换

在 WIM、SWM、ESD 格式之间自由转换。

温馨提示：禁用页面文件时，WIM/SWM 转 ESD 可能导致 Dism++因为内存不足而崩溃。

4.9. WIM、ESD 编辑器

用于编辑 WIM、SWM、ESD 的映像信息。

4.10. 文件浏览器

可以无视权限浏览文件，并随意删除。

5. 菜单功能说明

此处主要用于介绍菜单中的一些功能。

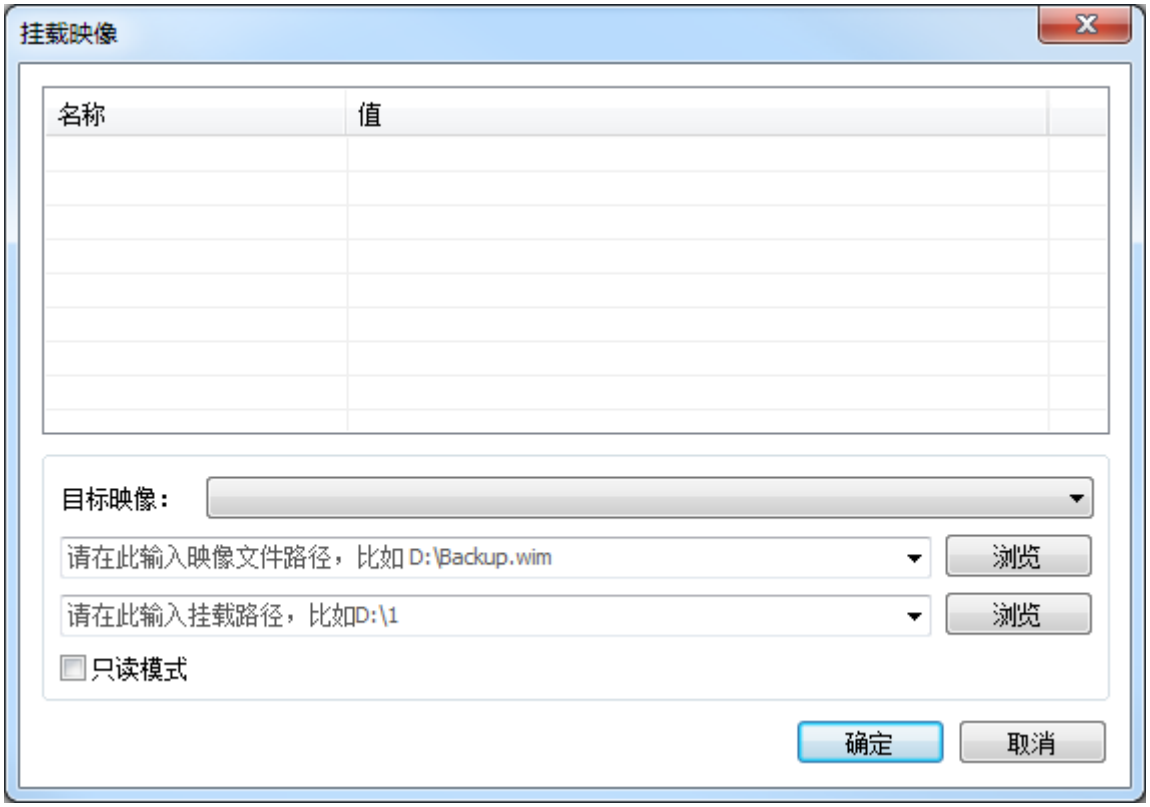
5.1. 文件

提供对一些文件的操作。

5.1.1. 挂载映像

该功能可以将 WIM/SWM 挂载到一个空文件夹中，等待修改完毕后，可以使用保存映像保存更改。

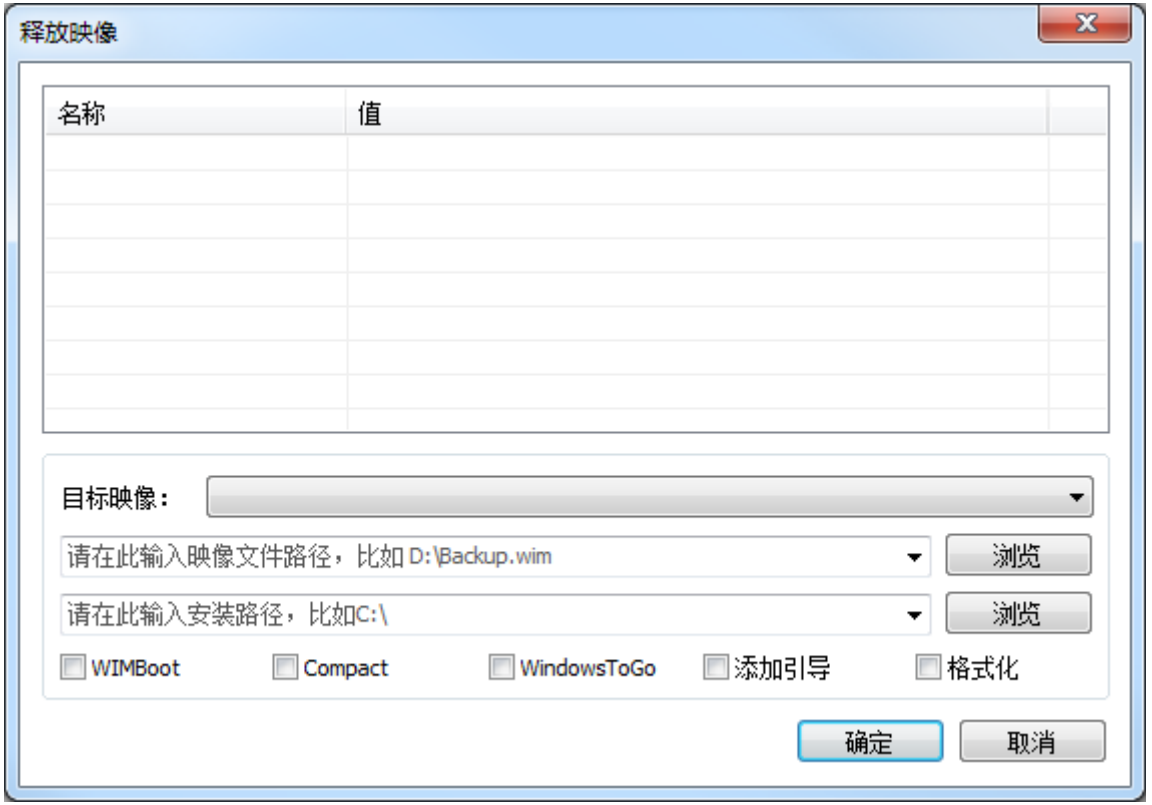
温馨提示：ESD 映像无法挂载。



名称	功能说明
只读模式	此模式挂载后不允许提交保存操作，可以挂载只读映像。

5.1.2. 释放映像

此功能可以将 wim/esd/swm 中的某个映像释放到某个目录。



温馨提示：Dism++支持 WIM、SWM、ESD 以及 ISO 释放，但是 ISO 释放需要 Host 系统为 Windows 8 或者更高。

名称	功能说明
WIMBoot	已经被 Compact 代替（不推荐使用），可创建指针文件，然后从 WIM 启动系统。（注意：极限压缩以及仅存储的 WIM 文件无法 WimBoot）
Compact	释放映像并启用 CompactOS 大幅度缩小系统体积。此选项不可与 WIMBoot 同时使用。
WindowsToGo	将系统释放成便携式系统，方便移动（此选项仅 Windows 8 以上系统可用，其他版本为灰色不可用）。
添加引导	释放后将添加该系统到 BCD 中，并且重建 MBR、PBR。如果你是系统还原，一般无需勾选此选项。（仅为目标为分区时可用）
格式化分区	释放前，先格式化硬盘。推荐使用，这样有助于减少垃圾，但是会丢失该分区所有数据。（仅为目标为分区时可用）

5.1.3. 添加路径

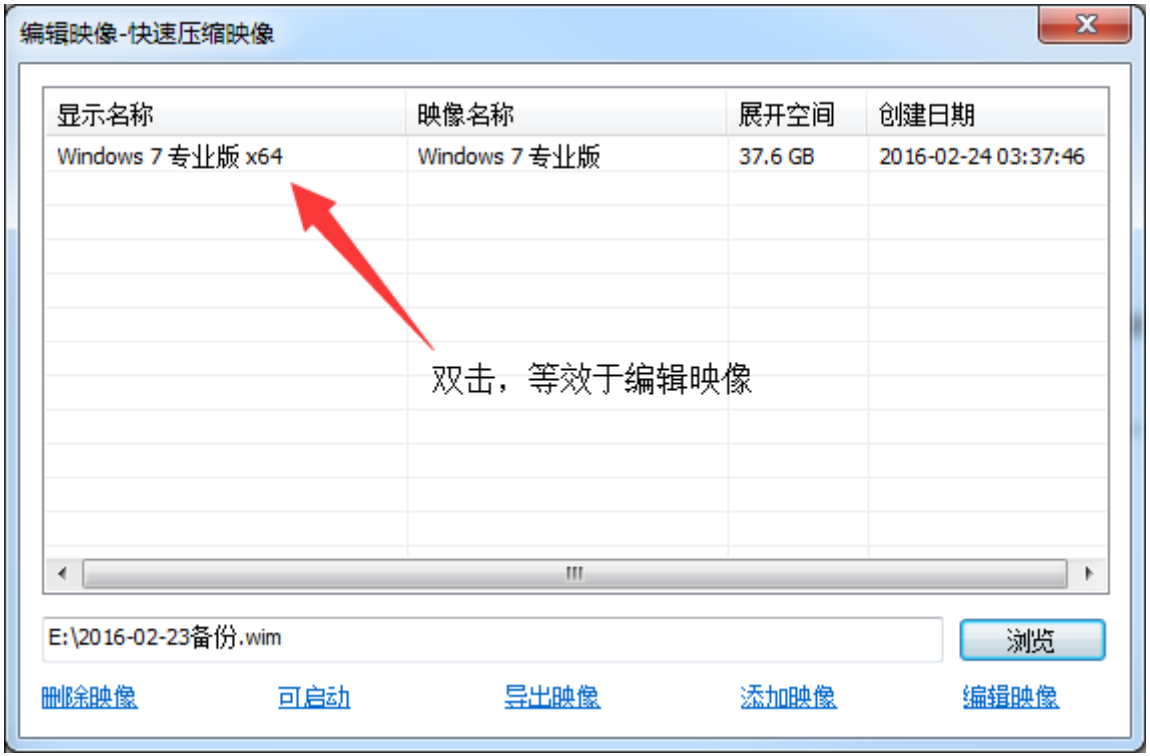
如果 Dism++的系统列表中没有显示你的系统，你可以使用该功能手动将系统添加至列表。

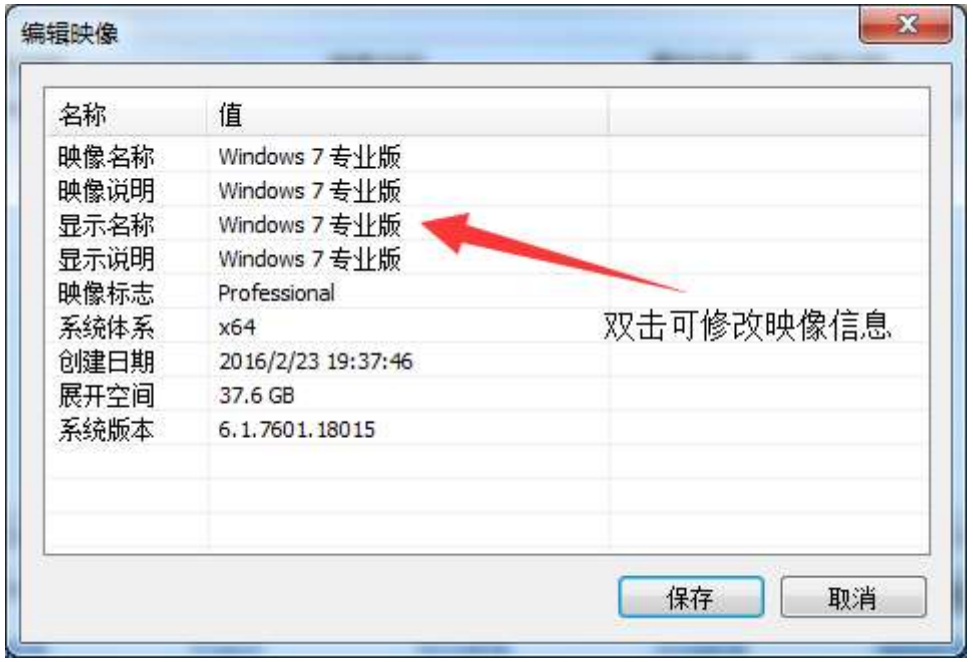
5.1.4. 打开映像文件

此功能用于查看编辑 WIM/SWM/ESD 文件的信息。

按钮名称	功能
浏览	用于浏览 WIM、SWM 以及 ESD 文件。
删除映像	删除映像文件中选中的 Index。
导出映像	将选中的映像导出至另一个映像文件，此操作可以用来合并映像并消除冗余。
添加映像	将一个路径添加至当前打开的映像文件中，可以理解为增量添加。

编辑映像	用于编辑映像文件的信息。





5.1.5. 卸载映像

当你映像挂载后，不需要了可以使用此功能，删除挂载点。

5.1.6. 保存映像

当挂载映像后，此功能可以将更改保存至原有 WIM 文件。如果选择的系统不是一个挂载映像，那么该操作等同于另存为映像。

温馨提示：保存映像并不会减少映像文件体积，如果你需要得到一个最优体积的映像文件，请使用格式转换——WIM<-->ESD/SWM 功能得到一个新的 WIM。

5.1.7. 另存为映像

此功能等同于 2.1 系统备份。

5.1.8. 格式转换

此处功能比较简单，只是单纯转换下格式，比如 WIM 与 ESD 互转、ESD 转 ISO 等等……

5.2. 恢复功能

当系统遇到问题时，这些功能可能可以帮助到你。

5.2.1. 验证受损

此功能仅 Windows 8 或者更高才能使用。

扫描映像是否受损，并报告系统是否受损以及能否修复。

5.2.2. 修复受损

此功能仅 Windows 8 或者更高才能使用。

修复系统文件错误。

5.2.3. 引导修复

此功能可以添加 BCD 引导信息，如果你的系统引导信息丢失或者损坏，可以使用此功能。

温馨提示：此功能依赖 Config\%PlatformShortName%\bcdboot.exe，删除此文件可能导致某些精简系统无法使用该功能。

5.2.4. 系统备份

此功能等同于 2.1 系统备份。

5.2.5. 系统还原

使用以前备份的映像文件（WIM/ESD/SWM）还原当前系统。

Dism++支持在线还原，但是需要 RE 环境完好。在线还原过程中，Dism++会临时释放到 Windows.BT 文件夹，重启后，再将老系统移动至 Windows.old，新系统从 Windows.BT 移动到当前位置，完成还原操作。

温馨提示：在线还原时，格式化分区以及添加引导功能不可用。如果你需要使用它们，请使用在 RE 中运行 Dism++然后在使用系统还原。

5.2.6. 在 RE 中运行

操作完成后，系统将重新启动，进入 RE 环境（Windows 恢复环境），并启动 Dism++。在这个环境中，系统彻底没有启动，可以随意进行格式化系统分区等。

5.2.7. 仅启动 RE

操作完成后，系统将重新启动，进入 RE 环境（Windows 恢复环境）。你可以使用 CMD 来执行你的操作。

5.2.8. 撤销 Compact

使用 CompactOS 后，如果你觉得电脑有问题可以使用此选项撤销，撤销后系统将恢复原样。

温馨提示：如果你的系统已经无法启动，你还可以在 RE/PE 中撤销。

5.3. 选项

更改 Dism++的默认设置，比如压缩率、排除列表等……

5.3.1. 详细设置

此处提供以下设置：

新手模式：启用后程序将隐藏存在风险的功能，以适合新手使用。
整合右键菜单：启用后将可以对单个文件进行 Compact 压缩或者解压，此外针对 ESD 文件提供 ESD 转 ISO 右键菜单。
整合引导菜单：启用后将在 BCD 引导中看到 Dism++选项，选择此条目可以即可启动 Dism++，方便系统维护。

排除列表：WIM/ESD 保存时跳过的文件，可以减少操作耗时。
CompactOS 压缩率：调整 CompactOS 的默认压缩率，压缩率越高，性能越差，反之性能越好，建议保持默认不要更改。

5.3.2. 操作完成后

你可以设置操作完成后关机还是重启。

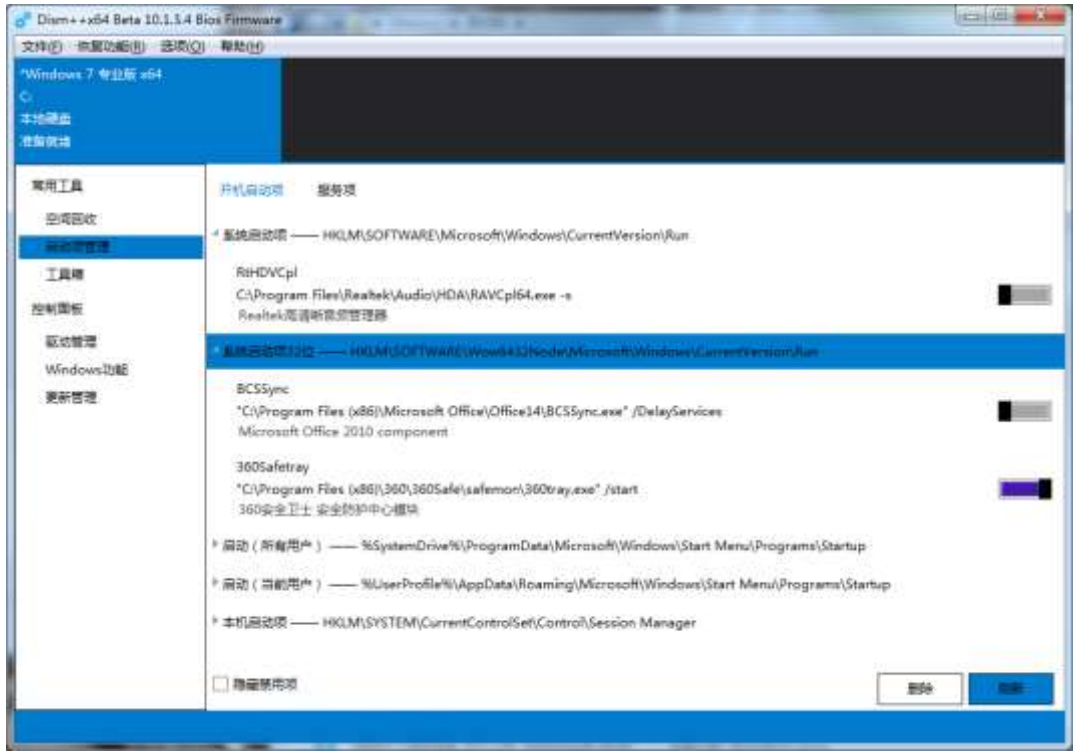
5.4. 帮助

此功能就不多说了，我用一句话解决。如果你有什么意见或者反馈欢迎通过网站（www.chuyu.me）或者邮箱（mingkuang@live.com）告诉我。

6. 启动项管理

6.1. 开机启动项

提供管理计算机中的启动项功能。

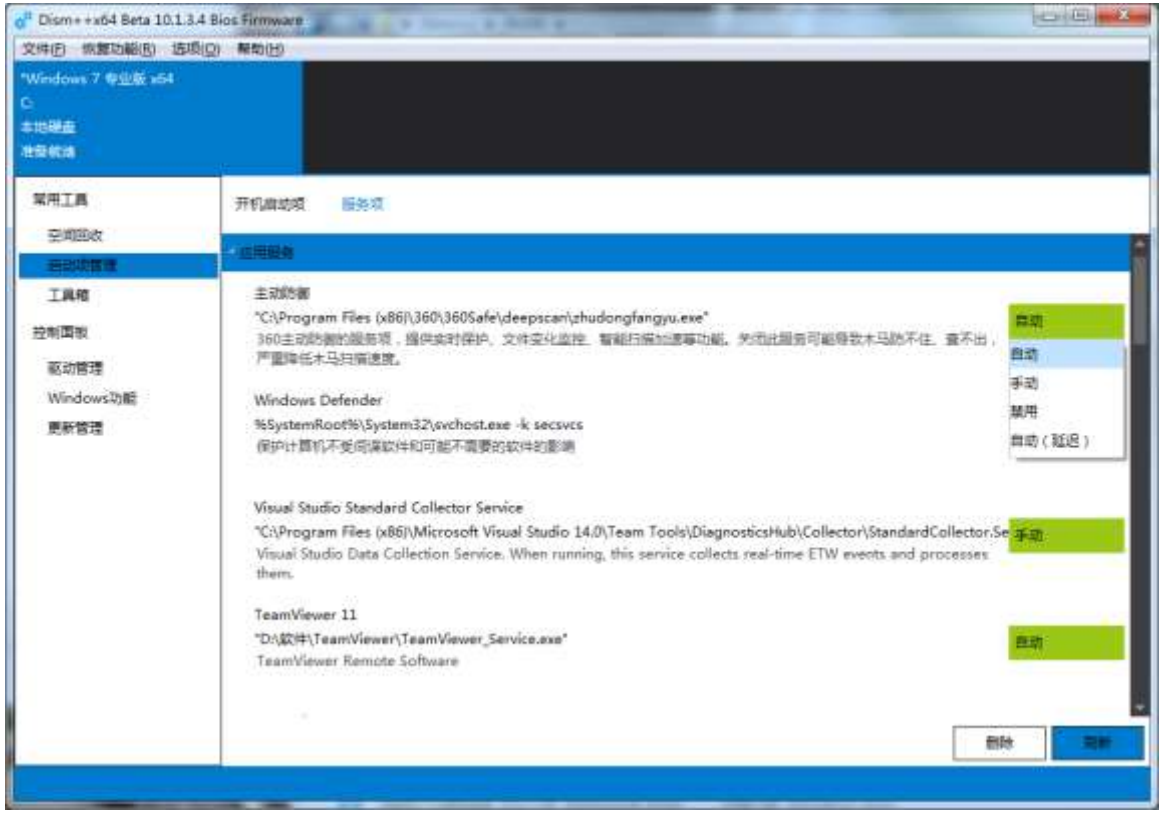


打开改功能后，程序会显示所有的开机启动项信息，如果你不需要某个启动项，可以点击右侧的按钮关闭。当你需要了，可以重新打开。此功能行为与 msconfig 一致。

名称	功能说明
隐藏禁用项	不显示已经禁用的项目。
删除	彻底删除此条目，而不是禁用，某些一次性条目只能删除，而不能禁用。警告：删除后无法恢复！
刷新	重新获取启动项信息。

6.2.服务项

名称	功能说明
删除	彻底删除此条目。警告：删除后无法恢复！
刷新	重新获取服务信息。



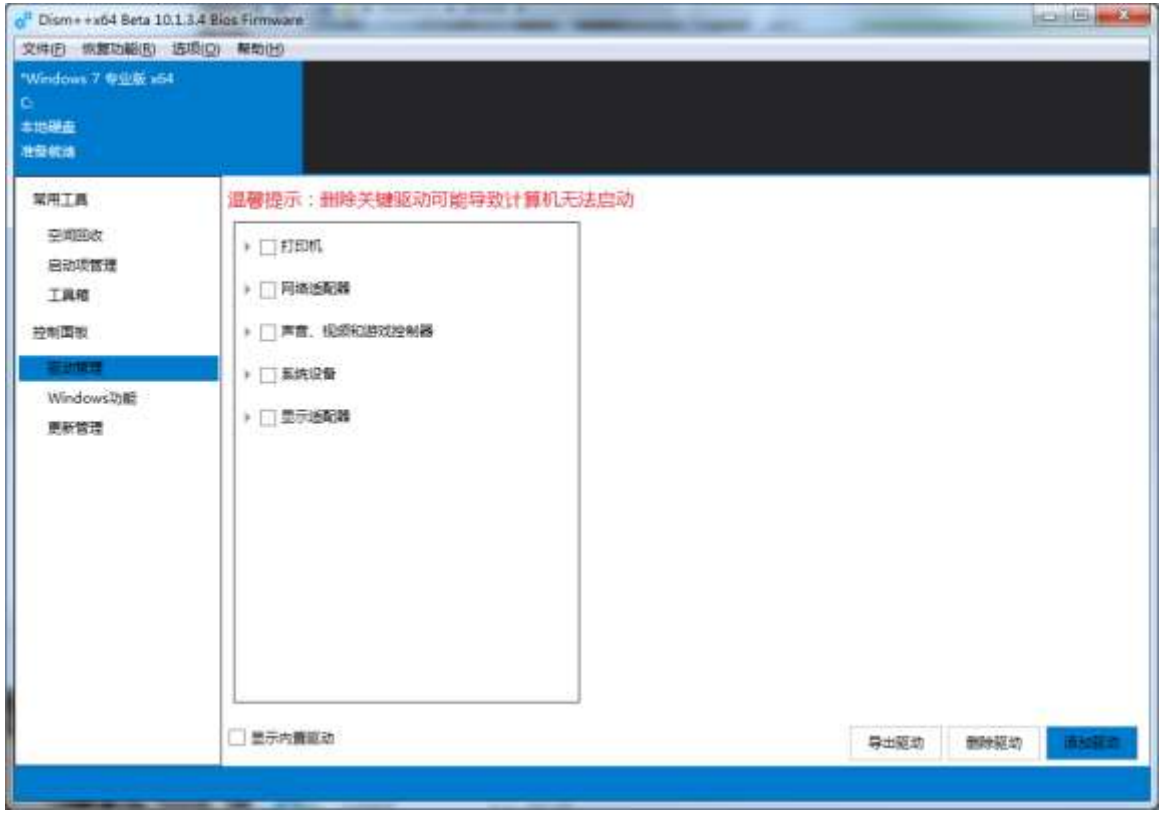
可以点击右侧绿色按钮，管理服务的启动状态。

7. 驱动管理

此功能需要 Windows 7 或者更高，更低版本尚未适配，不显示此功能。

驱动是生产力的灵魂，Windows 大概使用了 1/5 的空间用来存放各种驱动,可想而知驱动的重要性。但是现在随着你安装的驱动越来越多系统并不会删除那些驱动。典型的显卡驱动，每更新一次就留下了一个大大的包，有个网友还发现英伟达的显卡驱动已经占据了 10GB 的空间。

主要是因为正常情况下驱动只能添加，不能删除，所以导致驱动越来越多，体积越来越大，这个功能就是用来删除你不需要的第三方驱动。你可以在这里管理第三方驱动。



7.1.删除驱动

使用方式自然简单，在你需要删除的驱动上打勾，在点击删除驱动即可。

温馨提示：删除关键驱动可能导致系统无法启动。

温馨提示：选中显示内置驱动功能后，程序还将显示系统自带的驱动。如果你是 Windows 7 或者 Windows 2008 R2 系统，还能将他们删除。删除驱动操作是可逆的，如果你后悔了，还能将驱动重新添加至操作系统中。

7.2. 添加驱动

添加驱动顾名思义就是将驱动添加至系统中，此功能仅支持将 inf 形式的驱动安装包添加到系统。点击添加驱动时，程序会要求你选择一个文件夹。你只需要将你需要添加的驱动放在该文件夹或者子目录下（Dism++会自动遍历子文件以及子目录），里面存放的所有驱动都将添加到系统中。

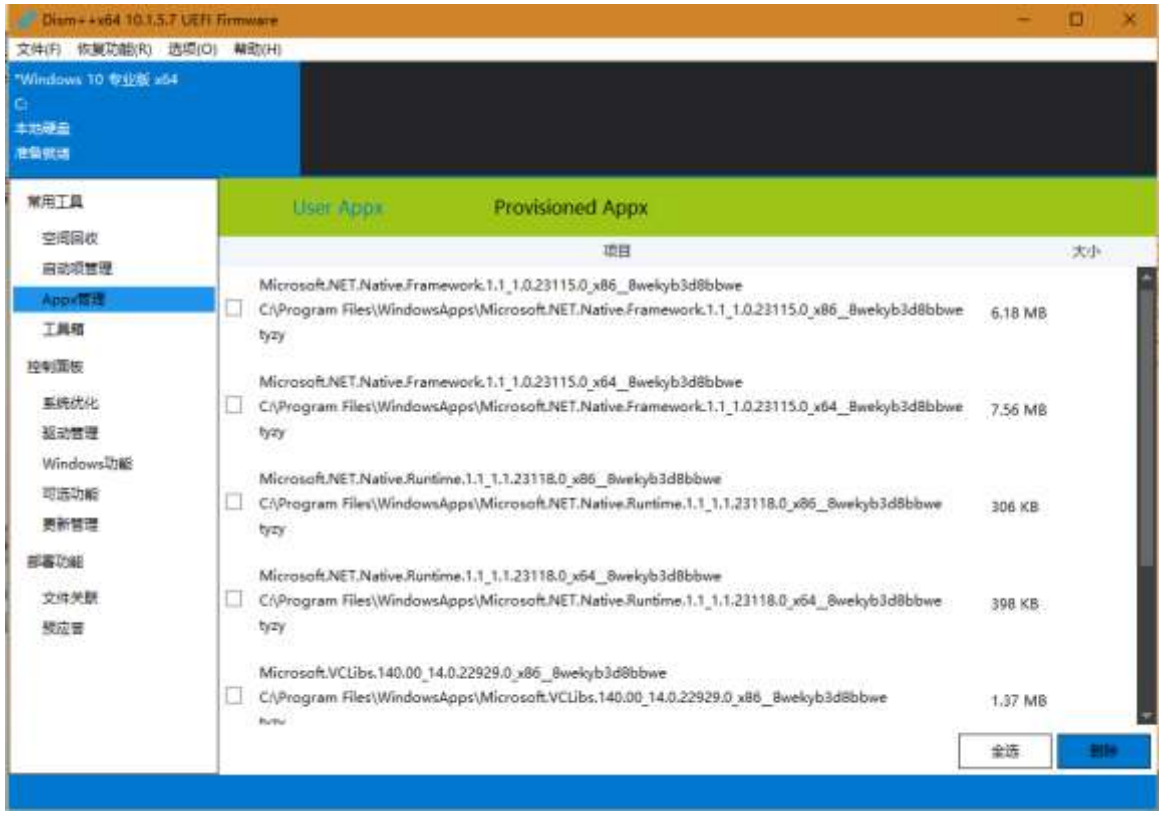
温馨提示：Dism++只支持 inf 形式的驱动安装包，如果你有 exe 等安装包。你可以先将驱动安装到电脑上，然后使用导出驱动功能得到 inf 形式的驱动安装包，在导入其他系统中。

8. Appx 管理

此功能用于删除不必要的 Appx 应用。

8.1. User Appx

此处显示的应用为用户拥有的应用。也就是在开始菜单中可以看到的应用。



你可以选中你不需要的 Appx，然后点击删除即可。某些 Appx 一直是 System 删不掉，那么可以试试旁边的 Provisioned Appx。

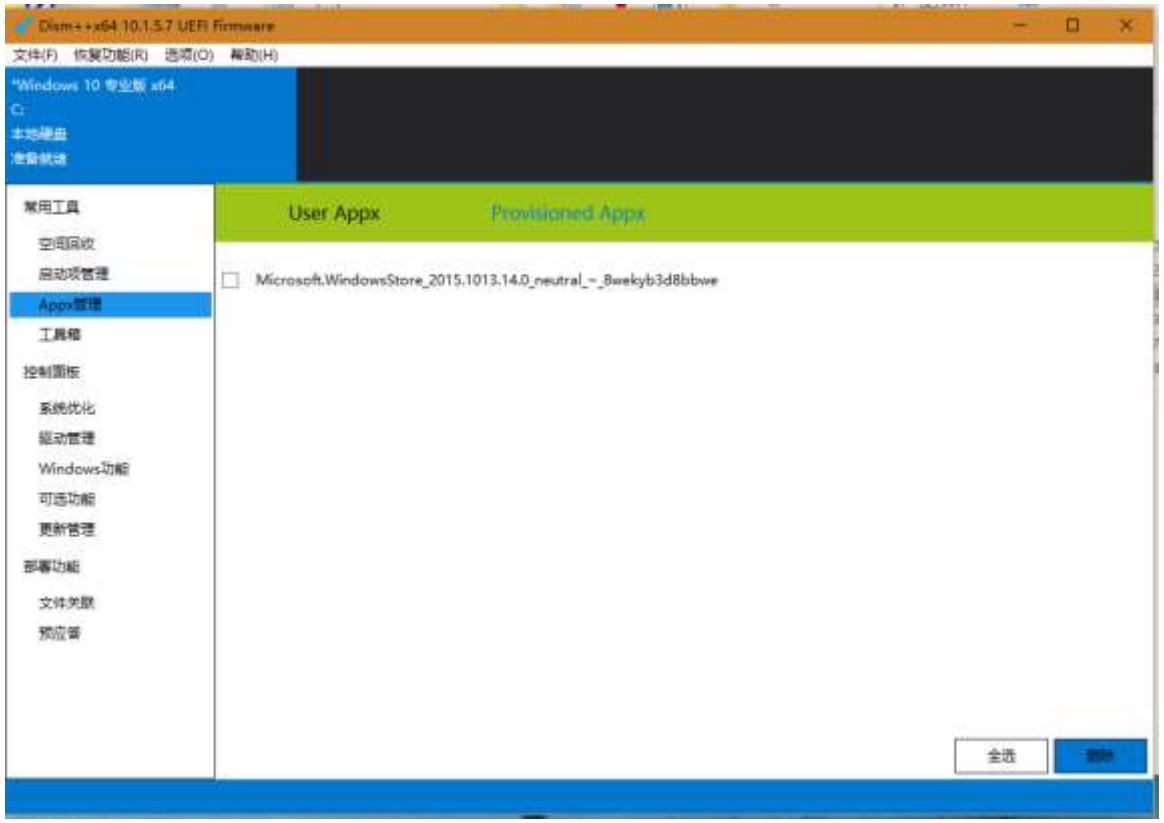
温馨提示：请勿手贱删除 Windows Store。删除后你的账号将无法通过应用商店安装应用。Dism++已经做了保护，删除 Store 前会弹窗警告。

温馨提示：封装人士必看！在线使用过期 Appx 清理后，请务必删除所有的 User Appx，不然一般化处理时将保存，原因就是存在 User Appx。

8.2. Provisioned Appx

Provisioned Appx 针对于未来新建用户，在你并且首次登录新建的用户时，系统会自动将 Provisioned Appx 部署到你的账号中。这也是为什么原版系统启动后就已经拥有一大把应用的原因。

如果你不需要他们，那么可以考虑删除这些应用。

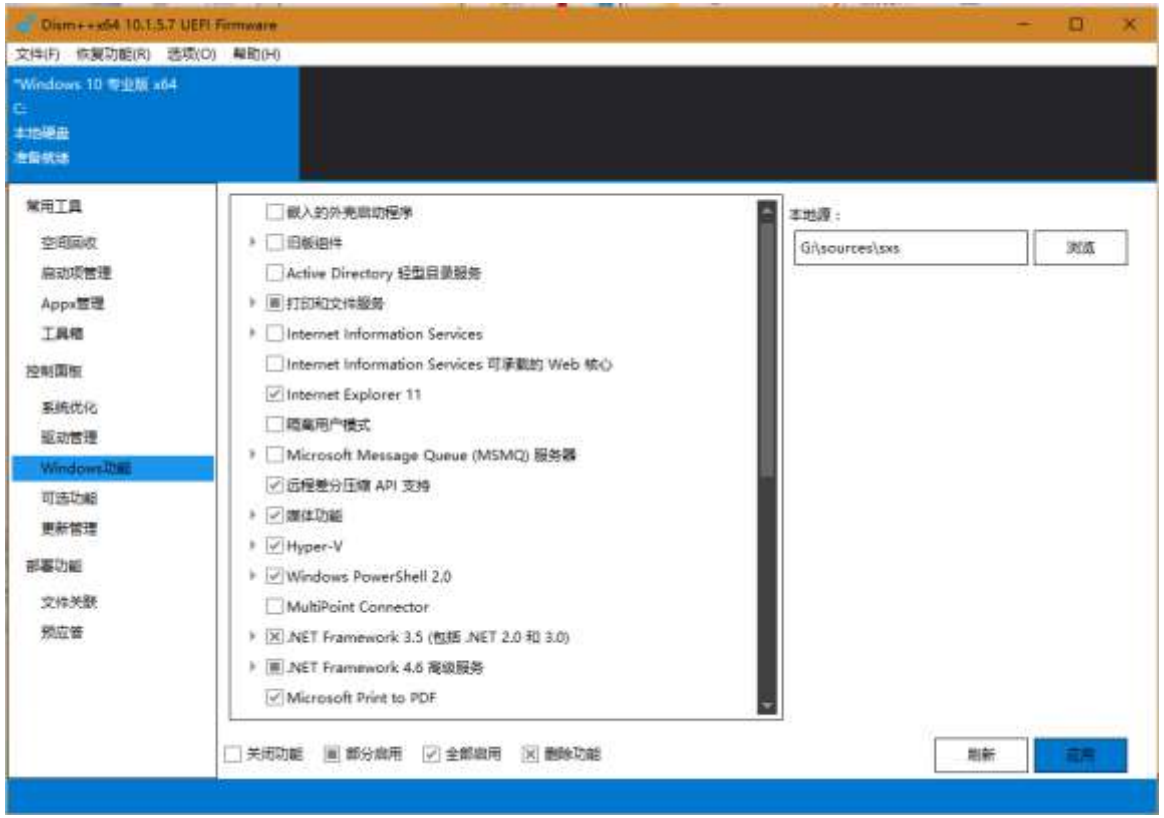


此功能也很简单，就是看到你不爽的应用，选中后删除。

温馨提示：请勿手贱删除 Windows Store。删除后你的账号将无法通过应用商店安装应用。Dism++已经做了保护，删除 Store 前会弹窗警告。

9. Windows 功能

这个功能也相对比较简单，在第一次启动 Windows 功能时，程序会检查当前计算机包含的功能，并且把当前的所有功能的状态显示在程序上。

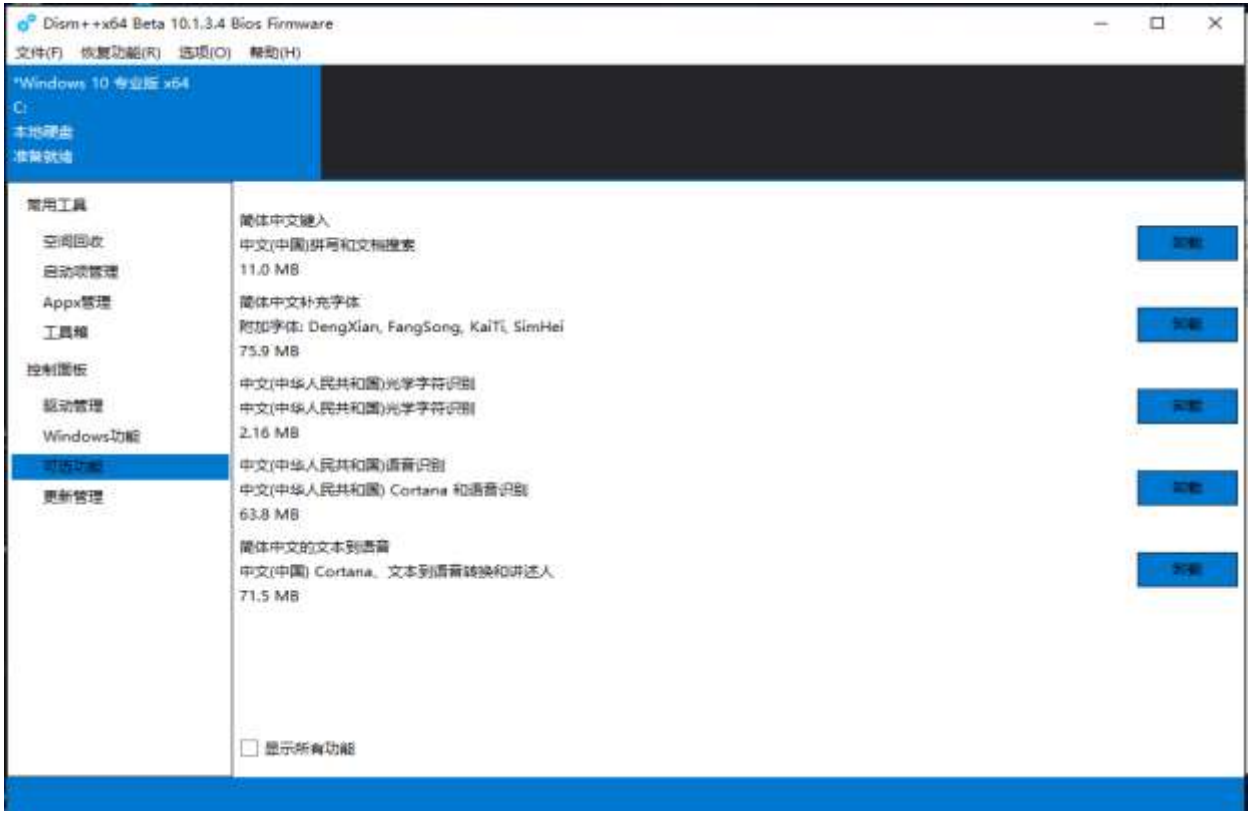


☑表示该功能以及子功能全部启用。

☐表示该功能以及子功能未启用。
■表示该功能已经启用，但是其子功能并未全部启用。
☒表示该功能及其子功能已经删除。（此状态仅 Windows 8 或者更高支持）
你可以更改功能状态，然后点击应用即可。

温馨提示：对于启用已经删除的功能时，可以设置本地源可以大幅度加快功能的启用速度。你可以把 ISO 挂载下，Dism++会自动自动将你挂载 ISO 作为本地源，如图所示，Dism++自动把 G:\sources\sxs 作为了本地源。

10. 可选功能



看到你不需要的功能，点击卸载。启用显示所有功能后，还能显示未安装的功能，你可以在这里添加他们。

11. 更新管理

11.1. Windows Update

此功能也很简单，点击扫描，然后在选择自己想要安装的更新后点击安装即可。更新数据库从 WSUS 服务器重全自动导出而成。

目前已经在 Windows Update 中收录 Windows 7、Windows 8.1、Windows 10、Windows 2008 R2、Windows 2012 R2 的所有系统更新。以及 Office 2010、Office 2013、Office 2016 所有更新（Office 更新只能在线安装，无法离线进行）。当然过期更新已经全部删除，恶意扫描工具也已经删除。

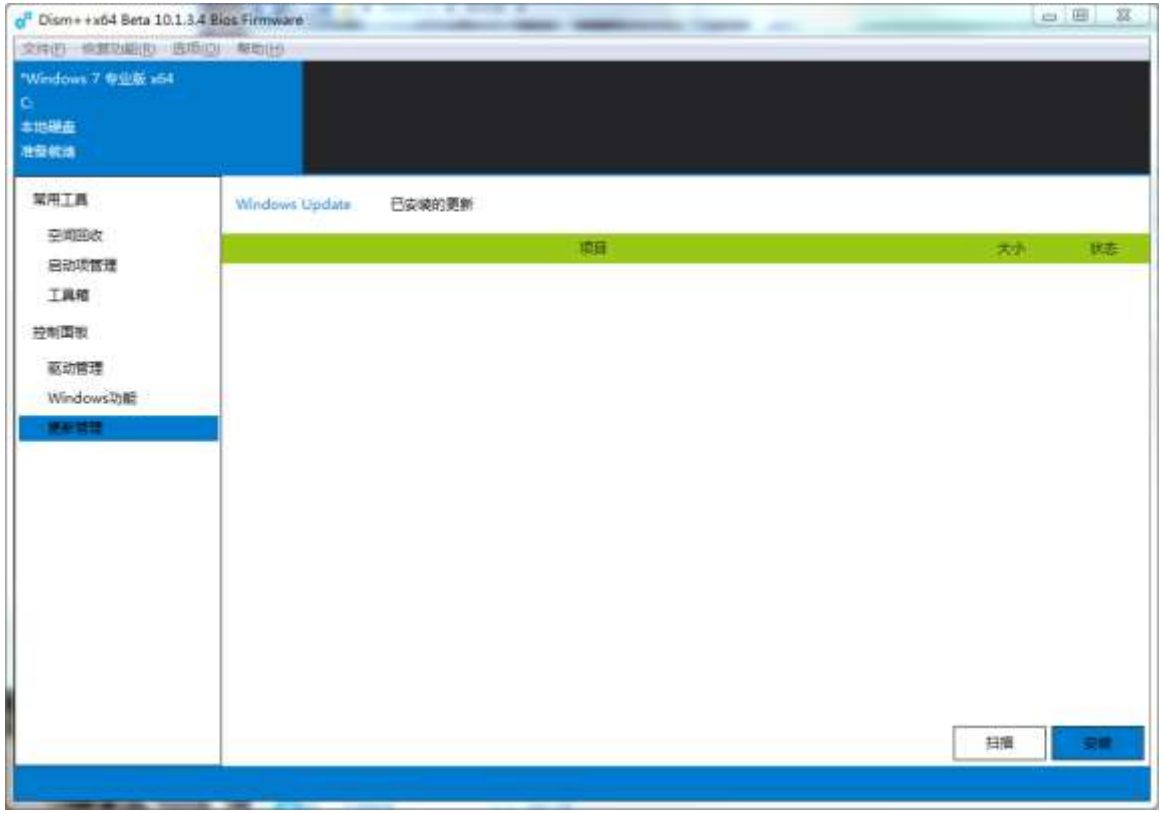
此外 Dism++会把下载的更新缓存在程序目录的“Config\UpdateCache”文件夹，下次需要时将直接跳过下载

温馨提示：某些更新直接提示失败的或者提示部分成功的，说明他存在 exe 或者 msp 方式的更新，此功能目前仅支持 cab 方式的更新，未来版本将支持其他形式的更新。

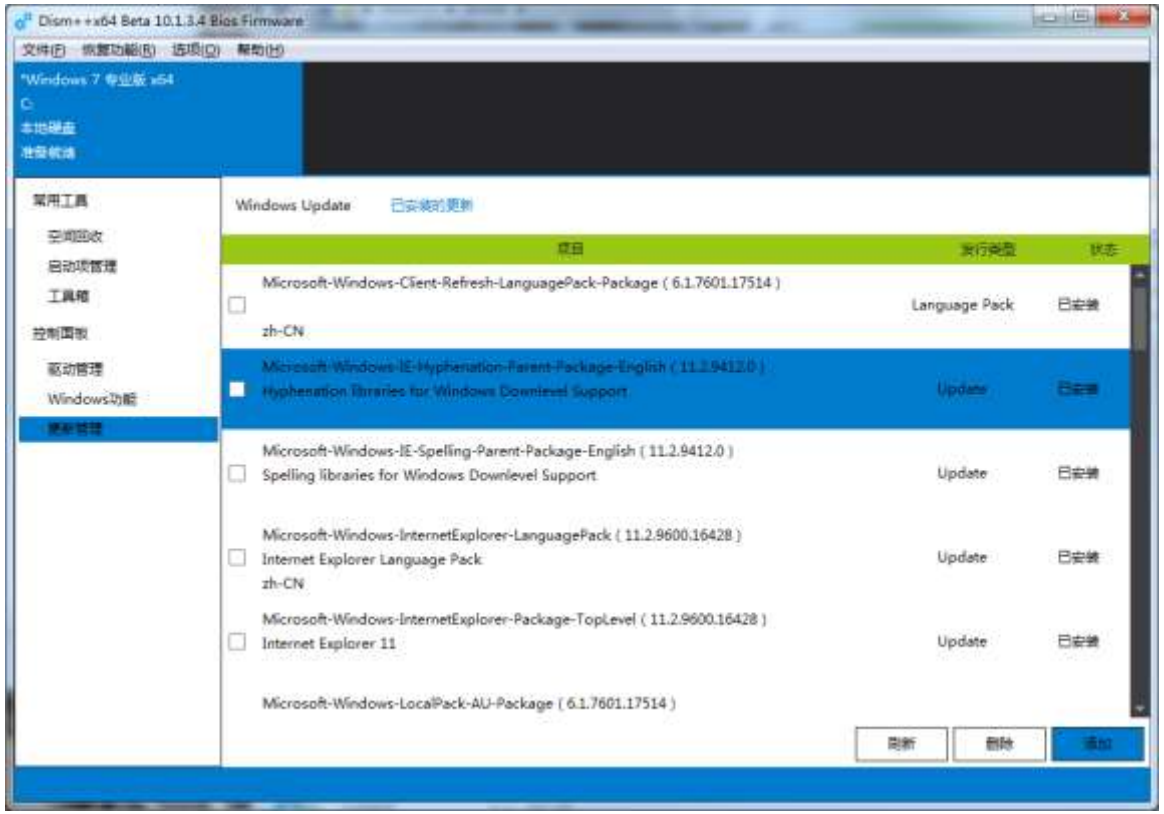
温馨提示：带橙色标记的更新是风险更新，大概意思就是安装后可能会有问题。默认选中的是微软默认推荐安装的。

11.1.1. 关于差异传输引擎

Dism++在下载更新过程中会默认使用差异传输引擎，来减少网络传输流量。一般可以节省 90%的网络传输流量（比如 1GB 的补丁实际仅需下载 120MB 左右）。但是使用差异传输后，Dism++仅下载本机需要的更新文件，可能不会下载完整安装包，这很可能导致下载的文件无法使用在其他计算机中。如果你有离线断网使用要求，请先详细设置中关闭差异传输引擎。


Dism++x64 Beta 10.1.3.4 Bios Firmware
*Windows 7 专业版 x64
C:
本地磁盘
常用工具
空间回收
启动项管理
工具箱
控制面板
驱动管理
Windows功能
Windows Update
已安装的更新
项目
大小
状态
扫描

11.2. 已安装的更新



名称	功能说明
刷新	重新获取已安装的更新信息。

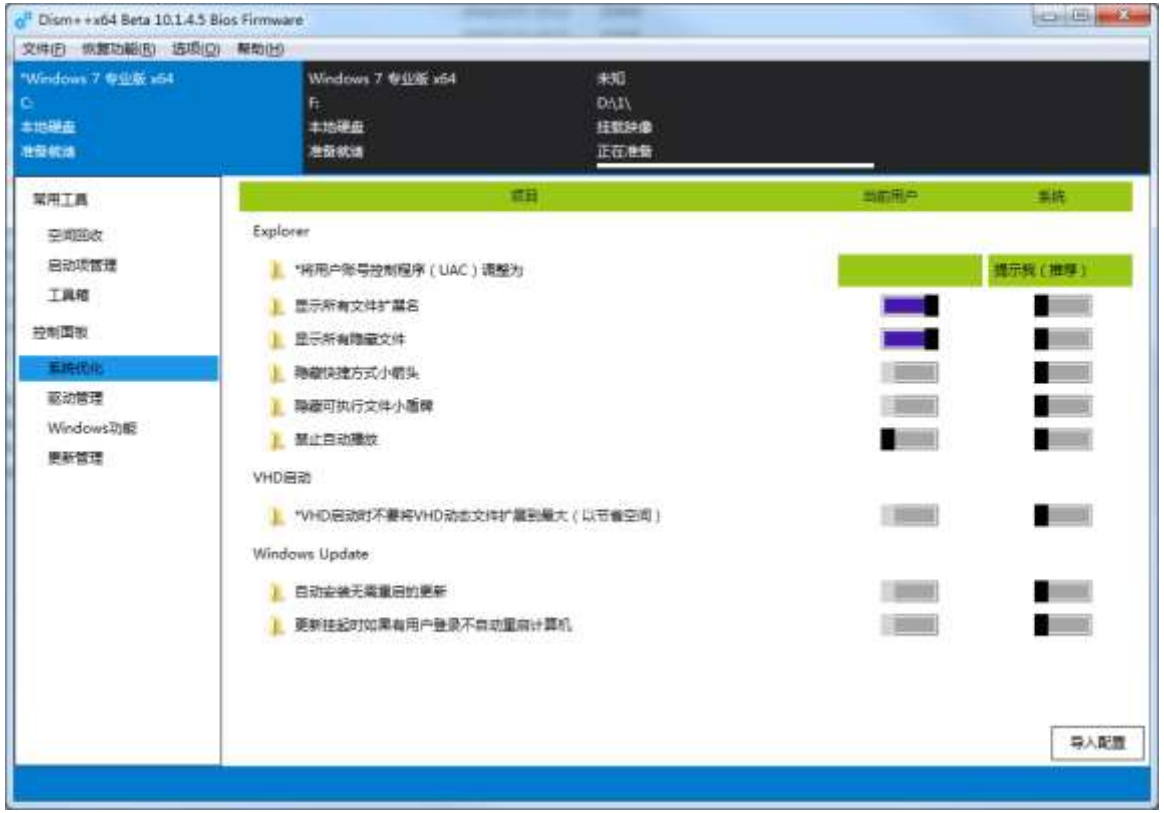
删除	删除选中更新。（如果更新状态是已固化，那么该更新无法删除）
添加	选中一个或者多个 mus、cab 文件，添加至系统。你可以安装 Ctrl 一次性选择多个更新。

以下是更新各个状态的解释，一般情况下，你无需了解这些内容，Dism++的更新清理会自动删除不需要的更新。

更新状态	状态详细说明
未知	具体情况不明。这个状态一般不会出现，除非微软增加了新的定义。
不适用	此更新不适用于此计算机，换句话说装了也是白装。
卸载挂起	卸载挂起，重启后才能彻底删除。用户使用删除更新后可能引入此状态。
已暂存	更新已经添加到 WinSxS 中，但是未生效，你可以手动为此更新切换到已安装。
已安装	更新正常安装，一般更新安装后就是这个结果，没什么好解释的。
安装挂起	即将完成安装，但是还有部分文件未更新，重启计算机后变成已安装。
被取代	此更新被其他更新所取代。简而言之就是这个更新没用了。
已固化	此更新已经被 Dism/Dism++清理，已经永久固化，效果等同与已安装，但是无法删除。

12. 系统优化

10.1.4.5 版本新增功能，用于调整系统常用设置。选择或者更改后立即生效，部分带*项目可能需要重启生效。如果你有更好的优化条目，欢迎来搞。



系统优化的设置分为当前用户以及系统，因为某些设置是保存在 HKEY_CURRENT_USER（用户配置文件）中的，修改保存在用户配置文件中的设置只影响当前用户，我们把这些设置都放到了当前用户下方。还有的是系统的，比如默认账号（HKEY_USERS\DEFAULT 在新建账号时，默认配置就是从默认账号中复制的，所以 Dism++提供了默认账号修改）以及 HKEY_LOCAL_MACHINE（修改这里的设置将影响所有的用户）。

温馨提示：离线使用时设置当前账号的相关设置均无法使用，看到一排灰色勿惊讶。

12.1. 导入配置

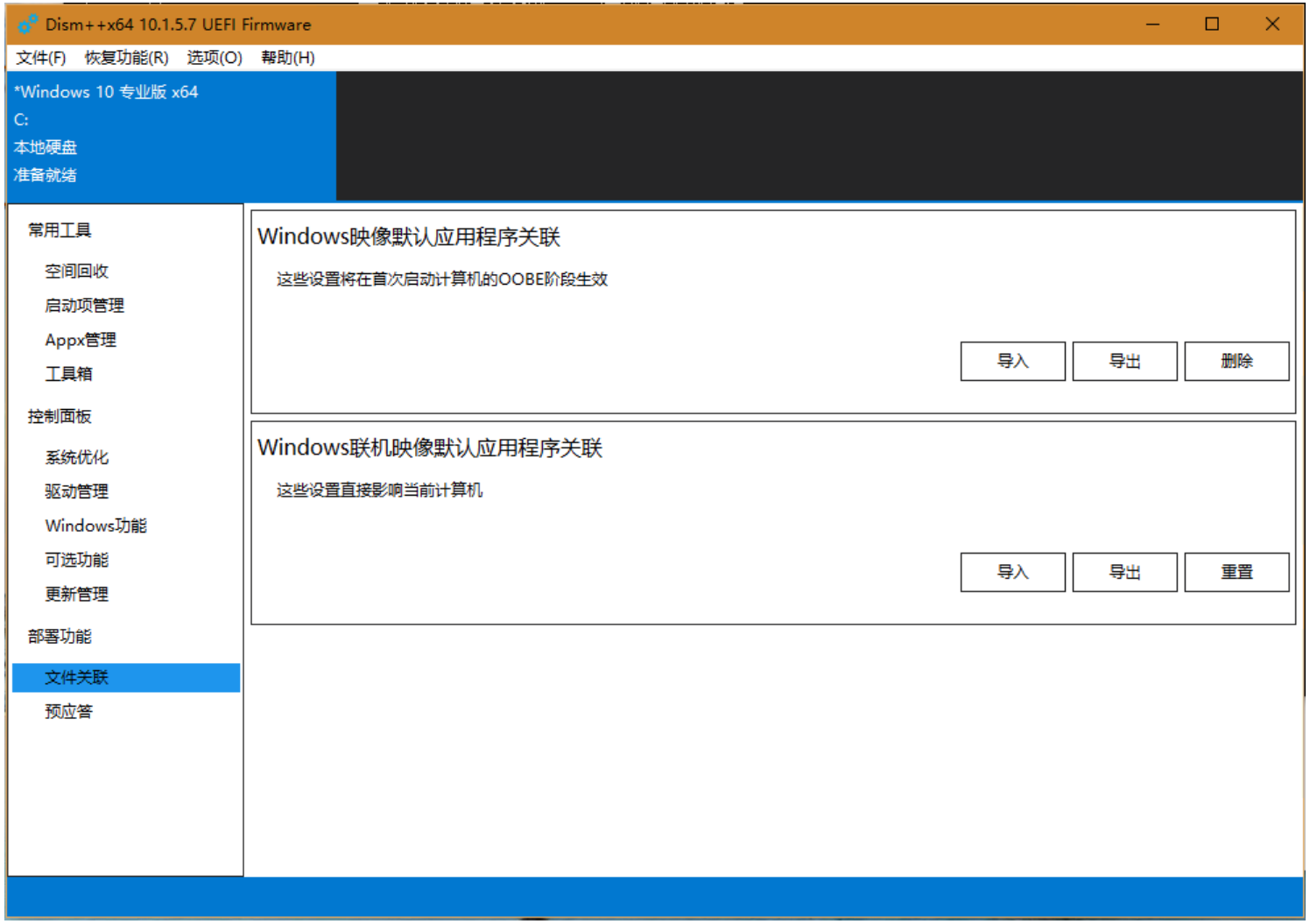
此功能支持将现有的 reg 文件导入到选中的系统中。对于离线系统，Dism++将自动完成重定向操作，你无需额外对 reg 文件进行调整。以下是离线系统 Dism++重定向细节。

源注册表	重定向后路径（X 表示离线系统路径）
HKEY_LOCAL_MACHINE\COMPONENTS	HKEY_LOCAL_MACHINE\ {bf1a281b-ad7b-4476-ac95-f47682990ce7}X:/Windows/System32/config/COMPONENTS
HKEY_LOCAL_MACHINE\SAM	HKEY_LOCAL_MACHINE\ {bf1a281b-ad7b-4476-ac95-f47682990ce7}X:/Windows/System32/config/SAM
HKEY_LOCAL_MACHINE\SOFTWARE	HKEY_LOCAL_MACHINE\ {bf1a281b-ad7b-4476-ac95-f47682990ce7}X:/Windows/System32/config/SOFTWARE
HKEY_LOCAL_MACHINE\SYSTEM	HKEY_LOCAL_MACHINE\ {bf1a281b-ad7b-4476-ac95-f47682990ce7}X:/Windows/System32/config/SYSTEM
HKEY_USERS\.DEFAULT	HKEY_LOCAL_MACHINE\ {bf1a281b-ad7b-4476-ac95-f47682990ce7}X:/Windows/System32/config/DEFAULT
HKEY_USERS\DEFAULT	HKEY_LOCAL_MACHINE\ {bf1a281b-ad7b-4476-ac95-f47682990ce7}X:/Users/Default/NTUSER.DAT
HKEY_CURRENT_USER	HKEY_LOCAL_MACHINE\ {bf1a281b-ad7b-4476-ac95-f47682990ce7}X:/Users/Default/NTUSER.DAT
HKEY_CLASSES_ROOT	HKEY_LOCAL_MACHINE\ {bf1a281b-ad7b-4476-ac95-f47682990ce7}X:/Windows/System32/config/SOFTWARE/Classes

通过上表可以明白，Reg 配置文件中的路径，Dism++将自动完成对应的重定向操作。

13. 文件关联

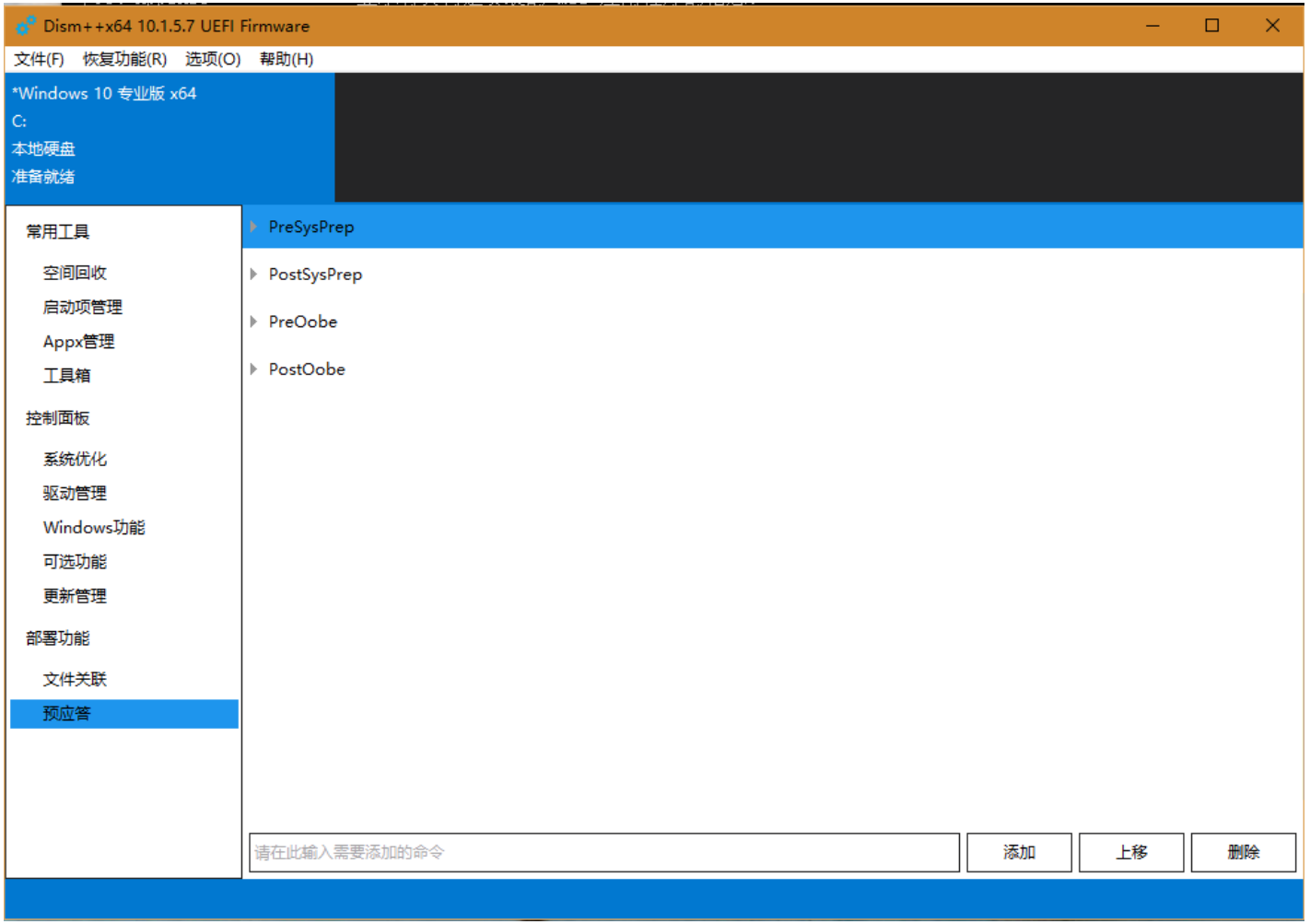
此功能等效于 Dism　DefaultAppAssociations 系列命令。用于操作文件打开方式。


Dism++x64 10.1.5.7 UEFI Firmware
文件(F)
恢复功能(R)
选项(O)
帮助(H)
*Windows 10 专业版 x64
C:
本地硬盘
准备就绪
常用工具
空间回收
启动项管理
Appx管理
工具箱
控制面板
系统优化
驱动管理
Windows功能
可选功能
更新管理
部署功能
文件关联
预应答
Windows映像默认应用程序关联
这些设置将在首次启动计算机的OOBE阶段生效
导入
导出
删除
Windows联机映像默认应用程序关联
这些设置直接影响当前计算机
导入
导出
重置

其使用方法也很简单，把联机映像的默认应用程序关联导出，在从 Windows 映像默认程序管理导入即可。

14. 预应答

这功能说的很简单，但是使用很复杂。预应答只是单纯添加执行命令，系统在首次启动时会在不同的时机分别执行他们。
另外命令有执行先后顺序，Dism++显示的越靠前就越先被执行。


Dism++x64 10.1.5.7 UEFI Firmware
文件(F) 恢复功能(R) 选项(O) 帮助(H)
*Windows 10 专业版 x64
C:
本地硬盘
准备就绪
常用工具
空间回收
启动项管理
Appx管理
工具箱
控制面板
系统优化
驱动管理
Windows功能
可选功能
更新管理
部署功能
文件关联
预应答
PreSysPrep
PostSysPrep
PreOobe
PostOobe
请在此输入需要添加的命令
添加
上移
删除

15. WinPE 命令

此功能仅在 PE 系统中显示，主要用于调整缓存大小，以及临时目录设置。这些都很简单，就不多说了。

16. Dism++实践

16.1. 使用 WIMBoot/Compact 方式安装 Windows 7

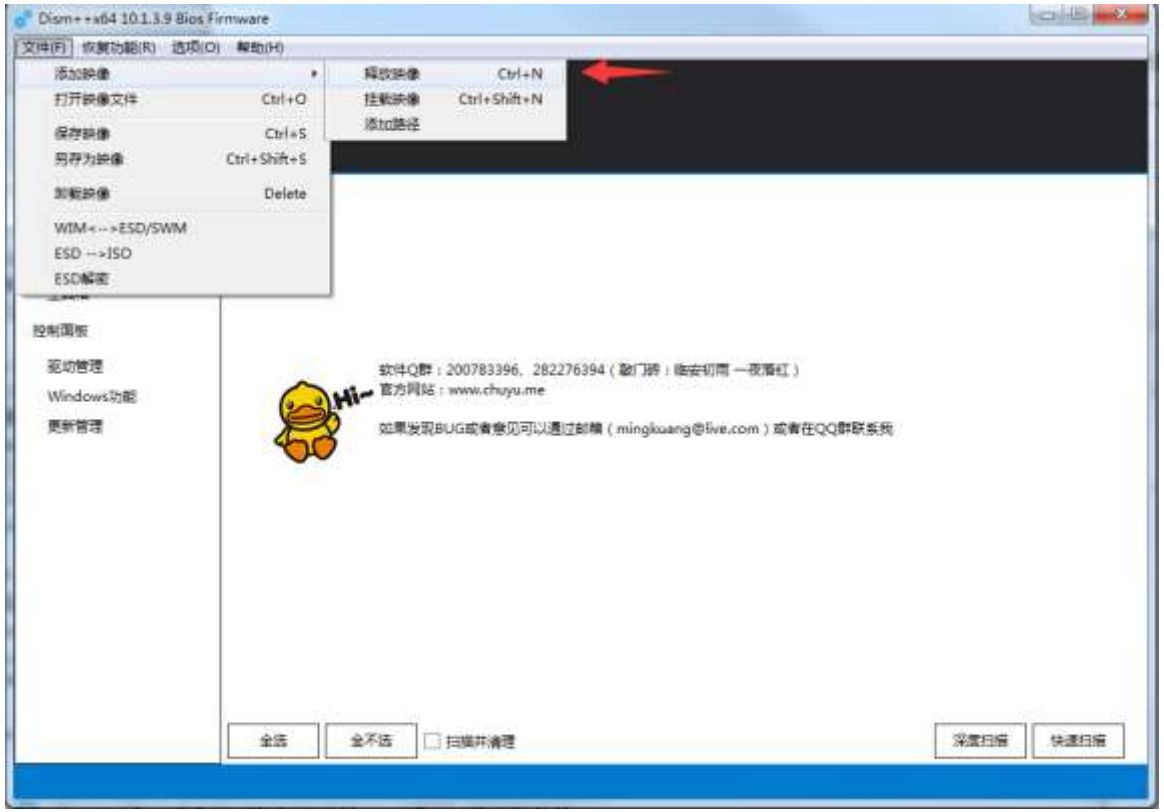
Dism++支持直接将原版系统以 WIMBoot 或者 Compact 方式安装，下面就来看看吧。（此功能需要 Windows 7 或者更高！）

16.1.1. 准备原版 ISO

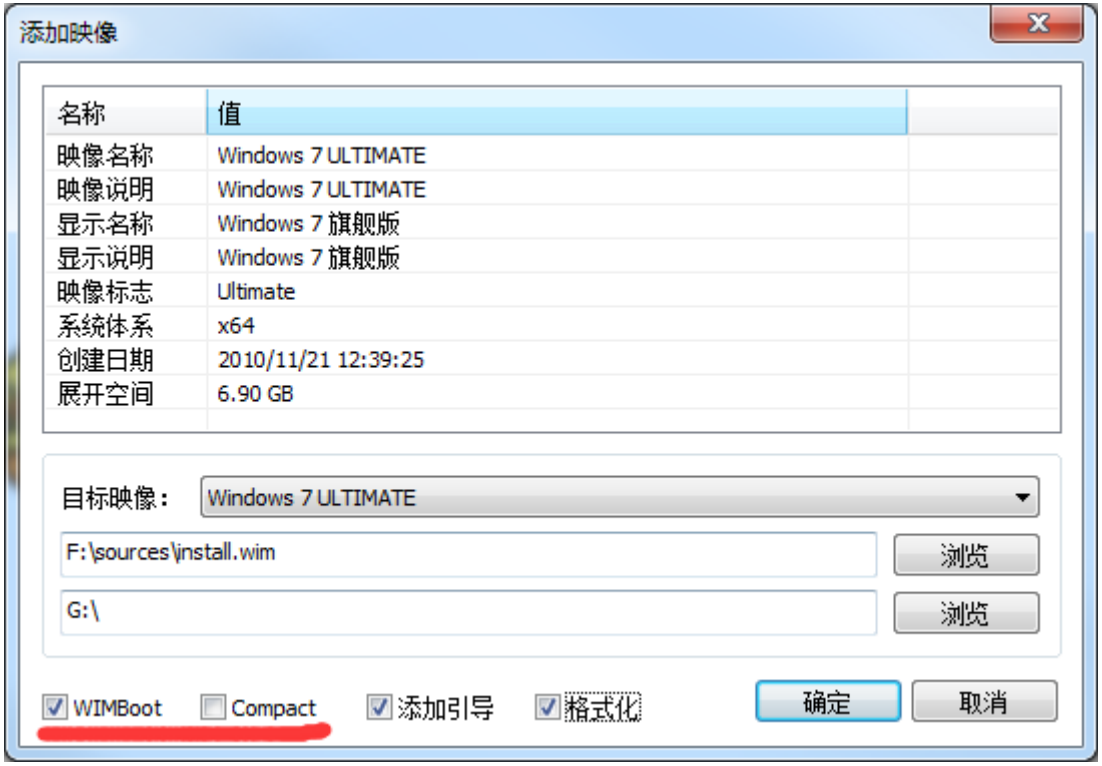
原版系统可以去 http://msdn.itellyou.cn/下载，虽然这个网站是第三方的，但是他上面的系统确实是微软原版的。如果你喜欢其他修改版，Dism++也是支持的，但是强烈建议大家使用原版或者未精简的系统。

温馨提示：制作 WIMBoot 启动不能使用虚拟光驱，请务必将 install.wim 复制到物理分区。

16.1.2. 启动 Dism++

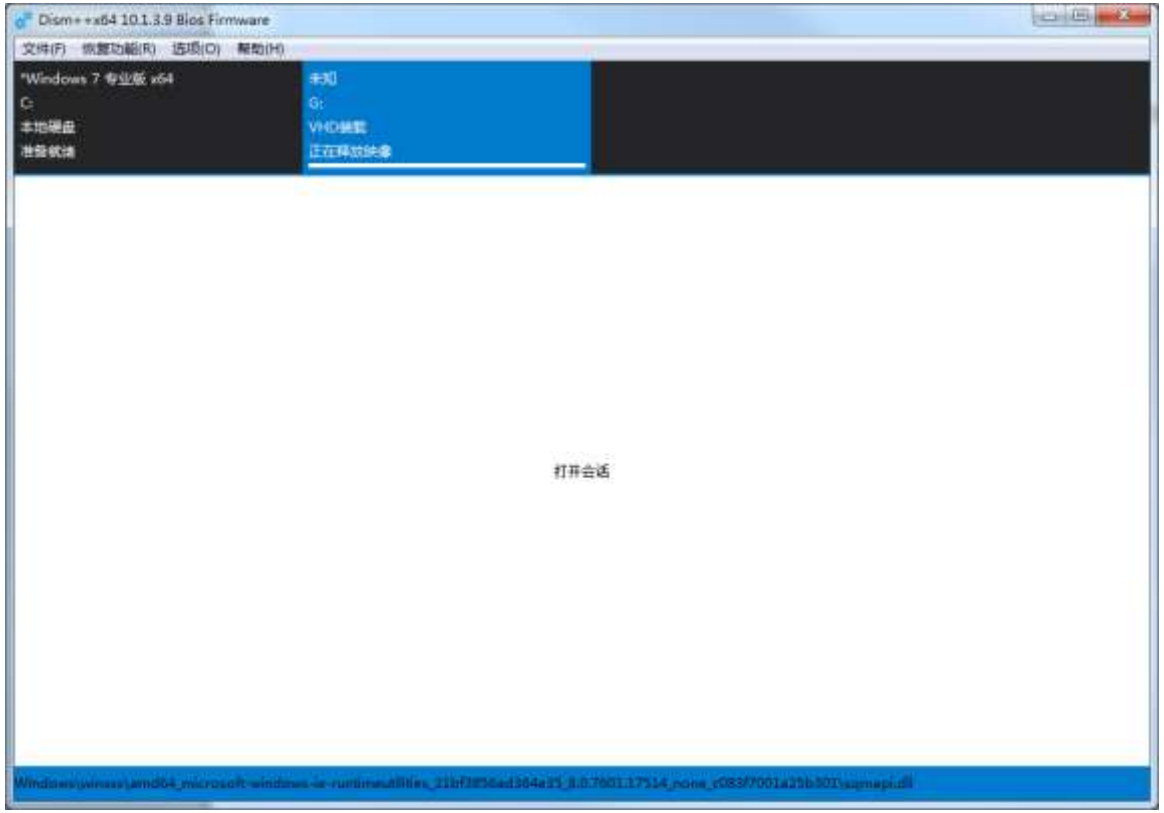


启动 Dism++然后选择释放映像。



如上填写信息，如果你需要使用 WIMBoot 则勾选 WIMBoot，需要 Compact 则勾选 Compact。注意：WIMBoot 跟 Compact 不可同时使用，不然程序会提示错误。在次提示：制作 WIMBoot 启动不能使用虚拟光驱，请务必将 install.wim 复制到物理分区。

完成后，程序会提示你选择启动分区，如果你看不懂建议你直接确定。



16.1.3. 享受 WIMBoot/Compact 技术

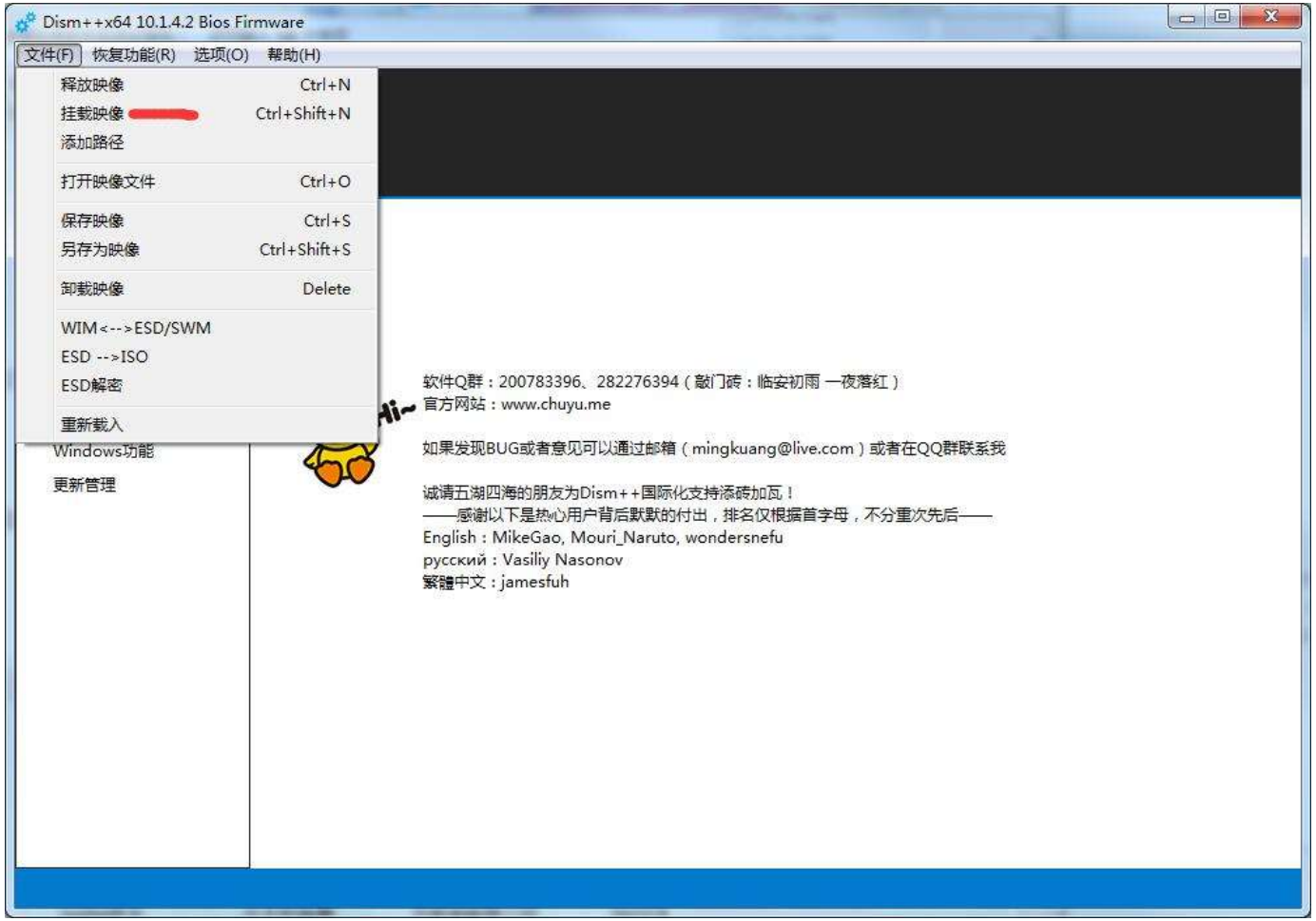
耐心等待释放完成……完成后重启电脑即可使用 WIMBoot。请尽情使用吧~~~

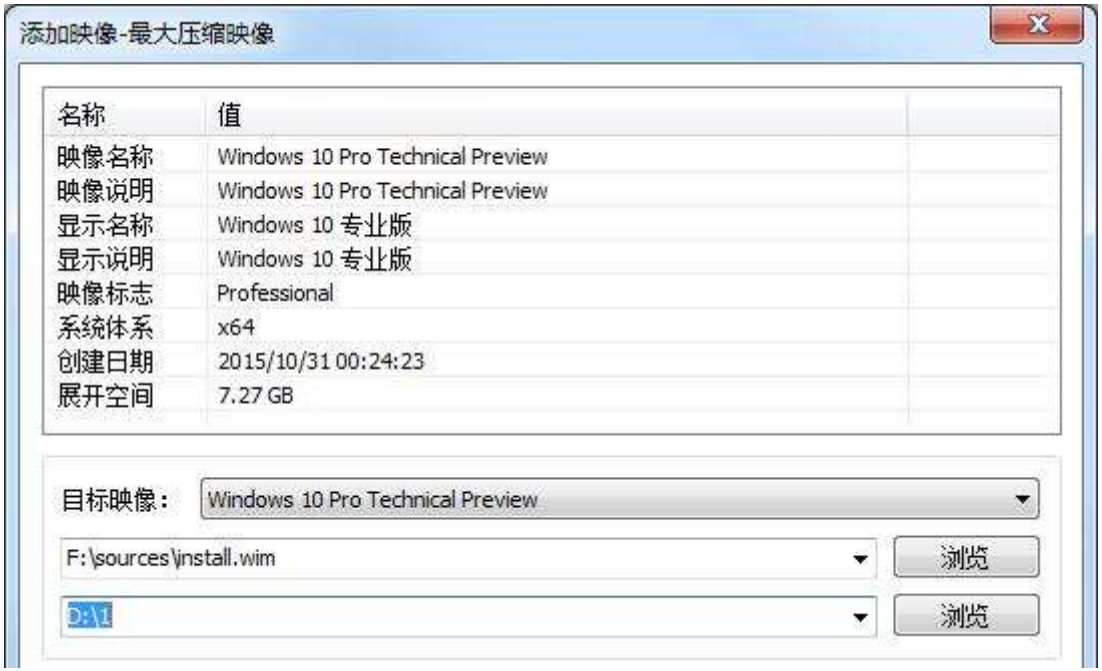
16.2. 给离线系统集成更新

大家是否厌倦了每次安装补丁后又要打一堆补丁的感觉？使用微软的 Dism 打补丁又需要自己收集补丁，而且每个月有新补丁，还有老补丁被取代。现在 Dism++重建了 Windows Update，直接从 WSUS 服务器中导出补丁数据库~~~全自动获取最新更新，自动剔除过期老补丁,在通过 CBS 判断系统是否需要此更新。下面来看看怎么使用吧~~~~

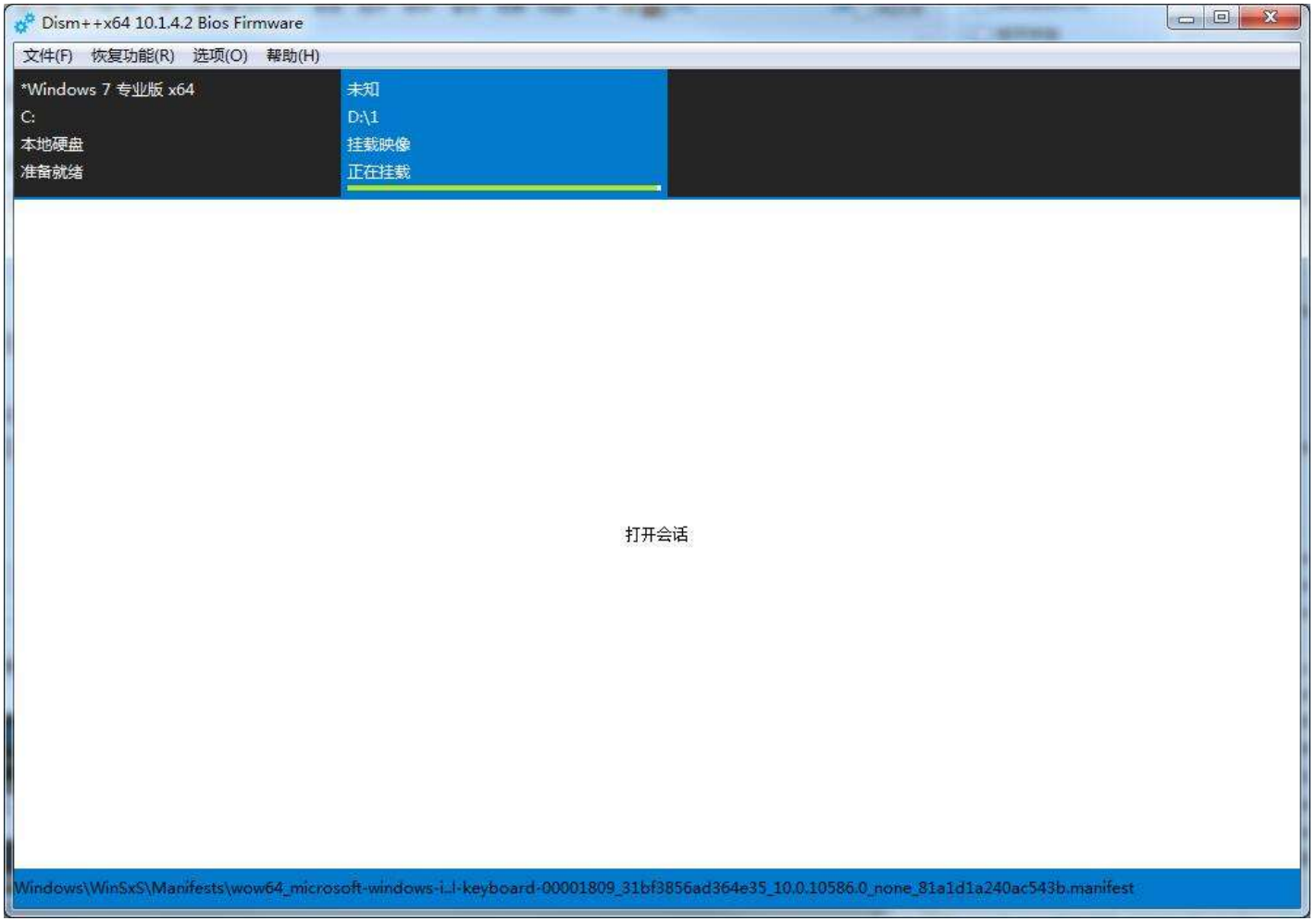
16.2.1. 挂载 install.wim

点击文件菜单 选择挂载映像。


Dism++x64 10.1.4.2 Bios Firmware
文件(F)
恢复功能(R)
选项(O)
帮助(H)
释放映像 Ctrl+N
挂载映像 Ctrl+Shift+N
添加路径
打开映像文件 Ctrl+O
保存映像 Ctrl+S
另存为映像 Ctrl+Shift+S
卸载映像 Delete
WIM<-->ESD/SWM
ESD -->ISO
ESD解密
重新载入
Windows功能
更新管理
软件Q群：200783396、282276394（敲门砖：临安初雨 一夜落红）
官方网站：www.chuyu.me
如果发现BUG或者意见可以通过邮箱（mingkuang@live.com）或者在QQ群联系我
诚请五湖四海的朋友为Dism++国际化支持添砖加瓦！
——感谢以下是热心用户背后默默的付出，排名仅根据首字母，不分重次先后——
English：MikeGao, Mouri_Naruto, wondersnefu
русский：Vasiliy Nasonov
繁體中文：jamesfuh



解压你的 ISO，然后找到里面的 install.wim，在选择一个空文件夹作为挂载路径，最后点击确定即可

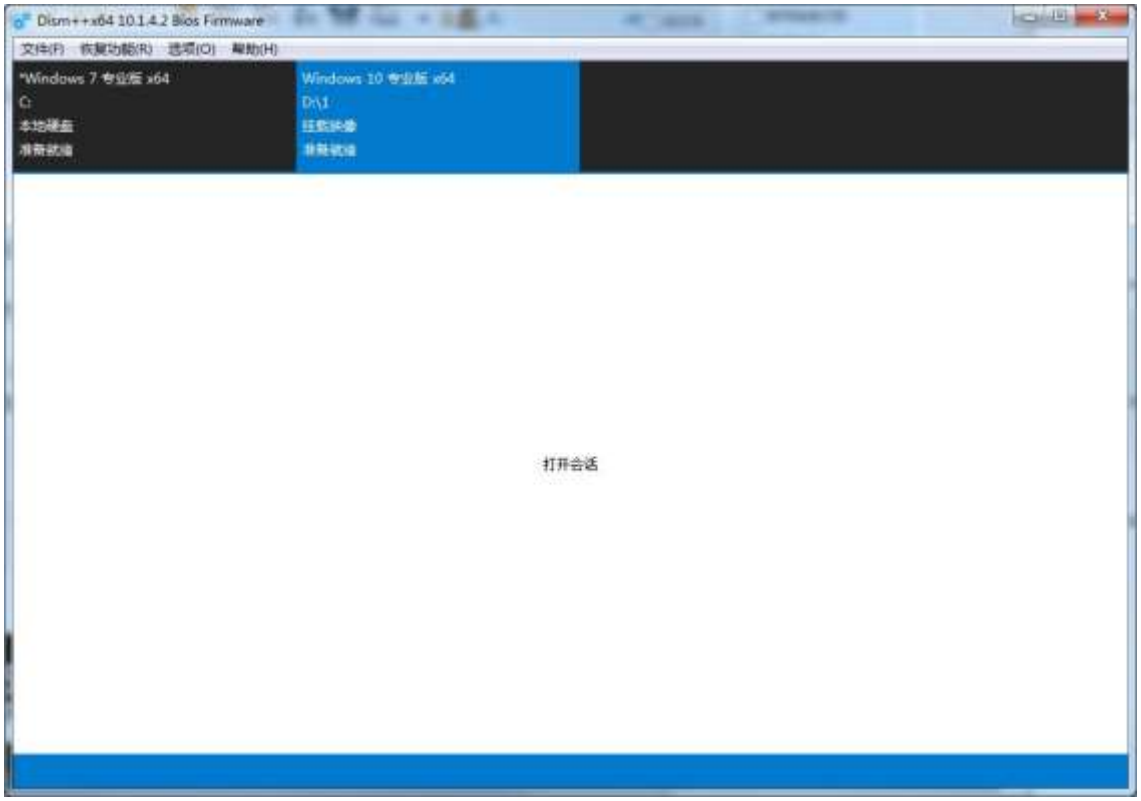

Dism++x64 10.1.4.2 Bios Firmware
文件(F) 恢复功能(R) 选项(O) 帮助(H)
*Windows 7 专业版 x64
C:
本地硬盘
准备就绪
未知
D:\1
挂载映像
正在挂载
打开会话
Windows\WinSxS\Manifests\wow64_microsoft-windows-i..l-keyboard-00001809_31bf3856ad364e35_10.0.10586.0_none_81a1d1a240ac543b.manifest

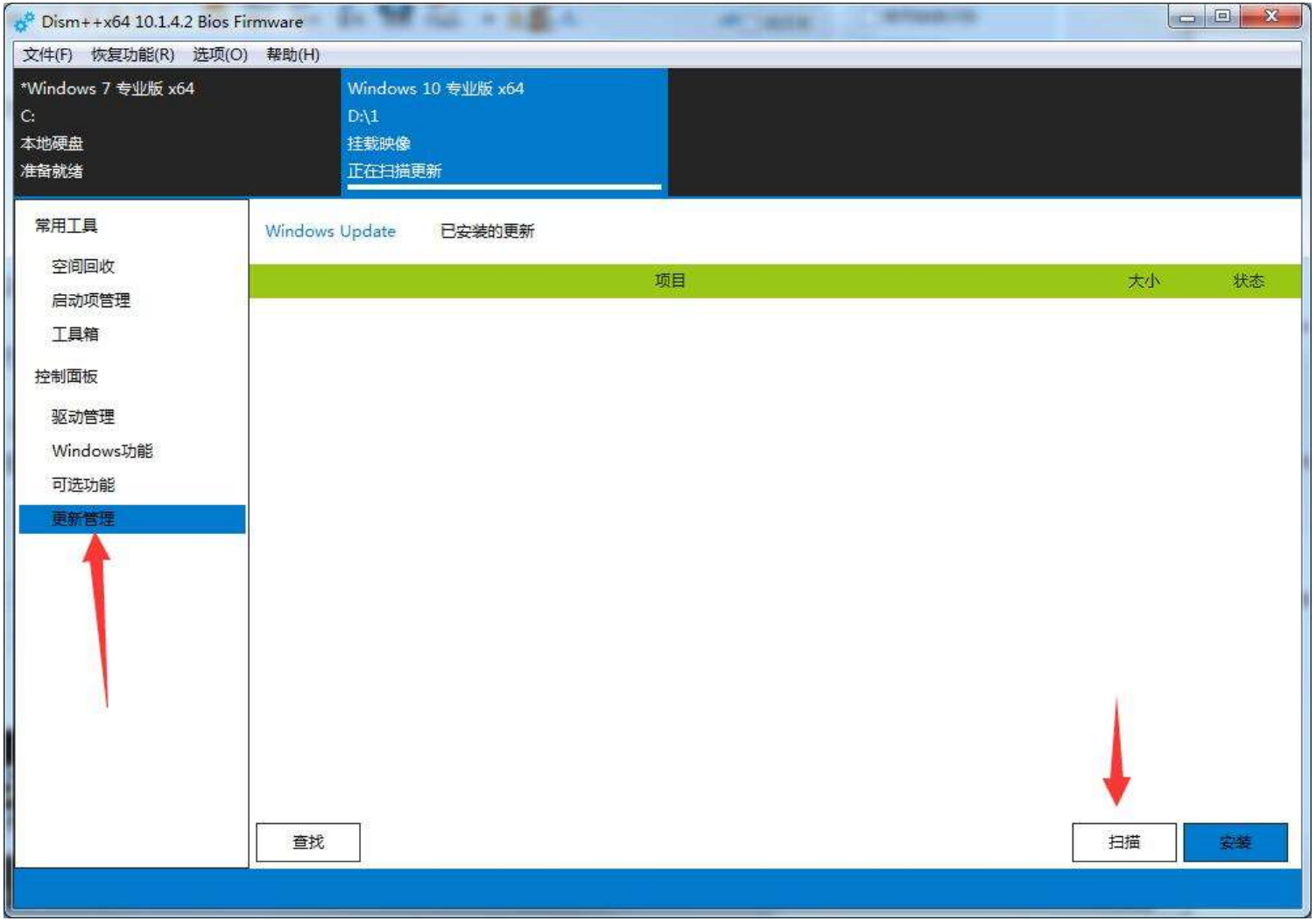
耐心的等待挂载完成……

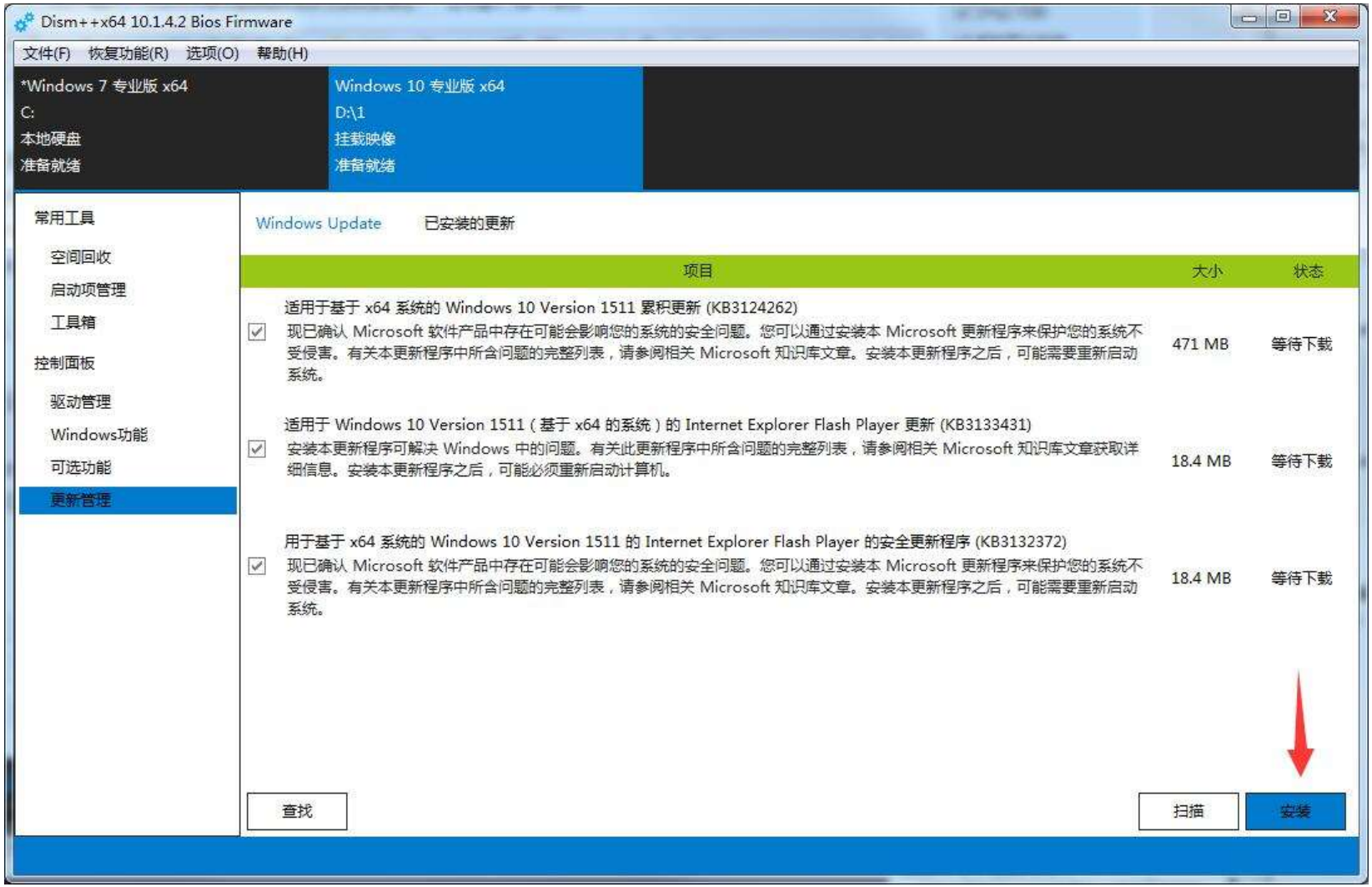
16.2.2. 扫描并安装更新

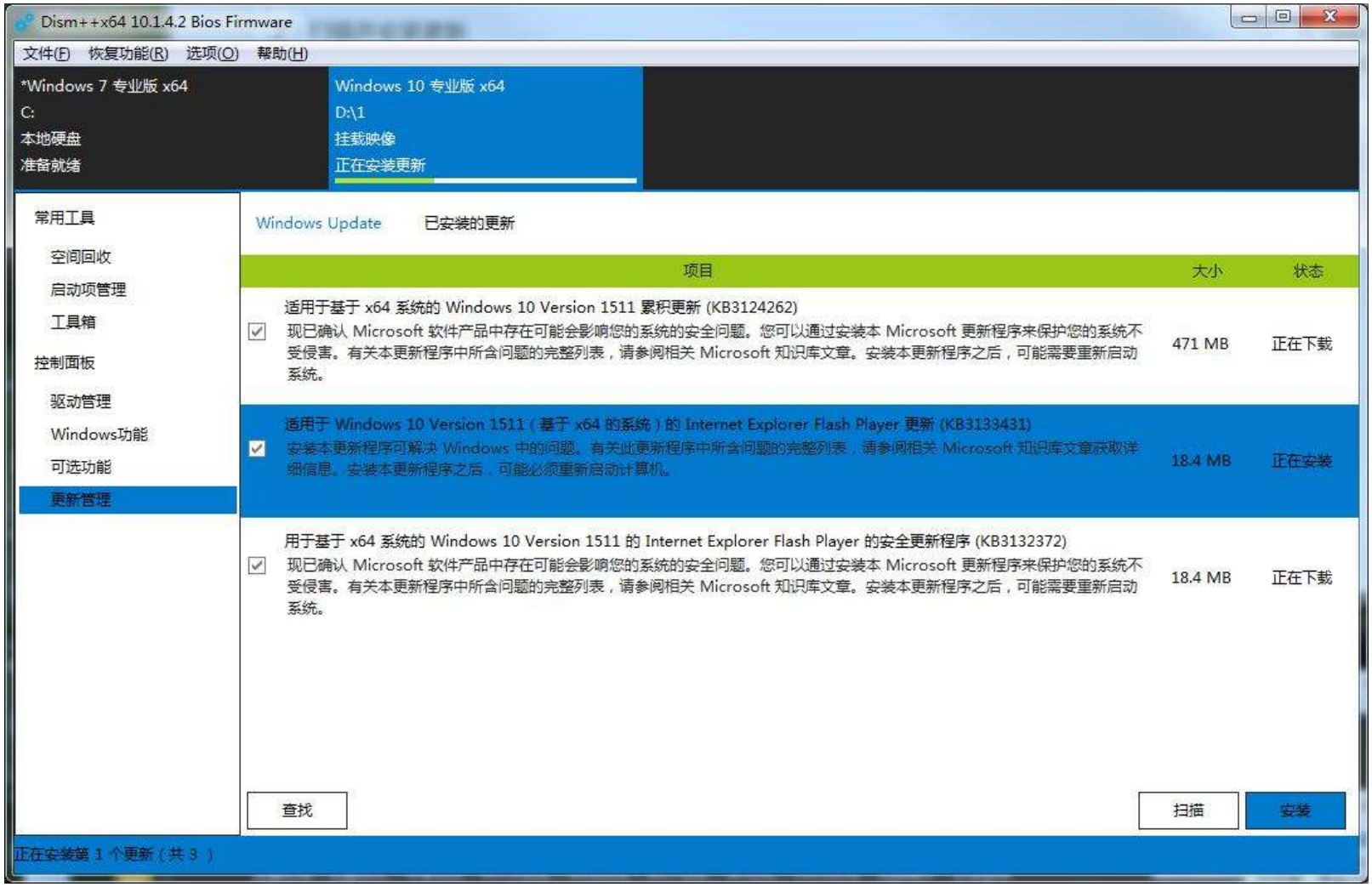
点击打开会话，然后点击 Windows Update，在点击扫描，扫描完成后程序会列出所有可以安装的更新，选中需要的更新点点击安装即可

温馨提示：Dism++会把更新缓存在 程序目录的 "Config\UpdateCache"文件夹，以后安装时 Dism++将直接使用上次的缓存。




Dism++x64 10.1.4.2 Bios Firmware
文件(F) 恢复功能(R) 选项(O) 帮助(H)
*Windows 7 专业版 x64
C:
本地硬盘
准备就绪
Windows 10 专业版 x64
D:\1
挂载映像
正在扫描更新
常用工具
空间回收
启动项管理
工具箱
控制面板
驱动管理
Windows功能
可选功能
更新管理
Windows Update
已安装的更新
项目
大小
状态
查找
扫描
安装


Dism++x64 10.1.4.2 Bios Firmware
文件(F) 恢复功能(R) 选项(O) 帮助(H)
*Windows 7 专业版 x64
C:
本地硬盘
准备就绪
Windows 10 专业版 x64
D:\1
挂载映像
准备就绪
常用工具
空间回收
启动项管理
工具箱
控制面板
驱动管理
Windows功能
可选功能
更新管理
Windows Update
已安装的更新
项目
大小
状态
适用于基于 x64 系统的 Windows 10 Version 1511 累积更新 (KB3124262)
现已确认 Microsoft 软件产品中存在可能会影响您的系统的安全问题。您可以通过安装本 Microsoft 更新程序来保护您的系统不受侵害。有关本更新程序中所含问题的完整列表，请参阅相关 Microsoft 知识库文章。安装本更新程序之后，可能需要重新启动系统。
471 MB
等待下载
适用于 Windows 10 Version 1511 (基于 x64 的系统) 的 Internet Explorer Flash Player 更新 (KB3133431)
安装本更新程序可解决 Windows 中的问题。有关此更新程序中所含问题的完整列表，请参阅相关 Microsoft 知识库文章获取详细信息。安装本更新程序之后，可能必须重新启动计算机。
18.4 MB
等待下载
用于基于 x64 系统的 Windows 10 Version 1511 的 Internet Explorer Flash Player 的安全更新程序 (KB3132372)
现已确认 Microsoft 软件产品中存在可能会影响您的系统的安全问题。您可以通过安装本 Microsoft 更新程序来保护您的系统不受侵害。有关本更新程序中所含问题的完整列表，请参阅相关 Microsoft 知识库文章。安装本更新程序之后，可能需要重新启动系统。
18.4 MB
等待下载
查找
扫描
安装


Dism++x64 10.1.4.2 Bios Firmware
文件(F) 恢复功能(R) 选项(O) 帮助(H)
*Windows 7 专业版 x64
C:
本地硬盘
准备就绪
Windows 10 专业版 x64
D:\1
挂载映像
正在安装更新
常用工具
空间回收
启动项管理
工具箱
控制面板
驱动管理
Windows功能
可选功能
更新管理
Windows Update
已安装的更新
项目
大小
状态
适用于基于 x64 系统的 Windows 10 Version 1511 累积更新 (KB3124262)
现已确认 Microsoft 软件产品中存在可能会影响您的系统的安全问题。您可以通过安装本 Microsoft 更新程序来保护您的系统不受侵害。有关本更新程序中所含问题的完整列表，请参阅相关 Microsoft 知识库文章。安装本更新程序之后，可能需要重新启动系统。
471 MB
正在下载
适用于 Windows 10 Version 1511（基于 x64 的系统）的 Internet Explorer Flash Player 更新 (KB3133431)
安装本更新程序可解决 Windows 中的问题。有关此更新程序中所含问题的完整列表，请参阅相关 Microsoft 知识库文章获取详细信息。安装本更新程序之后，可能必须重新启动计算机。
18.4 MB
正在安装
用于基于 x64 系统的 Windows 10 Version 1511 的 Internet Explorer Flash Player 的安全更新程序 (KB3132372)
现已确认 Microsoft 软件产品中存在可能会影响您的系统的安全问题。您可以通过安装本 Microsoft 更新程序来保护您的系统不受侵害。有关本更新程序中所含问题的完整列表，请参阅相关 Microsoft 知识库文章。安装本更新程序之后，可能需要重新启动系统。
18.4 MB
正在下载
查找
扫描
安装
正在安装第 1 个更新（共 3 ）

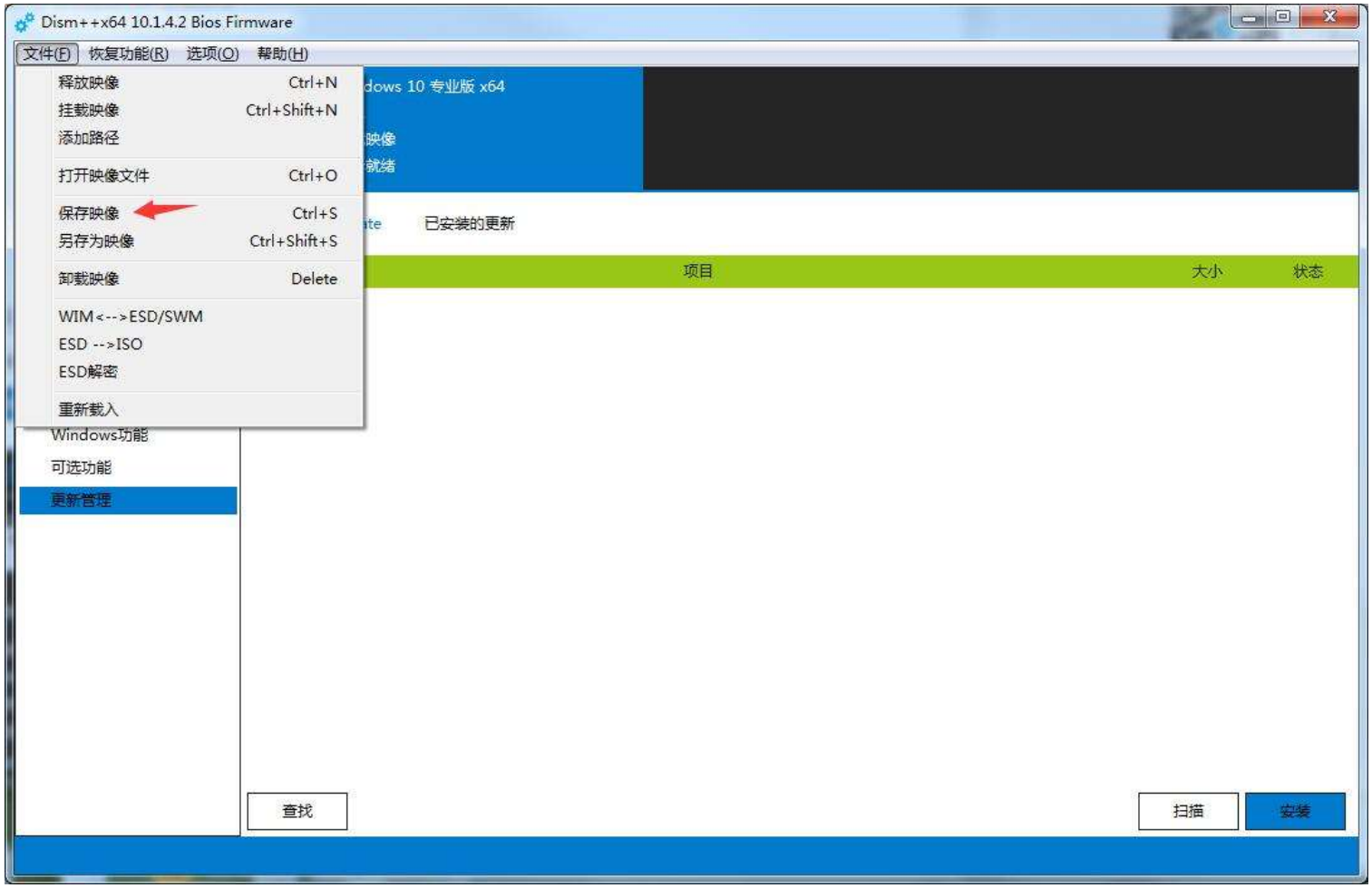
耐心的等待安装完成……温馨提示：直接提示下载失败的 说明 Dism++暂时不支持，目前 Dism++仅支持 cab（msu）那种更新，不支持 exe 等更新。安装完成后，建议再扫描一次更新，因为某些更新有父子依赖关系，可能一次扫描不完整。

如果你不喜欢 Win10 自带的一堆 Appx，你还可以在空间回收中使用过期 Appx 清理，这样就从系统安装包内就删除了 Appx，安装系统后也没有自带 Appx 了。

16.2.3. 保存 install.wim

操作成功后，你可以点击文件选择保存映像或者另存为映像……然后你去替换 iso 中的 install.wim 即可。（UltraISO 可以编辑 ISO~文件~~）

现在你可以使用你的新的 ISO，安装系统，不用在纠结一大把更新了~~~


Dism++x64 10.1.4.2 Bios Firmware
文件(F)
恢复功能(R)
选项(O)
帮助(H)
释放映像 Ctrl+N
挂载映像 Ctrl+Shift+N
添加路径
打开映像文件 Ctrl+O
保存映像 Ctrl+S
另存为映像 Ctrl+Shift+S
卸载映像 Delete
WIM<-->ESD/SWM
ESD -->ISO
ESD解密
重新载入
dows 10 专业版 x64
映像
就绪
已安装的更新
项目
大小
状态
Windows功能
可选功能
更新管理
查找
扫描
安装

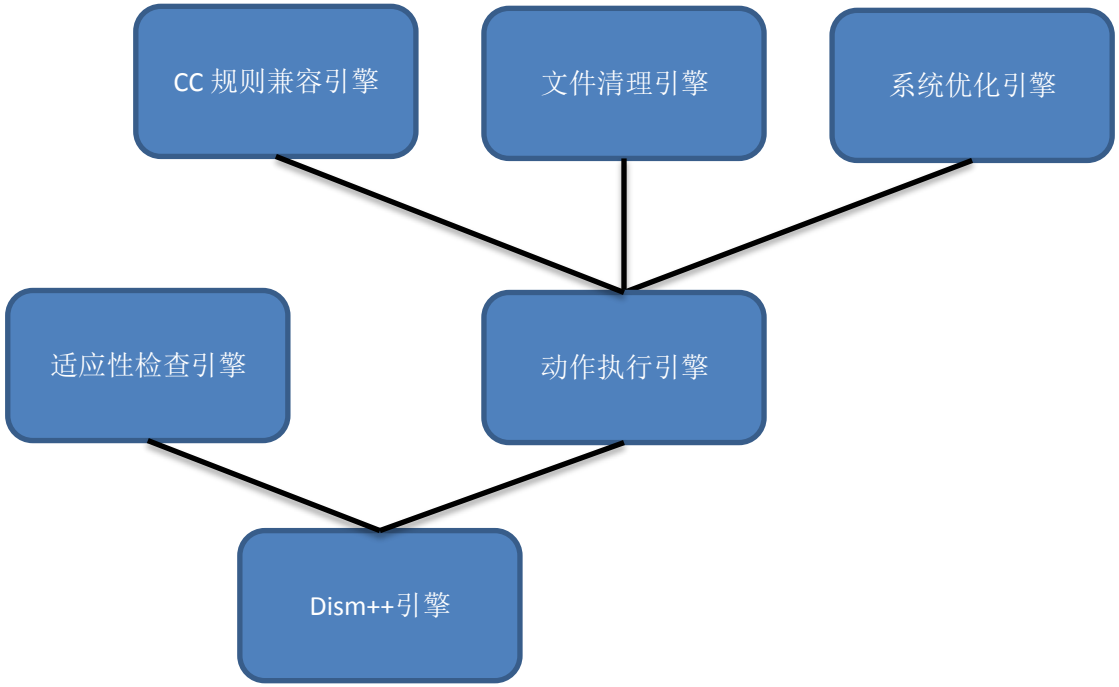
17. Dism++引擎白皮书

如果你想帮助 Dism++整合更加强大的规则，你可以了解本节内容，否则就无视吧。

Dism++为了方便后期添加清理项目，以及系统优化项目，所以在程序发布之处就将规则以及程序分离了。分离也让 Dism++规则变得公开透明，让大家知道程序到底删除什么，是否误杀？规则能否在改进？我也希望能有更多的人能为 Dism++添砖加瓦。为了让大家更快了解引擎机制，我特意编写了该文档。

17.1. Dism++引擎结构

温馨提示：Windows Update 规则由 WUSU 服务器自动产生，因此未公开其细节。



如上图所示，Dism++引擎分为 2 部分：

- 适应性检查引擎主要用于判断某规则是否适合当前环境。比如说你写了一个规则，但是只对 64 位 Windows 8 生效，这时就需要添加过滤规则，而此规则就由适应性检查引擎来提供支持。
- 动作执行引擎则用于清理文件以及系统优化的各种动作执行，你需要删除什么文件就需要告诉它如何扫描以及删除。

17.2. 适应性检查引擎功能列表

函数名称	功能说明	示例
Or	执行子规则进行或逻辑，任意一个规则为 TRUE，则返回 TRUE，所有规则均为 FALSE 才返回 FALSE。	<Or> <FileExist FilePath=”C:\123.txt”/> <FileExist FilePath=”C:\456.txt”/> </Or> C:\123.txt 或者 C:\456.txt 任意一个文件存在就返回 TRUE。
And	执行子规则进行与逻辑，所有规则均为 TRUE 返回 TRUE。	< And > <FileExist FilePath=”C:\123.txt”/> <FileExist FilePath=”C:\456.txt”/> </ And > C:\123.txt 以及 C:\456.txt 同时存在才返回 TRUE。
Not	执行子规则进行逻辑取反。如果 Not 后方存在多条自规则，则将先进行 And。然后再逻辑取反。	<Not> <FileExist FilePath=”C:\123.txt”/> </Not> C:\123.txt 不存在时才返回 TRUE

Arch	比较 Image 的 CPU 体系。此值只可能为以下值： 0 ： x86 CPU 5 ： arm CPU 9 ： amd64 CPU 12：arm64 CPU	<Arch>9</Arch> 如果需要处理的映像是 Windows 7 x64，那么将返回 TRUE，否则返回 FALSE <Arch>5</Arch> 如果需要处理的映像是 Windows 8.1 RT，那么将返回 TRUE，否则返回 FALSE
OSVersion	比较 Image 版本号，其版本号来自 ntdll.dll。 值：一个 1~4 位版本号，比如 6.1.7601.17451 此函数可以添加以下属性： Compare：可选，版本号比较方式，其值可以是==（默认）、>=、<=、>、<、!=	<OSVersion>6.0</OSVersion> 如果你的系统是 Vista，那么此规则将返回 TRUE。没有属性 Compare 则以等于方式比较。 <OSVersion Compare=“>=”>6.1.7601</OSVersion> 如果你的系统是 Win7 SP1、Win8、Win8.1 或者更高，那么返回 TRUE
ImageFlags	对系统状态进行二进制与，如果全部成立，那么返回逻辑真。 其二进制位可以是以下任意组合： 1：联机映像，即 Host 在线模式。 2：本地硬盘，即在 C:\Windows 这样的系统。 4：挂载映像，WIM 文件挂载的系统 8：WimBoot 映像，用 WimBoot 安装的系统 10：VHD 映像，存在于 VHD 的映像 20：Compact 映像，使用了 Compact 压缩 40：WinPE，此映像是个 PE	<ImageFlags>1</ImageFlags> 如果当前映像是联机映像，那么返回 TRUE <ImageFlags>21</ImageFlags> 如果当前映像是联机映像，并且使用了 Compact 压缩，那么返回 TRUE

RegExist	判断注册表是否存在 属性： Key：必选，注册表的路径 Wow64：可选，x86 映像将忽略此标志设置重定向类型，可以是以下值 True、False(默认)、All True 时表示开启重定向，即转向到 Wow6432Node 节点判断注册表。 False 时则不使用重定向，默认此逻辑。 All 时则进行二次判断，首先开启重定向，判断键是否存在，如果不存在，那么在关闭重定向，在判断键是否存在。 Value：可选，值的名称。如果此属性为空，那么仅判断键是否存在。 Type：可选，值的类型 Data：可选，值的数据。除字符串外，其他都是 16 进制表示 Compare：可选，支持==（默认）、!=、>=、<=、>、<、&，此外 REG_BINARY 还支持 N 前缀，表示比较指定字节数。REG_SZ 与 REG_EXPAND_SZ 还能支持 N、I 前缀。N 表示比较指定字符数，I 表示忽略大小写。N、I 支持同时使用，比如 NI==。	< RegExist Key=”HKEY_LOCAL_MACHINE\SOFTWARE\Dism++”/> 如果存在 HKEY_LOCAL_MACHINE\SOFTWARE\Dism++那么返回 TRUE <RegExist Key=”HKEY_LOCAL_MACHINE\SOFTWARE\Dism++” Value=”123”/> 如果存在 HKEY_LOCAL_MACHINE\SOFTWARE\Dism++，并且有 123 这个值那么返回 TRUE <RegExist Key=”HKEY_LOCAL_MACHINE\SOFTWARE\Dism++” Value=”123” Type=“REG_DWORD”/> 如果存在 HKEY_LOCAL_MACHINE\SOFTWARE\Dism++，并且有名字为 123，类型为 REG_DWORD 这个值那么返回 TRUE <RegExist Key=”HKEY_LOCAL_MACHINE\SOFTWARE\Dism++” Value=”123” Type=“REG_DWORD” Data=”FFFFFFFF”/> 如果存在 HKEY_LOCAL_MACHINE\SOFTWARE\Dism++，并且有名字为 123，类型为 REG_DWORD，值类容是 0xFFFFFFFF 这个值那么返回 TRUE。 <RegExist Key=”HKEY_LOCAL_MACHINE\SOFTWARE\Dism++” Value=”123” Type=“REG_DWORD” Data=”1” Compare=”amp;”/> 取出 Value 123，然后进行按位与，如果返回结果为 1，那么返回 TRUE。 <RegExist Key=”HKEY_LOCAL_MACHINE\SOFTWARE\Dism++” Value=”123” Type=“REG_BINARY” Data=”01000000” Compare=”N==”/> 取出 Value 123，如果开头 4 个字节是 0x00000001,那么返回 TRUE。
Flase	返回 Flase	<False/>
True	返回 True	<True/>

FileExist	判断某文件是否存在 属性： FilePath：必选，文件路径，支持环境变量以及扩展函数 Wow64：可选，是否使用重定向可以是以下值，False（默认）、All。X86 系统将忽略此标志。	< FileExist FilePath=” %ProgramFiles%\123.txt”/> %ProgramFiles%\123.txt 存在则返回 TRUE < FileExist FilePath=” %ProgramFiles%\123.txt” Wow64=“All”/> %ProgramFiles%\123.txt 或者%ProgramFilesX86%\123.txt 存在则返回 TRUE
QueryServiceStart	获取服务的状态，支持以下属性： Name：必选，服务的名称 Type：可选，如果为空则仅检查服务是否存在。服务状态，可以是以下值： 0：驱动启动 1：系统启动 2：自动 3：手动 4：禁用 5：自动延迟	<QueryServiceStart Name=”aaa” Type=”4”/> 检测 aaa 服务是否被禁用 <QueryServiceStart Name=”aaa”/> 检测 aaa 服务是否存在
FileVersion	获取文件版本，并执行比较，支持以下属性： FilePath：必选，文件路径 Wow64：可选，是否使用重定向可以是以下值，False（默认）。X86 系统将忽略此标志。 Compare：可选，版本号比较方式，其值可以是==（默认）、>=、<=、>、<、!= 值：一个 1~4 位版本号，比如 6.1.7601.17451	< FileVersion FilePath=”C:\ntdll.dll” Compare=”==”>6.1.7601</ FileVersion> 检测 ntdll 的版本号是否为 6.1.7601

CScript	使用 C 脚本进行适应性检查，具体语法规则，请参考 C 编程语言 C99 规范。
	
	支持以下属性：
	ProcName：执行脚本函数名称
	
	其函数名称必须是以下原形声明，函数名称可以随意
	BOOL WINAPI MyApplicableCScript(DismSession Session,const DismSystem* pSystemInfo);
	
	函数返回 TRUE 表示此规则适应，返回 FALSE 表示规则不适应，返回-1 表示执行出错。
		< CScript ProcName =" MyApplicableCScript"/>
		
		然后在 XML 的 Data/ CScript 中添加函数代码
		<Data>
		<CScript>
		<!—导入外部的 Kernel32.dll 中的 Sleep API-->
		<Imports Module="Kernel32.dll">
		<Import>Sleep</Import>
		<!—Import 可以有多个，可以填写其他 WinAPI，具体请参考 MSDN-->
		</Imports>
		<![CDATA[
		#include <Dism++.h>
		//声明外部 WinAPI，声明后以便于在 C Script 中使用
		void WINAPI Sleep(DWORD dwMilliseconds);
		BOOL WINAPI MyApplicableCScript(DismSession Session,const DismSystem* pSystemInfo)
		{
		//睡眠 500 毫秒
		Sleep(500);
		//调用 Dism++API，更多 API 请参考 Config\include\Dism++.h 文件
		DismWriteLog(DismLogLevelDebug,L"Code",L"这只是一个测试语句");
		return 1;
		}
		]]> </CScript>
		<Data>

17.3. 动作执行引擎引擎功能列表

Dism++规则文件 Activate 将触发的动作。
Dism++目前内置的扩展函数，你可以使用这些函数来进一步增强规则。
GetFileVersion(FilePath) – 根据指定路径，返回文件版本号
GetRegSz(Key, ValueName, bWow64) - 根据注册表路径返回指定字符串
GetIniData(IniPath, KeyPath, ValueName) - 读取指定 inf 节点信息

函数名称	说明	示例
RegWrite	执行注册表写入操作，支持以下属性： Key：必选，注册表路径 Type：可选，值的类型，可以是 REG_SZ、REG_DWORD、REG_QWORD、REG_BINARY、REG_EXPAND_SZ Value：可选，值的名称。 Data：可选，值的数据 Operator：可选，操作类型。仅 REG_DWORD、REG_QWORD、REG_BINARY 可用。支持=（默认）、|、&、^。 Default：可选，默认值。存在 Operator（|、&、^）时必须填写默认值。 Wow64：可选，默认否，指示重定向到 32 位注册表	1：创建注册表 HKEY_LOCAL_MACHINE\SOFTWARE\Test 并这个键中创建一个名字叫 123 的 REG_DOWRD 类型，值为 1 的值。 <RegWrite Key=”HKEY_LOCAL_MACHINE\SOFTWARE\Test” Type=”REG_DWORD” Value=”123” Data=”1”/> 2：创建注册表 HKEY_LOCAL_MACHINE\SOFTWARE\Test 并在子健中创建名字叫 123，类型为 REG_BINARY，值为 0xCC 0x10 <RegWrite Key=”HKEY_LOCAL_MACHINE\SOFTWARE\Test” Type=”REG_BINARY” Value=”123” Data=”CC10”/>

RegDelete	删除注册表，支持以下属性： Key：主键，注册表的路径 Value：可选，值的名称。如果不存在此属性那么删除整个注册表树，如果存在，那么仅删这个值。 Wow64：可选，默认否，指示重定向到 32 位注册表	1：删除整个 HKEY_LOCAL_MACHINE\SOFTWARE\7-Zip < RegDelete Key=" HKEY_LOCAL_MACHINE\SOFTWARE\7-Zip"/> 2：删除 HKEY_LOCAL_MACHINE\SOFTWARE\7-Zip 中的 Path 值 <RegDelete Key="HKEY_LOCAL_MACHINE\SOFTWARE\7-Zip" Value="Path"/>
SetServiceStart	调整服务状态，支持以下属性： Name：必选，服务的名称 Type：必选，服务状态，可以是以下值 0：驱动启动 1：系统启动 2：自动 3：手动 4：禁用 5：自动延迟	将 aaa 服务调整为禁用 <SetServiceStart Name="aaa" Type="4"/>

General	用于删除指定特征的文件，在垃圾清理常用 支持以下属性： RootPath：路径名称，支持增强函数，以及环境变量 Flags：可选，匹配类型 Directory 或者 File。如果此属性为空那么表示 Directory 与 File 同时匹配 支持以下子节点： Query：可选，可重复需要匹配的子路径，如果没有此节点，那么表示匹配所有文件 Excluded：可选，可重复，需要排除的字路径。如果没有此节点，那么表示没有排除列表。	将 C:\123 目录清空（保留 123 这个文件夹） <General RootPath= C:\123"/> 将 C:\123 目录删除（不保留 123 这个文件夹） <General RootPath= C:\"> < Query>123</Query > </General> 将 C:\Temp 里面的所有*.txt 文件删除，但是保留里面的 123.txt 以及 456.txt <General RootPath= C:\ Temp" Flags=”File”> < Query >*.txt</ Query > < Excluded>123.txt</Excluded> < Excluded>456.txt</Excluded> </General>

Custom	自定义清理动作，General 无法满足条件时可以使用 支持以下属性： ProcName：需要调用函数的名称 必选，需要调用的函数名称，其原型必须是 HRESULT (WINAPI*)(DismSession Session, DWORD Flags, UINT64 *CleanUpSpace, DismCallBack CallBack, LPVOID UserData); 参数说明： DismSession Session：Image 会话，可以使用此会话获取 Image 各种信息。 DWORD Flags：保留，Dism++现在不使用此参数，请忽略。 UINT64 *CleanUpSpace：如果 CleanUpSpace 为空，那么函数这执行清理动作。如果不为空，说明仅预估可清理的空间。最后将预估大小用此变量返回。 DismCallBack CallBack：Dism++清理回掉函数，用于展示进度，文件路径等信息。如果此参数为 NULL，则表示没有回调。 回调函数支持以下消息： DISM_MSG_PROGRESS – 用于反馈处理进度，进度信息 wParam 为当前完成百分比，lParam 暂不使用，设置为 0 即可。 DISM_MSG_PROCESS – 用于在状态栏中展示正在处理的文件路径，wParam = (PWSTR) pszFullPath，lParam 设置 0 即可。 DISM_MGS_RemoveInfo – 报告 UI 需要删除的文件，此消息仅扫描时可用，清理时将无视此消息。wParam 设置为 0，lParam=(LPCWSTR)FilePath 表示需要删除的路径。Dism++收到此消息后，会将文件路径展示在详细信息中。 LPVOID UserData：回调函数的 UserData 部分，请务必传入 CallBack 中。 返回值： 如果函数执行成功，请返回 S_OK，其他任何值都表示错误。

ExplorerNotify	通知系统设置更改，支持以下属性： Type：通知类型。只可能是 AssocChanged（刷新文件关联）、Restart（重启 Explorer）、Cmd（执行 Cmd 命令）、Custom（自定义） 如果 Type 是 Cmd，那么还允许以下属性（需要 10.1.5.0 Beta4 或者更高）： Cmd：需要执行的命令 Async：可选, True/False（默认）表示动作是否异步 如果 Type 是 Custom，那么还允许以下属性： msg：消息类型 wParam：WPARAM 参数 lParam：LPARAM 参数 Explorer：可选，True/False（默认），是否仅通知 Explorer	1：比如设置隐藏快捷方式图标后，我们需要重启 Explorer 进程生效。 <ExplorerNotify Type="Restart"/> 2：隐藏 Win10 6 个文件夹后 我们需要刷新 Explorer 配置，可以这样 <ExplorerNotify Type="Custom" msg="111" wParam="#A220" Explorer="true"/> 3：隐藏小盾牌后，刷新文件关联以显示新的图标。 <ExplorerNotify Type="AssocChanged"/> 4：关闭 netprofm 服务 <ExplorerNotify Type="Cmd" Cmd=”net stop netprofm”/> 5：刷新 Explorer 设置 <ExplorerNotify Type="Custom" msg="1A" lParam="TraySettings"/>
RegMove	用于移动注册表信息 Key：源 Key，信息将从这里读取 NewKey：目的 Key，信息将被移动到这里 Value：可选，如果存在此值，那么仅从源移动值信息 NewValue：可选，新的值名称 Wow64：可选，为 True 时重定向到 32 位注册表	1：移动 7-zip 注册表 <RegMove Key=” HKEY_LOCAL_MACHINE\SOFTWARE\7-Zip” NewKey=” HKEY_LOCAL_MACHINE\SOFTWARE\7-Zip2”/> 2：移动 7-zip 注册表的 Path 值为 OldPath <RegMove Key=” HKEY_LOCAL_MACHINE\SOFTWARE\7-Zip” NewKey=” HKEY_LOCAL_MACHINE\SOFTWARE\7-Zip2” Value=”Path” NewValue=“OldPath”/>
FileMove	用于移动文件路径 Src：源文件目录 Dst：目标文件目录	1：将系统盘的 test.xml 移动到 test.bak <FileMove Src="%SystemDrive%\test.xml" Dst="%SystemDrive%\test.bak"/>

CScript	使用 C 脚本进行垃圾清理，具体语法规则，请参考 C 编程语言 C99 规范。
	支持以下属性：
	ProcName：执行脚本函数名称
	
	必选，需要调用的函数名称，其原型必须是
	HRESULT WINAPI MyCleanup(DismSession Session, DWORD Flags, UINT64 *CleanUpSpace, DismCallBack CallBack, LPVOID UserData);
	
	具体函数请参考 Custom 规则。
		< CScript ProcName =" MyCleanup"/>
		
		然后在 XML 的 Data/ CScript 中添加函数代码
		<Data>
		<CScript><![CDATA[
		#include <Dism++.h>
		HRESULT WINAPI MyCleanup(DismSession Session, DWORD Flags, UINT64 *CleanUpSpace, DismCallBack CallBack, LPVOID UserData)
		{
		If(CleanUpSpace)
		{
		//仅检查可用清理项目
		//…
		//检查完成后，返回可以清理的条目，单位 Byte
		* CleanUpSpace=1024;
		}
		else
		{
		//进行垃圾清理 开始清理垃圾
		}
		//返回 0 表示操作成功，其他任意值均表示操作失败
		return 0;
		}
		]]> </CScript><Data>

FileCreateByZIP	从现有的 zip 文件中解压一个文件释放到指定目录 Path：需要释放的目录 ZIPFile：zip 文件目录以及子目录	<FileCreateByZIP Path="%systemroot%\Blank.ico" ZIPFile="Config\default.ui.zip/Data/Blank.ico"/>

17.3.1. General 清理条目示例

示例：删除 Windows 日志文件
分析：由于规则比较简单，单纯删除%SystemDrive%\Windows\里面的一些日志文件以及清空常见日志文件夹。

```
; Level 是推荐操作级别，目前有这几种级别（0：存在风险，1：推荐保留，2：可以删除，3：推荐删除 ）
; Work 是清理时常，某些规则可能清理时间比较长，你可以提高 Work 的数值，如果不存在 Work，那么表示值为 1
; Name 清理条目的名称
<Item Name="清理规则的名称，你可以自己修改成你喜欢的" Level="3" Work=”1”>
        <Discription>清理规则的描述信息。</Discription>
        <Warning>用户选择条目后会提示这条信息。</Warning>
        <Group>临时文件（规则的类别）</Group>
        <Scan>
          <Activate>
            <General RootPath="%SystemDrive%\Windows\" Flags="File">
              <Query>*.log</Query>
              <Query>*.bak</Query>
              <Query>*log.txt</Query>
              <Query>SchedLgU.txt</Query>
              <Query>Performance\WinSAT\*.log</Query>
```

```
            <Query>Panther\*.log</Query>
          </General>
          <General RootPath="%SystemDrive%\Users\*\AppData\" Flags="File">
            <Query>Local\Microsoft\Windows\*.log</Query>
            <Query>Local\MigWiz\*.log</Query>
            <Query>Local\Microsoft\CLR_*\*.log</Query>
            <Query>Local\Microsoft\CLR_*\*\*.log</Query>
            <Query>Roaming\Microsoft\Windows\*.log</Query>
          </General>
          <General RootPath="%SystemDrive%\Logs\"/>
          <General RootPath="%SystemDrive%\Windows\Logs\"/>
          <General RootPath="%SystemDrive%\Windows\Performance\WinSAT\DataStore\"/>
          <General RootPath="*\System Volume Information\Chkdsk"/>
          <General RootPath="%SystemDrive%\ProgramData\USOShared\Logs"/>
          <General RootPath="%SystemDrive%\Windows\System32\WDI"/>
          <General RootPath="%SystemDrive%\Windows\System32\LogFiles\*"/>
          <General RootPath="%SystemDrive%\Windows\debug"/>
        </Activate>
      </Scan>
</Item>
```

17.3.2. CheckBox 优化条目示例

示例：创建快捷方式时，不添加 快捷方式 文字

分析：因为这个功能就二种状态，启用或者关闭，所以我们选择了单选框（CheckBox）

Dism++在启动时需要暂时功能当前状态，因此，写一条状态检测代码

```
<State>
    <Applicable>
        <RegExist Key="HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer" Value="Link" Type="REG_BINARY" Data="00000000"/>
    </Applicable>
</State>
```

执行结果为 True 则表示该功能已经启用，否则就是未启用。

用户在调整状态时，如果用户需要启用此功能，那么我们在 True 节点添加对应的动作，这样启用时，Dism++用你添加的规则执行对应的动作。

```
<True>
    <Activate>
        <!--写入注册表信息-->
        <RegWrite Key="HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer" Value="Link" Type="REG_BINARY" Data="00000000"/>
        <!—重启 Explorer-->
        <ExplorerNotify Type="Restart"/>
    </Activate>
</True>
```

另外就是关闭此功能时，我们写规则把他复原（删除添加的注册表，并重启 Explorer）：

```
<False>
	<Activate>
		<RegDelete Key="HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer" Value="Link"/>
		<ExplorerNotify Type="Restart"/>
	</Activate>
</False>
```

OK，最后成品就是这样的，可以看看以下具体规则

```
	<Item Type="CheckBox" Name="创建快捷方式时不添&amp;quot;快捷方式&amp;quot;文字（此条目由 518516.net 提供）">
		<Current>
			<State>
				<Applicable>
<RegExist Key="HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer" Value="Link" Type="REG_BINARY" Data="00000000"/>
				</Applicable>
			</State>
			<True>
				<Activate>
<RegWrite Key="HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer" Value="Link" Type="REG_BINARY" Data="00000000"/>
					<ExplorerNotify Type="Restart"/>
				</Activate>
			</True>
			<False>
```

```
                <Activate>
                  <RegDelete Key="HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer" Value="Link"/>
                  <ExplorerNotify Type="Restart"/>
                </Activate>
              </False>
            </Current>
            <System>
              <State>
                <Applicable>
<RegExist Key="HKEY_USERS\DEFAULT\Software\Microsoft\Windows\CurrentVersion\Explorer" Value="Link" Type="REG_BINARY" Data="00000000"/>
                </Applicable>
              </State>
              <True>
                <Activate>
<RegWrite Key="HKEY_USERS\DEFAULT\Software\Microsoft\Windows\CurrentVersion\Explorer" Value="Link" Type="REG_BINARY" Data="00000000"/>
                </Activate>
              </True>
              <False>
                <Activate>
                  <RegDelete Key="HKEY_USERS\DEFAULT\Software\Microsoft\Windows\CurrentVersion\Explorer" Value="Link"/>
                </Activate>
              </False>
            </System>
          </Item>
```

17.3.3. Combo 优化条目示例

示例：调整 Windows 10 任务栏中的 Cortana 显示状态。
由于显示状态有多个（隐藏、仅显示图标、显示搜索框），所以我们用下拉框（Combo）来完成。此规则仅支持 Win10 以上系统，因此先写一条适用性检查规则，在大于等于 10.0 的电脑中才显示此规则：

```
<Item Type="Combo" Name="将任务栏中的 Cortana 调整为">
    <Applicable><OSVersion Compare=">=">10.0</OSVersion></Applicable>
</Item>
```

另外，程序需要知道当前是哪一个状态，所以每一个条目都需要些一个 Applicable，让 Dism++知道，当前是哪一个条目。先说隐藏吧，经过分析 SearchboxTaskbarMode 为时，表示隐藏，所以有如下代码：

```
<Dropdown Name="#隐藏">
      <Applicable>
<RegExist Key="HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Search" Value="SearchboxTaskbarMode" Type="REG_DWORD" Data="0"/>
      </Applicable>
      <Activate>
            <RegWrite  Key="HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Search"  Value="SearchboxTaskbarMode"  Type="REG_DWORD"
Data="0"/>
            <ExplorerNotify Type="Custom" msg="1A" lParam="TraySettings"/>
    </Activate>
</Dropdown>
```

Dism++会执行 Applicable 节点，如果结果为 True，那么表示当前状态是映像，另外 Activate 是执行动作。如果用户需要把状态调整为隐藏，那么 Dism++会执行隐藏下面的 Activate。

以此类推，最后得出了以下规则：

```
        <Item Type="Combo" Name="#将任务栏中的 Cortana 调整为">
          <Applicable>
            <OSVersion Compare=">=">10.0</OSVersion>
          </Applicable>
          <Current>
            <Dropdown Name="#隐藏">
              <Applicable>
                <RegExist Key="HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Search" Value="SearchboxTaskbarMode"
Type="REG_DWORD" Data="0"/>
              </Applicable>
              <Activate>
                <RegWrite Key="HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Search" Value="SearchboxTaskbarMode"
Type="REG_DWORD" Data="0"/>
                <ExplorerNotify Type="Custom" msg="1A" lParam="TraySettings"/>
              </Activate>
            </Dropdown>
            <Dropdown Name="#仅显示图标">
              <Applicable>
                <RegExist Key="HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Search" Value="SearchboxTaskbarMode"
Type="REG_DWORD" Data="1"/>
              </Applicable>
              <Activate>
                <RegWrite Key="HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Search" Value="SearchboxTaskbarMode"
Type="REG_DWORD" Data="1"/>
                <ExplorerNotify Type="Custom" msg="1A" lParam="TraySettings"/>
              </Activate>
```

```
            </Dropdown>
            <Dropdown Name="#显示搜索框">
              <Applicable>
                <Or>
                  <Not>
                    <RegExist Key="HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Search" Value="SearchboxTaskbarMode"/>
                  </Not>
                  <RegExist          Key="HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Search"          Value="SearchboxTaskbarMode"
Type="REG_DWORD" Data="2"/>
                </Or>
              </Applicable>
              <Activate>
                <RegWrite          Key="HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Search"          Value="SearchboxTaskbarMode"
Type="REG_DWORD" Data="2"/>
                <ExplorerNotify Type="Custom" msg="1A" lParam="TraySettings"/>
              </Activate>
            </Dropdown>
          </Current>

          <System>
            <Dropdown Name="#隐藏">
              <Applicable>
                <RegExist          Key="HKEY_USERS\DEFAULT\SOFTWARE\Microsoft\Windows\CurrentVersion\Search"          Value="SearchboxTaskbarMode"
Type="REG_DWORD" Data="0"/>
              </Applicable>
              <Activate>
```

```
                <RegWrite Key="HKEY_USERS\DEFAULT\SOFTWARE\Microsoft\Windows\CurrentVersion\Search" Value="SearchboxTaskbarMode"
Type="REG_DWORD" Data="0"/>
              </Activate>
            </Dropdown>
            <Dropdown Name="#仅显示图标">
              <Applicable>
                <RegExist Key="HKEY_USERS\DEFAULT\SOFTWARE\Microsoft\Windows\CurrentVersion\Search" Value="SearchboxTaskbarMode"
Type="REG_DWORD" Data="1"/>
              </Applicable>
              <Activate>
                <RegWrite Key="HKEY_USERS\DEFAULT\SOFTWARE\Microsoft\Windows\CurrentVersion\Search" Value="SearchboxTaskbarMode"
Type="REG_DWORD" Data="1"/>
              </Activate>
            </Dropdown>
            <Dropdown Name="#显示搜索框">
              <Applicable>
                <Or>
                  <Not>
                    <RegExist Key="HKEY_USERS\DEFAULT\SOFTWARE\Microsoft\Windows\CurrentVersion\Search" Value="SearchboxTaskbarMode"/>
                  </Not>
                  <RegExist Key="HKEY_USERS\DEFAULT\SOFTWARE\Microsoft\Windows\CurrentVersion\Search" Value="SearchboxTaskbarMode"
Type="REG_DWORD" Data="2"/>
                </Or>
              </Applicable>
              <Activate>
                <RegWrite Key="HKEY_USERS\DEFAULT\SOFTWARE\Microsoft\Windows\CurrentVersion\Search" Value="SearchboxTaskbarMode"
```

```
Type="REG_DWORD" Data="2"/>
                    </Activate>
                  </Dropdown>
               </System>
            </Item>
```