# 应用程序软件更新器

## 使用手册

版本：1.0

编写日期：2025年5月6日

**内部文档，请勿外传**

# 第1章：引言

欢迎使用应用程序软件更新器（以下简称“更新器”）。本手册旨在为您提供全面、详细的使用指南，帮助您充分理解和有效利用本更新器的各项功能，确保您的应用程序能够顺利、安全地完成更新过程。

## 2.1 手册目的与范围

本手册的主要目的是详细阐述应用程序软件更新器的使用方法、配置选项、工作流程以及相关的技术细节。通过阅读本手册，您将能够：

* 理解更新器的基本工作原理和设计理念。
* 掌握通过命令行参数精确控制更新过程的各种方法。
* 了解不同更新场景下的最佳实践和配置建议。
* 熟悉更新器的用户界面（如果启用）及其交互方式。
* 学会查看和分析更新日志，以便进行故障排查和问题诊断。
* 了解如何将更新器集成到您的软件分发和部署流程中。

本手册的范围涵盖了更新器的所有面向用户的功能和配置选项，包括但不限于：

* 更新器的启动与参数传递。
* 旧版本应用程序进程的管理（等待退出、强制终止）。
* 更新前的文件备份策略（是否备份、是否压缩）。
* 新旧文件的复制与替换逻辑。
* 快捷方式的创建与更新。
* 更新过程中的用户界面展示（标准窗口、动画窗口、隐藏模式）。
* 更新日志的生成、记录与上传。
* 错误处理与提示机制。
* 特定高级功能，如敏感DLL处理、服务器记录集成等。

需要注意的是，本手册主要关注更新器的**使用**层面。它不涉及更新器本身的源代码级开发、编译或修改。对于需要进行二次开发或深入了解内部实现细节的用户，建议参考相关的开发文档或直接阅读源代码。

## 2.2 目标读者

本手册主要面向以下类型的读者：

* **软件部署工程师与系统管理员：** 负责将应用程序部署到最终用户环境，并需要确保更新过程自动化、可靠、高效的技术人员。他们需要深入了解更新器的所有参数和功能，以便根据具体部署场景进行定制化配置。
* **应用程序开发者：** 需要将更新功能集成到自己开发的应用程序中的开发人员。他们需要了解如何调用更新器、传递哪些参数以及更新器的工作流程，以便实现无缝的更新体验。
* **技术支持工程师：** 需要帮助最终用户解决更新过程中遇到的问题的支持人员。他们可以通过本手册了解更新器的行为、日志格式和常见错误，从而更快地定位和解决问题。
* **对更新器工作机制感兴趣的高级用户：** 希望了解软件更新背后技术细节的用户。

我们假定读者具备基本的计算机操作知识和对Windows操作系统环境的一定了解。对于涉及命令行操作的部分，熟悉命令行工具的使用将有助于更好地理解和应用相关内容。

## 2.3 更新器概述

应用程序软件更新器是一款功能强大且灵活的工具，专门设计用于自动化Windows平台下应用程序的更新过程。它的核心目标是简化软件版本的升级部署，减少手动干预，提高更新的成功率和用户体验。

更新器通常作为一个独立的 可执行文件（.exe）提供。当需要更新某个应用程序时，主应用程序或部署脚本会调用此更新器，并通过命令行参数传递必要的指令和信息，例如新版本文件的存放路径、旧版本应用程序的安装路径、需要执行的操作（如是否备份、是否创建快捷方式）等。

更新器接收到指令后，会按照预设的逻辑执行一系列操作，主要包括：

1. **解析命令行参数：** 理解用户的更新意图和配置。
2. **处理旧版本进程：** 根据配置，等待或强制结束正在运行的旧版本应用程序进程，以解除文件占用。
3. **备份旧文件（可选）：** 将目标路径下的旧文件备份到指定位置，以备更新失败时回滚。
4. **复制新文件：** 将源路径下的新版本文件复制到目标路径，覆盖旧文件。
5. **执行额外操作（可选）：** 如创建或更新桌面/开始菜单快捷方式。
6. **清理临时文件（可选）：** 删除更新过程中产生的临时文件或下载的更新包。
7. **启动新版本（可选）：** 在更新完成后自动启动新版本的应用程序。
8. **记录日志与报告：** 详细记录更新过程的每一步操作及结果，并将更新状态（成功/失败）及日志（可选）上传到指定服务器。

更新器支持多种运行模式，包括带标准界面的交互模式、带动画效果的视觉模式以及完全隐藏的静默模式，以适应不同的应用场景和用户体验需求。

## 2.4 主要特性

本更新器具备一系列精心设计的特性，以满足复杂多变的更新需求：

* **强大的命令行控制：** 提供丰富的命令行参数，允许用户精确控制更新过程的每一个环节，如指定源/目标路径、管理旧进程、配置备份策略、控制UI显示、设置服务器报告等。
* **灵活的旧进程管理：** 支持通过进程ID或进程名定位旧版本程序，并可配置等待其自然退出（带超时）或强制终止。
* **可靠的文件备份与恢复：** 可选的文件备份功能，支持将旧版本文件备份，并提供压缩备份选项以节省空间。虽然代码中回滚逻辑（rollback\_file）存在但可能未完全启用或文档化，备份为手动恢复提供了可能。
* **智能的文件占用处理：** 在复制文件前检查目标文件是否被占用，并尝试结束占用进程（基于旧版本进程信息）。
* **快捷方式管理：** 支持自动创建或递归替换应用程序的快捷方式。
* **多种用户界面模式：**
  + **标准窗口模式：** 显示详细的更新日志和状态信息，适合需要监控过程的场景。
  + **动画窗口模式：** 提供一个简洁的加载动画和进度百分比，提升视觉体验。
  + **隐藏窗口模式：** 在后台静默执行更新，对最终用户完全透明。
* **详细的日志记录：** 生成详细的操作日志文件，记录时间戳、操作步骤、成功与否及错误信息，便于追踪和排查问题。
* **更新结果服务器报告：** 可将更新结果（成功/失败）及详细日志（可选）上传到指定的HTTP服务器，方便集中监控和分析更新部署情况。
* **错误提示与处理：** 在发生错误时，可配置是否弹窗提示用户（尤其在隐藏窗口模式下），并记录详细错误信息到日志。
* **敏感DLL处理：** 允许指定一系列“敏感”的DLL文件，可能用于绕过某些安全软件的拦截（具体机制需谨慎评估）。
* **辅助功能：** 提供命令行帮助（-help, -option）、查看默认敏感DLL列表（-default\_sensitive\_dll）、删除所有备份文件（-delete\_all\_backup\_file）等辅助命令。
* **可配置性：** 窗口标题、尺寸、置顶行为等均可通过参数进行配置。
* **独立性：** 更新器是一个独立的执行文件，易于分发和集成。

## 2.5 关于本文档

为了帮助您快速有效地掌握更新器的使用，本手册采用了清晰的结构组织：

* **章节划分：** 手册按照逻辑功能划分为多个章节，从基础入门到高级应用，循序渐进。
* **目录：** 文档开头提供了详细的目录，方便您快速跳转到感兴趣的部分。
* **命令行参数：** 对所有命令行参数进行了详细解释，并提供了使用示例。
* **流程说明：** 对核心的更新流程进行了分步说明。
* **界面截图（如有）：** 在介绍用户界面时，会辅以截图说明（本手册为文本描述，实际Word版可考虑加入）。
* **附录：** 包含更新日志、参数速查表、术语表等实用参考信息。

**排版约定：**

* 命令行参数 或 文件名 会使用等宽字体表示。
* **重要提示** 或 **注意事项** 会使用加粗或其他方式突出显示。
* 代码片段或日志示例会使用特定的格式块展示。

我们建议您按照章节顺序阅读本手册，特别是前几章的基础内容。对于有特定需求的用户，可以通过目录直接查阅相关章节。如果您在使用过程中遇到任何问题，可以查阅“日志与故障排查”章节或“常见问题解答”附录。

我们致力于不断改进更新器和本文档。如果您有任何建议或发现任何错误，欢迎随时与我们联系。

# 第2章：开始使用

本章将引导您开始使用应用程序软件更新器。我们将介绍运行更新器所需的基本环境要求，说明如何获取更新器程序，解释一些核心概念，并通过一个简单的示例让您快速了解其基本用法。

## 3.1 系统要求

应用程序软件更新器主要设计用于 **Microsoft Windows** 操作系统环境。为了确保更新器能够正常运行并执行其所有功能，请确保目标系统满足以下基本要求：

* **操作系统：**
  + Windows 7 或更高版本。请注意，根据更新日志（版本 1.0.0.2），对于需要通过 HTTPS 上传更新记录到服务器的功能，Windows 7 系统可能需要确保已启用或支持 TLS 1.1/1.2 加密协议，否则可能导致上传失败。建议使用更新版本的 Windows 以获得最佳兼容性。
  + 支持 32 位 (x86) 和 64 位 (x64) 架构，具体取决于更新器本身的编译版本以及它需要更新的目标应用程序。
* **磁盘空间：**
  + 更新器本身 (imp\_updater.exe) 体积很小，通常只需要几 MB 的空间。
  + 需要足够的磁盘空间来存放下载的更新文件（源路径 source\_path）。
  + 目标应用程序的安装路径（目标路径 destination\_path）需要有足够的空间来容纳更新后的文件。
  + 如果启用了备份功能 (-is\_backup\_file true)，则需要额外的空间来存储备份文件。备份文件的存储位置通常在用户的 LOCALAPPDATA 目录下的 imp\_updater\_backup 子目录中（例如 C:\Users\<用户名>\AppData\Local\imp\_updater\_backup）。如果启用了压缩备份 (-is\_compress\_backup\_file true)，所需的备份空间会相对减少。
* **内存 (RAM)：** 更新器运行时对内存的需求不高，通常几十 MB 即可满足。但在处理大型应用程序或大量文件的更新时，系统总内存建议至少满足 Windows 的基本运行要求。
* **网络连接（可选）：** 如果需要使用更新记录上传功能 (-upload\_update\_record\_flag 大于 0），目标机器需要能够访问指定的服务器地址 (-server\_address) 和端口 (-server\_port)。通常需要稳定的互联网连接。
* **权限：**
  + 更新器通常需要具有足够的权限来读取源路径、写入目标路径、终止旧版本进程、创建快捷方式以及写入日志文件（通常在更新器所在目录的 install\_log 子目录）。
  + 在大多数情况下，如果目标应用程序安装在 C:\Program Files 或 C:\Program Files (x86) 等受保护的目录下，更新器可能需要以**管理员权限**运行才能成功完成更新。建议在调用更新器时考虑权限提升。

## 3.2 获取更新器

应用程序软件更新器 (imp\_updater.exe) 通常不作为独立软件公开发布，而是作为特定应用程序更新解决方案的一部分提供。您可以通过以下途径获取更新器：

* **随应用程序分发：** 更新器可能包含在您需要更新的应用程序的安装包或更新包中。
* **从开发者处获取：** 如果您是应用程序开发者或部署工程师，可以直接从负责更新器开发的团队或源头获取最新的可执行文件及其相关文档。

请确保您获取的是针对目标平台（32位/64位）和预期功能的正确版本的更新器。

获取到 imp\_updater.exe 文件后，您可以将其放置在任何方便调用的位置。常见的做法是：

* 将其放置在主应用程序的安装目录下。
* 将其放置在一个专门用于部署工具的目录中。

无论放置在哪里，调用它时都需要确保能够正确指定其路径，或者将其路径添加到系统的 PATH 环境变量中。

## 3.3 基本概念

在深入了解更新器的具体参数和流程之前，理解以下几个核心概念非常重要：

* **更新器 (Updater)：** 指 imp\_updater.exe 这个可执行文件本身。它是执行更新操作的主体。
* **源路径 (Source Path)：** 命令行参数 -source\_path 指定的路径。这个路径包含了新版本应用程序的所有文件和文件夹。更新器会从这里复制文件到目标路径。
* **目标路径 (Destination Path)：** 命令行参数 -destination\_path 指定的路径。这是旧版本应用程序当前安装的位置，也是新文件将被复制到的地方。
* **旧版本进程 (Old Version Process)：** 指正在运行的、需要被更新的应用程序实例。更新器需要确保在复制文件之前，这些进程已经退出，以避免文件占用冲突。可以通过进程ID (-old\_version\_process\_id) 或进程名 (-old\_version\_process\_name) 来指定需要处理的旧进程。
* **命令行参数 (Command Line Arguments)：** 更新器通过命令行接收指令。这些参数以 - 开头，后面跟着参数名，有时还跟着参数值（例如 -source\_path C:\Updates\MyApp\_v2）。通过组合不同的参数，可以精确控制更新行为。
* **备份 (Backup)：** 一个可选过程，用于在覆盖旧文件之前将其保存起来。通过 -is\_backup\_file true 启用。备份文件通常存储在用户的本地应用数据目录中，并以进程名和版本号命名，以便区分。
* **快捷方式 (Shortcut)：** 指向应用程序可执行文件的桌面图标或开始菜单项。更新器可以根据配置 (-is\_create\_shortcut) 创建或更新快捷方式，使其指向新版本的应用程序。
* **日志 (Log)：** 更新器在执行过程中会记录详细的操作信息，包括时间戳、执行的步骤、遇到的错误等。日志文件通常保存在更新器所在目录下的 install\_log 子目录中，以日期时间命名（如 20250506103000.log）。日志对于诊断更新失败的原因至关重要。
* **用户界面 (UI)：** 更新器可以以不同的界面模式运行。标准窗口 (-window\_style 0) 显示详细日志；动画窗口 (-window\_style 1) 显示进度动画；隐藏模式 (-is\_hide\_window true) 则在后台运行，用户不可见。
* **更新记录上传 (Update Record Upload)：** 一个可选功能，允许更新器将更新结果（成功/失败）和日志信息发送到指定的服务器 (-server\_address, -server\_port)，用于集中监控。通过 -upload\_update\_record\_flag 控制上传行为。

## 3.4 快速启动示例

假设您需要更新一个名为 MyApp.exe 的应用程序，它的旧版本安装在 D:\Program Files\MyApp 目录下。新版本的文件已经准备好，存放在 C:\MyAppUpdates\NewVersion 目录下。您希望更新器执行以下操作：

1. 找到并结束名为 MyApp.exe 的旧进程。
2. 在更新前备份旧文件。
3. 将 C:\MyAppUpdates\NewVersion 下的所有文件复制到 D:\Program Files\MyApp。
4. 更新完成后，删除 C:\MyAppUpdates\NewVersion 目录下的更新文件。
5. 为 D:\Program Files\MyApp\MyApp.exe 创建或更新桌面快捷方式。
6. 显示一个带动画的更新进度窗口。
7. 记录详细日志，但不上传到服务器。

您可以通过以下命令行来调用更新器 (imp\_updater.exe)：

imp\_updater.exe ^  
 -old\_version\_process\_name MyApp.exe ^  
 -source\_path C:\MyAppUpdates\NewVersion ^  
 -destination\_path "D:\Program Files\MyApp" ^  
 -is\_backup\_file true ^  
 -is\_delete\_update\_file true ^  
 -is\_create\_shortcut true ^  
 -window\_style 1 ^  
 -upload\_update\_record\_flag 0

**命令解释：**

* imp\_updater.exe: 调用更新器程序。
* ^: Windows 命令行中的换行符，使长命令更易读（在实际单行执行时不需要）。
* -old\_version\_process\_name MyApp.exe: 指定需要处理的旧进程名为 MyApp.exe。
* -source\_path C:\MyAppUpdates\NewVersion: 指定新版本文件所在的源路径。
* -destination\_path "D:\Program Files\MyApp": 指定旧版本安装的目标路径。注意路径中包含空格，因此用双引号括起来。
* -is\_backup\_file true: 启用文件备份。
* -is\_delete\_update\_file true: 更新完成后删除源路径下的更新文件。
* -is\_create\_shortcut true: 创建或更新快捷方式。
* -window\_style 1: 使用动画窗口模式显示更新进度。
* -upload\_update\_record\_flag 0: 不上传更新记录到服务器。

执行这条命令后，更新器将启动，显示动画窗口，并按照指定的参数执行更新流程。更新完成后，窗口会自动关闭（或根据配置停留）。您可以在 install\_log 目录中找到本次更新的详细日志文件。

**注意：** 如果 D:\Program Files\MyApp 是受保护目录，您可能需要以管理员权限运行此命令。

这个简单的示例展示了更新器的基本用法。通过组合不同的命令行参数，您可以实现更复杂的更新逻辑。接下来的章节将详细介绍每个参数和功能。

# 第3章：安装与部署

本章主要介绍如何将应用程序软件更新器部署到您的环境中，并将其集成到现有的应用程序更新流程中。我们将讨论更新器相关的文件结构、运行所需的权限，以及将其嵌入应用程序更新机制的常见方法。

## 4.1 文件结构

应用程序软件更新器 (imp\_updater.exe) 本身是一个单一的可执行文件。然而，在实际部署和使用过程中，通常会涉及到以下相关的文件和目录结构：

1. **更新器可执行文件 (imp\_updater.exe)：**
   * 这是核心的更新工具。
   * **部署位置：** 可以放置在主应用程序的安装目录下，也可以放在一个独立的工具目录中。选择哪个位置取决于您的部署策略和调用方式。
     + 放在主程序目录：方便相对路径调用，但会随主程序一起安装/卸载。
     + 放在独立工具目录：便于统一管理更新工具，但调用时可能需要指定完整路径或配置环境变量。
2. **更新包（源路径 source\_path）：**
   * 这是一个包含新版本应用程序所有文件和子目录的文件夹。
   * **来源：** 通常由构建系统生成，或者从服务器下载。
   * **内容：** 应包含与目标应用程序安装目录结构相对应的新版本文件。
   * **部署：** 在执行更新前，需要确保这个源路径存在并且更新器有权访问。更新完成后，根据 -is\_delete\_update\_file 参数的设置，此目录下的文件可能会被删除。
3. **目标应用程序（目标路径 destination\_path）：**
   * 这是需要被更新的应用程序的实际安装目录。
   * 更新器需要对这个目录及其子目录和文件具有读、写、删除的权限。
4. **日志目录 (install\_log)：**
   * **位置：** 默认情况下，更新器会在其**自身所在的目录**下创建一个名为 install\_log 的子目录。
   * **内容：** 存放每次更新操作生成的详细日志文件（.log）。文件名通常基于更新开始时的日期和时间（例如 YYYYMMDDHHMMSS.log）。
   * **权限：** 更新器需要有权限在此目录下创建子目录和写入文件。
   * **注意：** 如果更新器以隐藏模式 (-is\_hide\_window true) 或仅启动进程模式 (-is\_only\_start\_process true) 运行，日志记录行为可能会有所不同或受限（代码显示 is\_only\_start\_process 时不打开日志文件，但写入逻辑仍存在，需实测确认）。
5. **备份目录 (imp\_updater\_backup)：**
   * **位置：** 当启用备份 (-is\_backup\_file true) 时，备份文件默认存储在当前用户的 LOCALAPPDATA 环境变量指向的路径下的 imp\_updater\_backup 子目录中。例如：C:\Users\<用户名>\AppData\Local\imp\_updater\_backup。
   * **结构：** 在 imp\_updater\_backup 目录下，会根据 -old\_version\_process\_name 和 -latest\_version 参数创建子目录来存放特定应用程序特定版本的备份。例如：C:\Users\<用户名>\AppData\Local\imp\_updater\_backup\MyApp.exe\1.0.0.0\_backup。
   * **内容：** 包含从目标路径备份出来的旧版本文件。如果启用了压缩备份 (-is\_compress\_backup\_file true)，这里可能存放的是一个压缩包（例如 .zip 文件），否则是原始文件结构。
   * **权限：** 更新器需要有权限在 LOCALAPPDATA 目录下创建子目录和写入文件。
   * **管理：** 可以使用 -delete\_old\_version\_backup\_file true 参数让更新器在成功更新后尝试删除旧的备份。也可以使用辅助命令 -delete\_all\_backup\_file 手动清空整个备份目录。

**典型部署结构示例：**

假设主应用程序 MyApp.exe 安装在 C:\Program Files\MyApp，更新器也放在该目录下，更新包下载到 C:\MyAppUpdates\NewVersion。

C:\Program Files\MyApp\  
 MyApp.exe  
 imp\_updater.exe <-- 更新器  
 install\_log\ <-- 日志目录 (由更新器创建)  
 20250506110000.log  
 ...  
 (其他应用程序文件...)  
  
C:\MyAppUpdates\  
 NewVersion\ <-- 源路径 (更新包)  
 MyApp.exe  
 (新版本文件...)  
  
C:\Users\<用户名>\AppData\Local\imp\_updater\_backup\  
 MyApp.exe\ <-- 备份根目录 (由更新器创建)  
 1.0.0.0\_backup\ <-- 特定版本备份 (假设latest\_version是1.0.0.0)  
 MyApp.exe  
 (旧版本文件备份...)

## 4.2 权限要求

权限是确保更新器成功运行的关键因素。由于更新器需要执行文件操作、进程管理等敏感任务，通常需要比普通应用程序更高的权限。

* **目标路径写入权限：** 这是最基本也是最重要的权限。更新器必须能够读取、写入和删除目标路径 (-destination\_path) 下的文件和文件夹。如果目标路径位于系统保护区域（如 C:\Program Files, C:\Windows），则几乎**必须以管理员权限运行**更新器。
* **源路径读取权限：** 更新器需要能够读取源路径 (-source\_path) 下的所有文件和文件夹。
* **进程终止权限：** 更新器需要有权限终止由 -old\_version\_process\_id 或 -old\_version\_process\_name 指定的旧版本进程。如果旧进程是以管理员权限运行的，那么更新器也必须以管理员权限运行才能终止它。
* **日志写入权限：** 需要在更新器所在目录（或其子目录 install\_log）创建和写入日志文件的权限。
* **备份写入权限：** 需要在用户的 LOCALAPPDATA 目录下创建和写入备份文件的权限。通常用户对其自己的 AppData 目录有完全控制权，但这仍可能受组策略等限制。
* **快捷方式创建/修改权限：** 创建或修改桌面、开始菜单等位置的快捷方式通常需要管理员权限，尤其是当快捷方式指向受保护位置的程序时。
* **注册表访问权限（如果涉及）：** 虽然当前代码似乎不直接操作注册表进行更新，但如果未来版本或特定场景涉及，也需要相应权限。

**如何确保权限：**

1. **请求管理员权限运行：** 这是最常见且可靠的方法。可以通过以下方式实现：
   * **手动运行：** 右键点击 imp\_updater.exe，选择“以管理员身份运行”。
   * **通过脚本调用：** 在调用更新器的批处理脚本或 PowerShell 脚本中，使用相应的命令（如 runas 或 PowerShell 的 -Verb RunAs）来请求管理员权限启动更新器。
   * **通过主程序触发：** 如果更新是由主应用程序触发的，主程序可以在启动更新器时请求 UAC (User Account Control) 提升。
2. **修改目标目录权限：** 如果应用程序安装在非系统保护目录下（例如用户自定义的安装路径），可以确保运行更新器的用户账户对该目录具有完全控制权限。但这通常不适用于标准安装路径。
3. **使用部署系统：** 专业的软件部署系统（如 SCCM, Intune 等）通常以系统权限运行代理，可以确保更新操作拥有足够的权限。

**强烈建议：** 在大多数标准部署场景下，**始终以管理员权限运行 imp\_updater.exe**，以避免因权限不足导致的更新失败。

## 4.3 集成到应用程序

将更新器集成到您的应用程序更新流程中，通常涉及以下几个步骤：

1. **检查更新：** 您的主应用程序需要有一个机制来检查是否有可用的新版本。这通常通过访问一个更新服务器、读取一个版本文件或通过其他方式实现。
2. **下载更新包：** 如果检测到新版本，主应用程序需要将包含新版本文件的更新包（一个文件夹）下载到本地的一个临时位置（这将作为更新器的 -source\_path）。
3. **准备命令行参数：** 根据检测到的更新信息和当前的应用程序状态，构建调用更新器所需的命令行参数字符串。关键参数包括：
   * -old\_version\_process\_name 或 -old\_version\_process\_id: 当前运行的主程序进程名或ID。
   * -source\_path: 下载的更新包所在的路径。
   * -destination\_path: 主应用程序的安装路径。
   * -latest\_version: 新版本的版本号（用于备份目录命名和日志记录）。
   * 其他根据需要配置的参数（备份、快捷方式、UI模式、服务器报告等）。
4. **调用更新器：** 主应用程序在准备退出或在用户同意后，启动 imp\_updater.exe 进程，并将构建好的命令行参数传递给它。
   * **权限提升：** 如前所述，启动更新器时很可能需要请求管理员权限。
   * **示例 (C++):**
   * #include <windows.h>  
     #include <shellapi.h>  
     #include <string>  
       
     std::string updaterPath = "C:\\Program Files\\MyApp\\imp\_updater.exe"; // 或者相对路径  
     std::string cmdLine =   
      " -old\_version\_process\_name MyApp.exe"  
      " -source\_path C:\\MyAppUpdates\\NewVersion"  
      " -destination\_path \"C:\\Program Files\\MyApp\""  
      " -latest\_version 2.0.0.0"  
      " -is\_backup\_file true"  
      " -window\_style 1";  
       
     SHELLEXECUTEINFO sei = { sizeof(sei) };  
     sei.lpVerb = "runas"; // 请求管理员权限  
     sei.lpFile = updaterPath.c\_str();  
     sei.lpParameters = cmdLine.c\_str();  
     sei.nShow = SW\_SHOWNORMAL;  
       
     if (!ShellExecuteEx(&sei)) {  
      // 处理启动更新器失败的情况  
      DWORD error = GetLastError();  
      // ... log error ...  
     }  
     else {  
      // 更新器已启动，主程序可以准备退出了  
      // PostQuitMessage(0); // 例如  
     }
   * **示例 (Batch Script):**
   * @echo off  
     set UPDATER\_PATH="C:\Program Files\MyApp\imp\_updater.exe"  
     set SOURCE\_PATH="C:\MyAppUpdates\NewVersion"  
     set DEST\_PATH="C:\Program Files\MyApp"  
     set NEW\_VERSION=2.0.0.0  
       
     REM 这里需要一种方式来请求管理员权限，例如使用第三方工具或 VBScript  
     REM 或者假设此脚本本身就是以管理员权限运行的  
       
     %UPDATER\_PATH% ^  
      -old\_version\_process\_name MyApp.exe ^  
      -source\_path %SOURCE\_PATH% ^  
      -destination\_path %DEST\_PATH% ^  
      -latest\_version %NEW\_VERSION% ^  
      -is\_backup\_file true ^  
      -window\_style 1  
       
     REM 主程序脚本可以在调用更新器后退出  
     exit /b
5. **主程序退出：** 在成功启动更新器后，主应用程序应该自行退出，以便更新器能够顺利结束其进程并替换文件。确保在调用 ShellExecuteEx 或类似函数后，主程序有逻辑来终止自身。
6. **更新器执行：** 更新器接管更新过程，执行参数指定的任务。
7. **（可选）更新后操作：** 更新器可以通过 -start\_process\_file\_path 和 -start\_process\_command\_line 参数在更新成功后自动启动新版本的应用程序。如果未使用此功能，则用户需要手动重新启动应用程序。

**集成注意事项：**

* **错误处理：** 主程序需要处理启动更新器失败的情况（例如文件未找到、权限不足）。
* **用户体验：** 在启动更新器之前，最好给用户明确的提示，告知即将进行更新，并可能需要关闭应用程序。
* **静默更新：** 如果希望实现对用户干扰最小的静默更新，应使用 -is\_hide\_window true 参数，并确保主程序能够正确处理更新检查、下载和调用更新器的逻辑，同时处理好权限问题。
* **更新器自身的更新：** 如果更新器本身也需要更新，需要设计一个更复杂的引导程序 (bootstrapper) 逻辑，先更新更新器，再用新的更新器去更新主程序。

通过仔细规划部署结构、确保权限到位并将调用逻辑正确集成到应用程序中，您可以利用 imp\_updater.exe 实现稳定可靠的软件更新。

# 第4章：命令行参数详解

应用程序软件更新器 (imp\_updater.exe) 的核心驱动力来自于其强大的命令行参数系统。通过在启动更新器时传递不同的参数组合，您可以精确地控制更新过程的每一个细节。本章将详细解析所有可用的命令行参数及其用法。

## 5.1 参数格式

更新器的命令行参数遵循标准的格式：

* 每个参数以一个连字符 (-) 开头，后跟参数名称（例如 -source\_path）。
* 参数名称是大小写敏感的（虽然代码实现中可能使用了不区分大小写的比较，但建议遵循代码中定义的格式）。
* 如果参数需要一个值，该值紧跟在参数名称之后，用空格分隔（例如 -source\_path C:\Updates）。
* 如果参数值包含空格，必须使用双引号将其括起来（例如 -destination\_path "C:\Program Files\MyApp"）。
* 布尔类型的参数（表示“是/否”或“启用/禁用”）通常接受 true 或 false 作为值（例如 -is\_backup\_file true）。根据代码实现，某些布尔参数可能仅通过出现即表示 true，但这不明确，建议显式指定 true 或 false。
* 可以在一个命令行中指定多个参数。

**示例：**

imp\_updater.exe -parameter1 value1 -parameter2 "value with spaces" -boolean\_flag true -another\_param 123

## 5.2 核心参数

这些参数定义了更新操作最基本的信息，通常是必需的（除非使用 -is\_only\_start\_process 模式）。

1. **-old\_version\_process\_name**
   * **注解：** 旧版本进程名
   * **说明：** 指定需要更新的应用程序的进程名称（例如 MyApp.exe）。更新器将使用此名称来查找并（根据配置）终止正在运行的旧版本实例，以解除文件占用。如果同时指定了 -old\_version\_process\_id，则 ID 优先。
   * **值类型：** 字符串 (进程文件名)
   * **默认值：** 无 (空字符串 "")
   * **必需性：** 在标准更新模式下通常是必需的（除非提供了 -old\_version\_process\_id）。
2. **-old\_version\_process\_id**
   * **注解：** 旧版本进程ID
   * **说明：** 指定需要更新的应用程序的进程ID (PID)。这是一个数字，可以更精确地定位特定进程实例。如果提供了此参数，更新器将优先使用 PID 来处理旧进程，而不是进程名。
   * **值类型：** 整数 (DWORD)
   * **默认值：** 0
   * **必需性：** 在标准更新模式下，如果未提供 -old\_version\_process\_name，则此参数可能是必需的。
3. **-source\_path**
   * **注解：** 源路径（更新的文件夹路径）
   * **说明：** 指定包含新版本应用程序文件的文件夹路径。更新器将从此路径递归复制所有文件和子目录到目标路径。
   * **值类型：** 字符串 (有效的文件系统路径)
   * **默认值：** 无 (空字符串 "")
   * **必需性：** 在标准更新模式下是必需的。
4. **-destination\_path**
   * **注解：** 目标路径（需要更新的文件夹路径）
   * **说明：** 指定当前安装的旧版本应用程序所在的文件夹路径。这是新文件将被复制到的地方，旧文件将被覆盖（或先备份）。
   * **值类型：** 字符串 (有效的文件系统路径)
   * **默认值：** 无 (空字符串 "")
   * **必需性：** 在标准更新模式下是必需的。
5. **-latest\_version**
   * **注解：** 最新版本号（备份文件时会使用到）
   * **说明：** 指定正在安装的新版本的版本号字符串（例如 1.2.3.4）。此版本号主要用于在创建备份时，生成具有版本标识的备份文件夹名称（例如 ...\imp\_updater\_backup\<进程名>\<版本号>\_backup）。它也可能被记录在日志或上传的更新记录中。
   * **值类型：** 字符串
   * **默认值：** "0.0.0.0"
   * **必需性：** 虽然有默认值，但强烈建议在启用备份时提供准确的版本号，以便管理和识别备份。

## 5.3 行为控制参数

这些参数用于控制更新过程中的具体行为和附加操作。

1. **-is\_create\_shortcut**
   * **注解：** 是否创建快捷方式
   * **说明：** 控制是否在更新完成后为目标应用程序的可执行文件创建或更新快捷方式。具体创建位置（桌面、开始菜单）和行为（是否覆盖）可能依赖于内部实现逻辑（代码中有 create\_shortcut 函数，但细节未完全在参数注释中说明）。
   * **值类型：** 布尔 (true / false)
   * **默认值：** true
2. **-is\_recursion\_replace\_shortcut**
   * **注解：** 是否递归替换快捷方式（遍历整个桌面，替换所有快捷方式）
   * **说明：** 当 -is\_create\_shortcut 为 true 时生效。如果此参数也为 true，更新器会尝试查找并替换桌面上所有指向旧版本应用程序的快捷方式。这是一个更强的替换逻辑。
   * **值类型：** 布尔 (true / false)
   * **默认值：** false
3. **-is\_delete\_update\_file**
   * **注解：** 是否删除更新文件
   * **说明：** 控制在更新成功完成后，是否删除源路径 (-source\_path) 下的更新文件。这有助于清理临时下载的更新包。
   * **值类型：** 布尔 (true / false)
   * **默认值：** true
4. **-is\_backup\_file**
   * **注解：** 是否备份文件
   * **说明：** 控制在用新文件覆盖目标路径 (-destination\_path) 中的旧文件之前，是否先将旧文件备份。
   * **值类型：** 布尔 (true / false)
   * **默认值：** true
5. **-is\_compress\_backup\_file**
   * **注解：** 是否压缩备份文件（仅备份文件时生效）
   * **说明：** 仅当 -is\_backup\_file 为 true 时生效。如果此参数为 true，备份的文件将被压缩成一个存档文件（可能是 .zip 格式，依赖于 miniz\_compress\_folder 或 libzip\_compress 的实现）存放在备份目录中，以节省磁盘空间。
   * **值类型：** 布尔 (true / false)
   * **默认值：** false
6. **-is\_delete\_old\_version\_backup\_file**
   * **注解：** 是否删除旧版本备份文件
   * **说明：** 控制在更新成功完成后，是否尝试删除之前为**同一应用程序**（基于进程名）创建的**其他版本**的备份文件。这有助于管理备份占用的空间，只保留最近一次更新前的备份（或根据具体实现逻辑决定删除哪些）。注意，它删除的是**旧版本**的备份，而不是本次更新产生的备份。
   * **值类型：** 布尔 (true / false)
   * **默认值：** false
7. **-is\_wait\_old\_version\_exit**
   * **注解：** 是否等待旧版本退出
   * **说明：** 控制在开始复制文件之前，如何处理找到的旧版本进程。如果为 true，更新器会等待旧进程自行退出，直到达到超时时间 (-wait\_old\_version\_exit\_timeout)。如果为 false（或未指定），更新器可能会尝试直接终止旧进程（行为可能依赖于 -kill\_old\_version\_process\_flag）。
   * **值类型：** 布尔 (true / false)
   * **默认值：** false
8. **-wait\_old\_version\_exit\_timeout**
   * **注解：** 等待旧版本退出超时
   * **说明：** 仅当 -is\_wait\_old\_version\_exit 为 true 时生效。指定等待旧版本进程自行退出的最长时间（单位：毫秒）。如果在超时时间内进程仍未退出，更新器可能会根据 -kill\_old\_version\_process\_flag 的设置尝试强制终止它，或者报告错误并中止更新。
   * **值类型：** 整数 (毫秒)
   * **默认值：** 3000 (3秒)
9. **-kill\_old\_version\_process\_flag**
   * **注解：** 杀死旧版本进程标志（0：根据进程ID进行关闭，1：根据进程名进行关闭）
   * **说明：** 控制在需要终止旧版本进程时（例如 -is\_wait\_old\_version\_exit 为 false，或等待超时后），使用哪种方式来查找并终止进程。
     + 0: 优先使用 -old\_version\_process\_id (如果提供且有效) 来终止特定进程实例。
     + 1: 使用 -old\_version\_process\_name 来终止所有同名进程。
   * **值类型：** 整数 (0 或 1)
   * **默认值：** 0
10. **-start\_process\_file\_path**
    * **注解：** 启动进程文件路径（仅在开启只启动进程时生效）
    * **说明：** 主要用于 -is\_only\_start\_process true 模式，指定需要启动的程序的完整路径。也可以在标准更新模式下使用，表示更新成功后需要自动启动的新版本应用程序的主程序路径。如果未指定，更新器更新完成后不会自动启动任何程序。
    * **值类型：** 字符串 (有效的文件系统路径)
    * **默认值：** 无 (空字符串 "")
11. **-start\_process\_command\_line**
    * **注解：** 启动进程命令行参数
    * **说明：** 配合 -start\_process\_file\_path 使用。指定在启动新进程时需要传递给它的命令行参数。
    * **值类型：** 字符串
    * **默认值：** 无 (空字符串 "")
12. **-start\_process\_working\_directory**
    * **注解：** 启动进程工作路径（仅在开启只启动进程时生效）
    * **说明：** 配合 -start\_process\_file\_path 使用。指定启动新进程时的工作目录 (Working Directory)。如果未指定，工作目录可能是新进程文件所在的目录或更新器的工作目录，具体取决于 start\_process 函数的实现。
    * **值类型：** 字符串 (有效的文件系统路径)
    * **默认值：** 无 (空字符串 "")
13. **-is\_only\_start\_process**
    * **注解：** 是否只启动进程（如果为true，窗口将会被强制隐藏）
    * **说明：** 一个特殊的模式开关。如果设置为 true，更新器将**跳过**所有文件复制、备份、快捷方式创建等标准更新流程，仅执行启动由 -start\_process\_file\_path 指定的进程的操作。在此模式下，窗口会被强制隐藏 (-is\_hide\_window 会被设为 true)，并且日志记录等功能可能会被禁用。
    * **值类型：** 布尔 (true / false)
    * **默认值：** false

## 5.4 界面与交互参数

这些参数控制更新器运行时是否显示窗口、窗口的样式以及错误提示方式。

1. **-is\_hide\_window**
   * **注解：** 是否隐藏窗口
   * **说明：** 控制更新器是否在后台静默运行，不显示任何用户界面。如果为 true，则忽略 -window\_style, -window\_title, -window\_width, -window\_height 等界面相关参数。注意，如果 -is\_only\_start\_process 为 true，此参数会被强制设为 true。
   * **值类型：** 布尔 (true / false)
   * **默认值：** false
2. **-is\_show\_topmost**
   * **注解：** 是否将窗口置顶
   * **说明：** 如果窗口可见（-is\_hide\_window 为 false），此参数控制更新器窗口是否保持在其他所有窗口的最顶层。
   * **值类型：** 布尔 (true / false)
   * **默认值：** false
3. **-window\_title**
   * **注解：** 窗口标题（仅窗口样式0时生效）
   * **说明：** 仅当窗口样式 -window\_style 为 0 (标准窗口) 时生效。设置更新器窗口的标题栏文本。
   * **值类型：** 字符串
   * **默认值：** 从资源文件加载 (IDS\_APP\_TITLE)，如果未提供此参数。
4. **-window\_width**
   * **注解：** 窗口宽度（仅窗口样式0时生效）
   * **说明：** 仅当窗口样式 -window\_style 为 0 (标准窗口) 时生效。设置更新器窗口的宽度（单位：像素）。
   * **值类型：** 整数
   * **默认值：** 500
5. **-window\_height**
   * **注解：** 窗口高度（仅窗口样式0时生效）
   * **说明：** 仅当窗口样式 -window\_style 为 0 (标准窗口) 时生效。设置更新器窗口的高度（单位：像素）。
   * **值类型：** 整数
   * **默认值：** 350
6. **-window\_style**
   * **注解：** 窗口样式（0：常规窗口样式，1：动画样式）
   * **说明：** 控制更新器窗口的外观和行为（前提是 -is\_hide\_window 为 false）。
     + 0: 显示一个包含状态文本和详细日志列表的标准窗口。
     + 1: 显示一个带有旋转加载动画和进度百分比的简洁窗口。
   * **值类型：** 整数 (0 或 1)
   * **默认值：** 1
7. **-is\_prompt\_if\_error**
   * **注解：** 如果错误，是否弹窗提示（仅在隐藏窗口和窗口样式为1时生效）
   * **说明：** 控制在更新过程中发生错误时，是否通过弹窗消息框 (MessageBox) 通知用户。此选项主要用于在界面不显示详细日志的情况下（隐藏模式 -is\_hide\_window true 或动画模式 -window\_style 1）提供错误反馈。
   * **值类型：** 布尔 (true / false)
   * **默认值：** true

## 5.5 高级参数

这些参数涉及一些高级功能，如与服务器通信、处理特定文件等。

1. **-sensitive\_dll**
   * **注解：** 敏感的动态链接库（将会绕过杀毒软件拦截，格式（a.dll;b.dll））
   * **说明：** 指定一个或多个被认为是“敏感”的 DLL 文件名，用分号 (;) 分隔。更新器在处理这些文件时可能会采取特殊措施，注释中提到“绕过杀毒软件拦截”，但这通常是非常规且风险较高的行为，具体实现和效果需要详细测试和评估。使用此参数需谨慎。
   * **值类型：** 字符串 (分号分隔的DLL文件名列表)
   * **默认值：** 无 (空字符串 "")
   * **内部列表：** 代码中包含一个硬编码的 often\_intercepted\_dll\_name 列表，这些 DLL 可能默认被视为敏感或在某些操作中被特殊处理，即使不通过此参数指定。
2. **-server\_address**
   * **注解：** 服务器地址（用于上传更新记录）
   * **说明：** 指定用于接收更新记录的服务器的域名或 IP 地址。
   * **值类型：** 字符串
   * **默认值：** "www.mes.com"
3. **-server\_port**
   * **注解：** 服务器端口（用于上传更新记录）
   * **说明：** 指定接收更新记录的服务器的端口号。
   * **值类型：** 整数
   * **默认值：** 60682
4. **-upload\_update\_record\_flag**
   * **注解：** 上传更新记录标志（0：不上传，1：不包含更新日志，2：如果失败包含更新日志，3：包含更新日志）
   * **说明：** 控制是否以及如何将更新结果上传到由 -server\_address 和 -server\_port 指定的服务器。
     + 0: 不上传任何信息。
     + 1: 上传更新结果（成功/失败、耗时等基本信息），但不包含详细的日志文本。
     + 2: 仅在更新失败时，上传更新结果和详细的日志文本（受 -upload\_update\_log\_max\_size 限制）。
     + 3: 无论成功或失败，都上传更新结果和详细的日志文本（受 -upload\_update\_log\_max\_size 限制）。
   * **值类型：** 整数 (0, 1, 2, 或 3)
   * **默认值：** 1
5. **-upload\_update\_log\_max\_size**
   * **注解：** 上传更新日志最大大小（单位：MB，默认5MB）
   * **说明：** 当 -upload\_update\_record\_flag 为 2 或 3 时生效。限制上传的日志文本的最大大小（单位：MB）。如果实际日志超过此大小，可能会被截断或不上传。
   * **值类型：** 整数 (MB)
   * **默认值：** 5

## 5.6 辅助命令

除了上述控制更新流程的参数外，更新器还支持一些特殊的命令行参数，用于获取信息或执行维护任务，它们通常单独使用，不会触发标准的更新流程。

* **-help**
  + **说明：** 显示一个包含基本用法和可用辅助命令列表的消息框。
  + **示例：** imp\_updater.exe -help
* **-option**
  + **说明：** 显示一个包含所有可用命令行参数及其注解的消息框。这是快速查阅所有参数的便捷方式。
  + **示例：** imp\_updater.exe -option
* **-default\_sensitive\_dll**
  + **说明：** 显示一个包含内部硬编码的“默认敏感DLL”列表的消息框。这些DLL可能在更新过程中受到特殊处理。
  + **示例：** imp\_updater.exe -default\_sensitive\_dll
* **-delete\_all\_backup\_file**
  + **说明：** 立即尝试删除位于用户 LOCALAPPDATA 目录下的整个 imp\_updater\_backup 文件夹及其所有内容。这是一个清理操作，会移除所有由更新器创建的文件备份。
  + **示例：** imp\_updater.exe -delete\_all\_backup\_file

## 5.7 参数组合示例

以下是一些常见场景的参数组合示例：

**场景1：标准更新，带备份和快捷方式，显示动画窗口，完成后删除更新包**

imp\_updater.exe ^  
 -old\_version\_process\_name MyApp.exe ^  
 -source\_path C:\Updates\MyApp\_v2 ^  
 -destination\_path "C:\Program Files\MyApp" ^  
 -latest\_version 2.0.0.0 ^  
 -is\_backup\_file true ^  
 -is\_compress\_backup\_file false ^  
 -is\_create\_shortcut true ^  
 -is\_delete\_update\_file true ^  
 -window\_style 1 ^  
 -upload\_update\_record\_flag 1 ^  
 -server\_address updates.mycompany.com ^  
 -server\_port 8080

**场景2：静默更新，不备份，不创建快捷方式，强制终止旧进程，更新后启动新程序**

imp\_updater.exe ^  
 -old\_version\_process\_name Service.exe ^  
 -source\_path C:\Temp\ServiceUpdate ^  
 -destination\_path "C:\Services\MyService" ^  
 -latest\_version 1.5.0 ^  
 -is\_backup\_file false ^  
 -is\_create\_shortcut false ^  
 -is\_delete\_update\_file true ^  
 -is\_hide\_window true ^  
 -is\_wait\_old\_version\_exit false ^  
 -kill\_old\_version\_process\_flag 1 ^  
 -start\_process\_file\_path "C:\Services\MyService\Service.exe" ^  
 -upload\_update\_record\_flag 3 ^  
 -server\_address logserver.internal

**场景3：仅启动一个程序（非更新操作），隐藏窗口**

imp\_updater.exe ^  
 -is\_only\_start\_process true ^  
 -start\_process\_file\_path "C:\Tools\Utility.exe" ^  
 -start\_process\_command\_line "-config C:\config.xml" ^  
 -start\_process\_working\_directory "C:\Tools"

**场景4：更新时显示标准窗口，包含详细日志，不置顶，出错时弹窗提示**

imp\_updater.exe ^  
 -old\_version\_process\_name Editor.exe ^  
 -source\_path X:\NewEditor ^  
 -destination\_path "D:\MyApps\Editor" ^  
 -latest\_version 3.1 ^  
 -is\_backup\_file true ^  
 -is\_delete\_update\_file false ^  
 -window\_style 0 ^  
 -window\_title "Editor 更新程序" ^  
 -window\_width 600 ^  
 -window\_height 400 ^  
 -is\_show\_topmost false ^  
 -is\_prompt\_if\_error true ^  
 -upload\_update\_record\_flag 0

通过灵活运用这些参数，您可以为不同的应用程序和部署环境定制精确的更新策略。

# 第5章：更新流程详解

当应用程序软件更新器 (imp\_updater.exe) 被调用时，它会按照一系列预定义的步骤来执行更新任务。理解这个流程有助于您更好地配置参数、预测行为以及在出现问题时进行诊断。本章将详细分解更新器在标准更新模式下的主要工作流程。

请注意，如果使用了特殊模式参数 -is\_only\_start\_process true，更新器将跳过大部分标准流程，仅执行启动指定进程的操作。

## 6.1 启动更新器

更新流程始于外部调用 imp\_updater.exe，并通过命令行传递参数。这通常由主应用程序的更新检查逻辑、部署脚本或用户手动触发。

1. **进程启动：** imp\_updater.exe 进程开始运行。
2. **命令行解析：** 更新器首先解析传递给它的命令行字符串。它会识别所有 - 开头的参数及其对应的值，并将这些配置存储在内部的数据结构中（如 cmdline\_option\_t 结构体）。代码中 parse\_command\_line\_option 函数负责此任务。
3. **辅助命令处理：** 在解析参数后，更新器会检查是否接收到了特殊的辅助命令，如 -help, -option, -default\_sensitive\_dll, -delete\_all\_backup\_file。如果检测到这些命令，更新器会执行相应的操作（通常是显示一个消息框或执行清理任务），然后**直接退出**，不会进入标准的更新流程。
4. **模式判断：** 检查是否设置了 -is\_only\_start\_process true。如果是，则跳转到仅启动进程的简化流程（见 6.8 节相关部分），否则继续标准更新流程。

## 6.2 检查参数与环境

在正式开始文件操作之前，更新器会进行一系列的检查，以确保更新可以安全、正确地进行。

1. **核心参数校验：** 检查标准更新模式下必需的参数是否已提供且基本有效。
   * check\_path\_is\_valid 函数会验证 -source\_path 和 -destination\_path 是否存在且可访问。如果路径无效，更新可能会中止。
   * 检查 -old\_version\_process\_name 或 -old\_version\_process\_id 是否至少提供了一个，用于定位旧进程。
2. **初始化日志：** 调用 open\_log\_file 函数。如果不是仅启动进程模式，它会在更新器所在目录下创建 install\_log 子目录（如果不存在），并以当前时间戳创建一个 .log 文件用于记录后续操作。
3. **初始化界面：** 根据 -is\_hide\_window 和 -window\_style 参数，创建并（根据需要）显示更新器窗口。InitInstance 函数处理窗口的创建、定位、样式设置（标准、动画或隐藏）。如果窗口可见，会设置定时器（用于动画模式）或创建静态文本和列表框控件（用于标准模式）。
4. **记录启动信息：** 向日志文件写入启动信息和解析到的命令行参数。

## 6.3 处理旧版本进程

这是更新流程中的关键一步，目的是确保旧版本应用程序已完全退出，以便其文件可以被安全地替换。

1. **定位旧进程：** 根据 -kill\_old\_version\_process\_flag 的设置，优先使用 -old\_version\_process\_id 或 -old\_version\_process\_name 来查找正在运行的旧版本进程。get\_all\_processes 函数可能用于获取系统当前所有进程列表以进行匹配。
2. **等待或终止：**
   * **等待模式 (-is\_wait\_old\_version\_exit true)：** 更新器会进入等待状态，周期性地检查目标进程是否已退出。这个等待过程有超时限制，由 -wait\_old\_version\_exit\_timeout 参数指定（默认3秒）。wait\_process\_exit 或类似的逻辑（如 check\_process\_is\_exit 配合循环）会执行此操作。
   * **直接终止模式 (-is\_wait\_old\_version\_exit false 或等待超时后)：** 更新器会尝试强制终止找到的旧进程。terminate\_process 函数（内部可能调用 Windows API 如 TerminateProcess）会被用来结束进程。终止方式（按ID或名称）由 -kill\_old\_version\_process\_flag 控制。
3. **结果处理：**
   * 如果成功找到并使旧进程退出（无论是自行退出还是被终止），流程继续。
   * 如果找不到指定的旧进程，更新器可能会记录一个警告但继续执行（假设程序未运行）。
   * 如果在等待超时后仍无法终止进程（例如权限不足），更新器会记录错误，并可能中止更新过程，通过 update\_finish 函数报告失败。
4. **日志记录：** 详细记录查找、等待、终止旧进程的操作和结果。

## 6.4 备份旧文件

如果用户通过 -is\_backup\_file true 参数启用了备份功能，更新器会在覆盖文件之前执行备份。

1. **检查备份开关：** 确认 -is\_backup\_file 为 true。
2. **确定备份路径：** 根据 -old\_version\_process\_name 和 -latest\_version 构建备份目标文件夹的路径（通常在 LOCALAPPDATA\imp\_updater\_backup\<进程名>\<版本号>\_backup）。
3. **创建备份目录：** 使用 create\_path 或类似逻辑确保备份目录存在。
4. **执行备份：** 调用 backup\_file 函数。
   * 该函数会遍历目标路径 (-destination\_path) 下的文件。
   * 根据版本 1.0.0.3 的更新日志，优化后的逻辑是**只备份那些在源路径 (-source\_path) 中也存在对应新版本的文件**，而不是备份整个目标目录。
   * **非压缩模式 (-is\_compress\_backup\_file false)：** 将需要备份的旧文件直接复制到备份目录下，保持原始的目录结构。
   * **压缩模式 (-is\_compress\_backup\_file true)：** 将需要备份的旧文件添加到一个压缩包（如 .zip）中，并将该压缩包存放在备份目录下。这通常通过 miniz\_compress\_folder 或 libzip\_compress 实现。
5. **进度回调（可能）：** 备份函数（尤其是压缩）可能支持一个进度回调 (progress\_cb\_t)，用于更新界面上的进度信息（例如动画窗口的百分比）。
6. **错误处理：** 如果在备份过程中发生错误（如磁盘空间不足、文件访问权限问题），会记录错误。根据严重程度，更新可能会中止。
7. **日志记录：** 记录备份操作的开始、结束、是否压缩以及任何错误。

## 6.5 复制新文件

这是更新的核心步骤，将新版本的文件应用到目标位置。

1. **准备复制：** 确认源路径 (-source\_path) 和目标路径 (-destination\_path) 有效。
2. **检查文件占用（优化）：** 虽然主要依赖于预先结束旧进程，但代码中可能仍有检查目标文件是否被占用的逻辑（如 is\_occupied 或 check\_file\_is\_occupy，get\_occupied\_file\_pids）。版本 1.0.0.3 的日志提到优化了此检查，仅检查目标父目录中的文件，不递归子文件夹。如果发现意外的文件占用，可能会尝试再次终止相关进程（kill\_occupied\_file\_process）或报告错误。
3. **执行复制：** 调用 copy\_folder 函数（或类似的文件遍历与复制逻辑）。
   * 该函数会递归地遍历源路径 (-source\_path)。
   * 对于源路径中的每个文件和子目录，它会在目标路径 (-destination\_path) 中创建对应的项。
   * 使用 copy\_file 函数（内部调用 Windows API 如 CopyFileEx）将源文件复制到目标位置，覆盖同名的现有文件。
   * **敏感DLL处理：** 如果 -sensitive\_dll 参数指定了某些DLL，或者内部列表包含某些DLL，复制这些文件时可能采取特殊的文件操作方式（具体细节不明，需谨慎）。
4. **进度更新：** 文件复制过程中，可能会通过回调更新界面进度（uInstallProgress 变量和 ChangeStatusText 函数）。
5. **错误处理：** 如果在复制过程中发生错误（如磁盘满、权限问题、文件损坏），会记录详细错误信息 (set\_last\_error)。关键文件的复制失败通常会导致整个更新过程失败。
6. **日志记录：** 记录文件复制的开始、结束、复制的文件数量、遇到的任何错误。

## 6.6 创建快捷方式

如果 -is\_create\_shortcut true，更新器会尝试创建或更新快捷方式。

1. **检查开关：** 确认 -is\_create\_shortcut 为 true。
2. **确定目标文件：** 通常是目标路径下的主应用程序可执行文件（可能需要根据 -old\_version\_process\_name 或其他逻辑确定）。
3. **执行创建/替换：** 调用 create\_shortcut 函数。
   * 该函数使用 Windows Shell Link (IShellLink) API 来创建 .lnk 文件。
   * **标准模式 (-is\_recursion\_replace\_shortcut false)：** 可能只在特定位置（如桌面）创建一个新的快捷方式，或更新一个已知的快捷方式。
   * **递归替换模式 (-is\_recursion\_replace\_shortcut true)：** 会遍历桌面，查找所有指向旧目标文件的快捷方式，并将其目标更新为新路径。
4. **错误处理：** 创建快捷方式可能因权限不足或路径问题失败，会记录错误。
5. **日志记录：** 记录创建或更新快捷方式的操作和结果。

## 6.7 清理工作

更新的主要文件操作完成后，会进行一些清理。

1. **删除更新文件（可选）：** 如果 -is\_delete\_update\_file true，更新器会尝试删除源路径 (-source\_path) 下的所有文件和文件夹。调用 remove\_folder 函数执行此操作。如果删除失败（例如文件被占用），会记录错误，但这通常不影响更新的整体成功状态。
2. **删除旧备份（可选）：** 如果 -is\_delete\_old\_version\_backup\_file true，更新器会调用 delete\_old\_version\_backup\_file 函数，尝试删除之前为该应用程序（基于进程名）存储的、非当前 -latest\_version 对应的备份目录。
3. **日志记录：** 记录清理操作的结果。

## 6.8 启动新版本

根据配置，更新器可以在完成所有操作后自动启动新更新的应用程序。

1. **检查启动参数：** 检查 -start\_process\_file\_path 是否已指定。
2. **执行启动：** 如果指定了路径，调用 start\_process 函数。
   * 该函数使用 Windows API（如 CreateProcess）来启动指定的应用程序 (-start\_process\_file\_path)。
   * 可以传递命令行参数 (-start\_process\_command\_line)。
   * 可以设置工作目录 (-start\_process\_working\_directory)。
3. **错误处理：** 如果启动新进程失败（如文件未找到、权限问题），会记录错误。
4. **日志记录：** 记录尝试启动新进程的操作和结果。

**对于 -is\_only\_start\_process true 模式：** 此模式下，流程大大简化。在解析参数后，直接跳转到此步骤，执行 start\_process 函数启动指定的程序，然后直接进入结束阶段。

## 6.9 更新完成与报告

无论更新成功还是失败，更新器都会执行最后的收尾工作。

1. **标记完成状态：** 设置内部标志（如 bIsUpgradeFinish）表示更新流程结束。
2. **最终状态更新：** 更新界面状态（如果可见），显示“更新完成”或“更新失败”等信息。
3. **上传更新记录（可选）：** 如果 -upload\_update\_record\_flag 大于 0，调用 upload\_update\_record 函数。
   * 收集更新结果（成功/失败标志 success）、总耗时 (elapsed\_time)。
   * 根据 upload\_update\_record\_flag 的值，决定是否包含日志文本 (log\_filetext)，并注意 -upload\_update\_log\_max\_size 的限制。
   * 构建 HTTP 请求（可能是 POST 请求），将数据发送到 -server\_address 和 -server\_port 指定的服务器。
   * 处理服务器响应，记录上传操作的结果。
4. **错误提示（可选）：** 如果更新失败且 -is\_prompt\_if\_error true（并且窗口是隐藏或动画模式），会弹出一个消息框向用户报告错误（可能显示 last\_error 中的信息）。
5. **关闭日志文件：** 调用 close\_log\_file，确保所有日志信息已写入磁盘。
6. **退出更新器：** 结束 imp\_updater.exe 进程。在窗口模式下，可能需要用户手动关闭窗口，或者在 update\_finish 中触发窗口关闭（例如发送 WM\_CLOSE 或 PostQuitMessage）。代码显示 update\_finish 会根据情况发送 WM\_USER\_HIDE\_WINDOW 或直接调用 exit(0)，具体行为取决于参数和错误状态。

这个详细的流程分解揭示了更新器内部的工作机制，有助于您理解每个参数如何影响更新行为，并在遇到问题时更有针对性地进行排查。

# 第6章：用户界面说明

应用程序软件更新器提供了多种用户界面（UI）模式，以适应不同的更新场景和用户体验需求。您可以根据需要选择显示详细信息的标准窗口、提供视觉反馈的动画窗口，或者在后台完全静默地执行更新。本章将详细介绍这三种主要的界面模式及其相关配置。

选择哪种界面模式主要由 -window\_style 和 -is\_hide\_window 这两个命令行参数控制。

## 7.1 标准窗口模式 (Style 0)

**触发方式：** \* 设置 -window\_style 0 \* 同时确保 -is\_hide\_window 为 false (或不指定，因为其默认值为 false)

**外观与行为：**

标准窗口模式提供了一个传统的 Windows 应用程序窗口，旨在向用户或管理员展示详细的更新过程信息。其主要特点包括：

1. **窗口外观：**
   * 一个标准的、具有标题栏和边框的窗口。
   * **标题：** 默认标题从资源文件加载，但可以通过 -window\_title 参数自定义（例如 -window\_title "MyApp 更新程序"）。
   * **尺寸：** 窗口的宽度和高度可以通过 -window\_width (默认 500 像素) 和 -window\_height (默认 350 像素) 参数进行调整。
   * **窗口行为：** 窗口通常可以移动，但默认配置下可能禁用了最大化按钮和改变大小的功能（WS\_OVERLAPPEDWINDOW & ~WS\_MAXIMIZEBOX, SetWindowLongPtr(hWnd, GWL\_STYLE, GetWindowLongPtr(hWnd, GWL\_STYLE) & ~WS\_THICKFRAME)）。
   * **置顶：** 可以通过设置 -is\_show\_topmost true 使窗口保持在所有其他窗口的顶部。
2. **窗口内容：**
   * **状态栏 (Static Text)：** 窗口顶部通常会有一个静态文本区域（控件 ID: IDC\_STATIC\_STATUS），用于显示当前更新的主要状态信息。例如，“正在检查旧版本…”、“正在备份文件…”、“正在复制文件 (xx%)…”、“更新完成”、“更新失败：[错误信息]”等。这些状态文本由 ChangeStatusText 函数更新。
   * **日志列表框 (ListBox)：** 窗口的主要区域是一个列表框控件（控件 ID: IDC\_LISTBOX\_LOG），用于实时显示详细的更新日志。每一条日志信息（通过 LogInfo, LogWarn, LogError 函数记录）都会被添加为列表框中的一个新行，并带有时间戳。列表框通常支持垂直和水平滚动条，以便查看完整的日志内容。最新的日志条目会自动滚动到视图中。
3. **交互性：**
   * 用户通常可以滚动日志列表框查看历史记录。
   * 窗口可以通过标题栏的标准控件（关闭按钮）来关闭。更新器在完成或失败后，可能会自动关闭窗口，或者需要用户手动关闭，具体行为取决于 update\_finish 函数的实现逻辑。

**适用场景：**

* 需要详细监控更新过程的场景，例如在开发、测试阶段或由管理员手动执行更新时。
* 当用户需要了解更新的具体步骤和可能遇到的问题时。
* 不追求极致简洁用户体验的内部应用或工具。

**配置示例：**

imp\_updater.exe ... -window\_style 0 -window\_title "详细更新日志" -window\_width 700 -window\_height 500 ...

## 7.2 动画窗口模式 (Style 1)

**触发方式：** \* 设置 -window\_style 1 (这是默认值，如果不指定 -window\_style 且 -is\_hide\_window 为 false，则会使用此模式) \* 同时确保 -is\_hide\_window 为 false

**外观与行为：**

动画窗口模式旨在提供一个更现代化、对最终用户更友好的视觉反馈，同时隐藏底层的复杂日志信息。其主要特点包括：

1. **窗口外观：**
   * 一个简洁的、通常没有标准边框和标题栏的弹出式窗口 (WS\_POPUP)。
   * **尺寸：** 窗口尺寸通常是固定的，代码中显示为 200x250 像素，旨在屏幕中央显示。
   * **背景：** 窗口背景通常是透明的或单一颜色（代码中使用 WS\_EX\_LAYERED 和 SetLayeredWindowAttributes 设置透明色键，使得背景透明，只显示绘制的内容）。
   * **置顶：** 同样可以通过 -is\_show\_topmost true 使其置顶。
2. **窗口内容：**
   * **加载动画：** 窗口中央会显示一个动态的加载动画。代码中 DrawLoadingAnimation 函数实现了一个由12个彩色圆点组成的旋转动画，通过定时器 (TIMER\_ID) 驱动帧更新 (WM\_TIMER 消息处理)。
   * **进度文本：** 动画下方会显示更新进度的文本，通常格式为“更新中 xx%”。这个百分比值 (uInstallProgress) 在文件备份和复制等耗时操作中被更新。文本颜色可能会动态变化（代码显示使用了 colors 数组循环设置文本颜色）。
3. **交互性：**
   * 此模式下的窗口通常是不可交互的，用户无法移动、关闭（除非更新器自行关闭）或查看详细日志。
   * 更新完成后，窗口会自动消失。
4. **错误提示：**
   * 由于此模式不显示详细日志，错误反馈依赖于 -is\_prompt\_if\_error 参数。如果该参数为 true (默认值)，在更新失败时会弹出一个包含错误信息的消息框。

**适用场景：**

* 面向最终用户的应用程序更新，希望提供简洁、美观的更新进度反馈，避免过多技术细节干扰。
* 作为默认的界面模式，平衡了视觉反馈和简洁性。

**配置示例：**

imp\_updater.exe ... -window\_style 1 -is\_show\_topmost true -is\_prompt\_if\_error true ...

(由于 window\_style 1 是默认值，可以省略 -window\_style 1)

## 7.3 隐藏窗口模式

**触发方式：** \* 设置 -is\_hide\_window true

**外观与行为：**

隐藏窗口模式（也称为静默模式）使更新器在后台运行，完全不显示任何用户界面。

1. **无可见窗口：** 更新器进程启动后，不会创建任何可见的窗口。用户不会察觉到更新器正在运行（除非通过任务管理器查看进程）。
2. **忽略界面参数：** 所有与窗口相关的参数（-window\_style, -window\_title, -window\_width, -window\_height, -is\_show\_topmost）在此模式下都会被忽略。
3. **日志记录：** 尽管没有界面显示日志，但日志文件（在 install\_log 目录）通常仍然会生成（除非被 -is\_only\_start\_process 模式禁用），记录详细的后台操作。
4. **错误提示：**
   * 错误反馈同样依赖于 -is\_prompt\_if\_error 参数。如果该参数为 true (默认值)，即使在隐藏模式下，更新失败时**仍然会弹出一个错误消息框**。这可能会打断用户的静默体验。
   * 如果希望实现完全静默的更新（即使失败也不弹窗），需要设置 -is\_prompt\_if\_error false。在这种情况下，更新结果只能通过检查日志文件或服务器上传的记录来判断。

**适用场景：**

* 需要完全自动化的后台更新，对最终用户透明，不希望有任何干扰。
* 通过集中管理工具（如 SCCM）部署更新，由系统账户执行，不需要用户交互。
* 服务器应用程序或服务的更新。

**配置示例：**

* **完全静默更新（失败也不弹窗）：** bash imp\_updater.exe ... -is\_hide\_window true -is\_prompt\_if\_error false ...
* **静默更新（失败时弹窗提示）：** bash imp\_updater.exe ... -is\_hide\_window true -is\_prompt\_if\_error true ... (由于 is\_prompt\_if\_error true 是默认值，可以省略)

**总结：**

|  |  |  |  |
| --- | --- | --- | --- |
| 特性 | 标准窗口 (Style 0) | 动画窗口 (Style 1) | 隐藏窗口 (Hide True) |
| **触发参数** | -window\_style 0 | -window\_style 1 (默认) | -is\_hide\_window true |
| **可见性** | 可见 | 可见 | 不可见 |
| **外观** | 标准窗口，带标题栏、边框 | 简洁弹出窗口，无边框，透明背景 | 无窗口 |
| **内容** | 状态文本 + 详细日志列表 | 加载动画 + 进度百分比 | 无内容 |
| **交互性** | 可滚动日志，可关闭 | 不可交互 | 不可交互 |
| **日志显示** | 实时显示在列表框中 | 不显示 | 不显示 |
| **错误弹窗** | 不依赖 -is\_prompt\_if\_error (日志可见) | 依赖 -is\_prompt\_if\_error (默认弹窗) | 依赖 -is\_prompt\_if\_error (默认弹窗) |
| **可配置项** | 标题、宽度、高度、置顶 | 置顶 | 无界面配置 |
| **默认模式** | 否 | 是 | 否 |

根据您的具体需求选择合适的界面模式，可以显著影响用户对更新过程的感知和体验。

# 第7章：日志与故障排查

尽管应用程序软件更新器设计力求稳定可靠，但在复杂的实际环境中，更新过程仍可能遇到各种问题。幸运的是，更新器提供了详细的日志记录功能，这是诊断和解决更新失败问题的关键工具。本章将指导您如何查找和理解日志文件，列举一些常见的错误及其可能的解决方案，并说明在需要寻求帮助时如何有效地报告问题。

## 8.1 日志文件

更新器在执行过程中会生成详细的日志文件，记录每一步操作、配置参数、时间戳以及发生的任何警告或错误。

**1. 日志文件位置：**

* 默认情况下，日志文件存储在**更新器 (imp\_updater.exe) 自身所在的目录**下的一个名为 install\_log 的子目录中。
* 例如，如果 imp\_updater.exe 位于 C:\Program Files\MyApp\，那么日志目录就是 C:\Program Files\MyApp\install\_log\。
* **注意：** 更新器需要有权限在此位置创建目录和写入文件。如果权限不足（例如，在系统保护目录下以非管理员权限运行），日志可能无法生成或写入不完整。

**2. 日志文件命名：**

* 每个日志文件都以其创建时的日期和时间命名，格式通常为 YYYYMMDDHHMMSS.log（年年月月日日时时分分秒秒.log）。
* 例如，20250506143005.log 表示该日志文件记录的是从 2025年5月6日 14:30:05 开始的那次更新操作。
* 这种命名方式使得按时间顺序查找特定更新过程的日志变得容易。

**3. 日志文件格式与内容：**

日志文件是纯文本文件，可以使用任何文本编辑器（如记事本、Notepad++）打开查看。每一行日志通常包含以下信息：

* **时间戳：** 精确到毫秒，格式通常为 YYYY-MM-DD HH:MM:SS.ms。
* **日志级别（可选）：** 代码中 write\_log\_type 和 write\_log\_type\_fmt 函数允许指定日志级别，如 [info], [warn], [error]。这有助于快速筛选重要信息。
* **日志消息：** 描述当前执行的操作、状态信息、参数值或错误详情。

**典型的日志内容可能包括：**

* 更新器启动时间。
* 解析到的命令行参数列表。
* 开始处理旧版本进程（查找、等待、终止的结果）。
* 备份操作的开始、结束、备份路径、是否压缩。
* 文件复制操作的开始、结束、源路径、目标路径、复制进度（可能）。
* 创建快捷方式的操作结果。
* 清理操作（删除更新文件、删除旧备份）的结果。
* 尝试启动新进程的操作结果。
* 更新成功或失败的最终状态。
* 遇到的任何错误信息，包括 Windows API 错误代码及其描述（通过 format\_error 函数获取）。
* 上传更新记录到服务器的操作结果（如果启用）。
* 更新器退出时间。

**4. 日志记录的注意事项：**

* **日志级别：** 仔细关注标记为 [warn] 或 [error] 的日志条目，它们通常指示了问题的关键点。
* **日志完整性：** 如果更新器意外崩溃或被强制终止，日志文件可能不完整，最后的操作可能没有被记录下来。
* **日志大小：** 对于非常大的应用程序或长时间运行的更新，日志文件可能会变得很大。虽然代码中没有明确的日志轮转或大小限制逻辑（除了上传大小限制 -upload\_update\_log\_max\_size），但在分析时需要注意。
* **-is\_only\_start\_process 模式：** 在此模式下，open\_log\_file 不会被调用，因此**不会生成**标准的日志文件。
* **上传的日志：** 如果配置了上传日志 (-upload\_update\_record\_flag 为 2 或 3），服务器端收到的日志文本 (log\_filetext) 可能与本地日志文件内容略有不同（例如路径分隔符被替换为 /），并且可能因为大小限制 (-upload\_update\_log\_max\_size) 而被截断。

## 8.2 常见错误与解决方案

以下是一些在更新过程中可能遇到的常见错误及其排查思路和解决方案：

**1. 错误：无法终止旧版本进程 / 文件仍被占用**

* **日志表现：**
  + [error] Failed to terminate process <PID/Name>. Error: Access is denied. (错误代码 5)
  + [error] Failed to copy file from <source> to <destination>. Error: The process cannot access the file because it is being used by another process. (错误代码 32)
  + [warn] Wait for old process <PID/Name> timeout.
* **原因分析：**
  + **权限不足：** 更新器没有足够的权限去终止旧版本进程（例如，旧进程以管理员权限运行，而更新器没有）。
  + **进程定位错误：** -old\_version\_process\_name 或 -old\_version\_process\_id 指定错误，未能找到正确的进程。
  + **进程顽固：** 旧进程可能卡死或有保护机制，难以正常终止。
  + **其他进程占用：** 即使旧的主进程已关闭，其关联的服务、后台任务或其他第三方程序（如杀毒软件、索引服务）可能仍然锁定了目标文件。
* **解决方案：**
  + **确保管理员权限：** 始终以管理员权限运行更新器。
  + **核对进程信息：** 确认 -old\_version\_process\_name 或 -old\_version\_process\_id 准确无误。
  + **调整等待与终止策略：**
    - 尝试增加 -wait\_old\_version\_exit\_timeout 的值，给旧进程更多时间自行退出。
    - 如果允许，使用 -is\_wait\_old\_version\_exit false 直接尝试终止。
    - 尝试切换 -kill\_old\_version\_process\_flag (0 或 1)，看哪种方式更能有效终止进程。
  + **识别并处理其他占用进程：** 使用系统工具（如资源监视器、Process Explorer）查找是哪个进程锁定了文件，并考虑在更新前手动或通过脚本停止相关服务或进程。
  + **检查杀毒软件：** 临时禁用杀毒软件或将其配置为信任更新器和目标应用程序目录，排除干扰。
  + **使用 -sensitive\_dll (谨慎)：** 如果确定是特定DLL被安全软件拦截，可以尝试将其加入 -sensitive\_dll 列表，但这有潜在风险，需充分测试。

**2. 错误：文件复制失败 (权限、路径、磁盘空间)**

* **日志表现：**
  + [error] Failed to copy file from <source> to <destination>. Error: Access is denied. (错误代码 5)
  + [error] Failed to create directory <path>. Error: Access is denied. (错误代码 5)
  + [error] Failed to copy file ... Error: The system cannot find the path specified. (错误代码 3)
  + [error] Failed to copy file ... Error: There is not enough space on the disk. (错误代码 112)
* **原因分析：**
  + **权限不足：** 更新器没有对目标路径 (-destination\_path) 的写入权限，或对源路径 (-source\_path) 的读取权限。
  + **路径无效：** -source\_path 或 -destination\_path 指定的路径不存在或格式错误。
  + **磁盘空间不足：** 目标驱动器没有足够的空间来容纳新文件。
  + **文件损坏：** 源路径中的文件可能已损坏。
* **解决方案：**
  + **确保管理员权限：** 再次确认更新器以管理员权限运行。
  + **检查路径：** 仔细核对 -source\_path 和 -destination\_path 的值是否正确、存在且可访问。注意路径中的空格和特殊字符，确保必要时使用引号。
  + **检查磁盘空间：** 确保目标驱动器有足够的可用空间。
  + **验证源文件：** 检查源路径下的文件是否完整且未损坏。

**3. 错误：备份失败**

* **日志表现：**
  + [error] Failed to create backup directory <path>. Error: Access is denied.
  + [error] Failed to backup file <file>. Error: ...
  + [error] Failed to compress backup folder. Error: ... (如果使用压缩备份)
* **原因分析：**
  + **权限不足：** 更新器没有权限在用户的 LOCALAPPDATA\imp\_updater\_backup 目录下创建子目录或写入文件。
  + **磁盘空间不足：** 存储备份的驱动器空间不足。
  + **压缩库问题：** 如果使用压缩备份，依赖的压缩库（miniz, libzip）可能存在问题或文件损坏。
* **解决方案：**
  + **检查 LOCALAPPDATA 权限：** 确保运行更新器的用户对其 AppData\Local 目录有写入权限。
  + **检查磁盘空间：** 确保备份目标驱动器有足够空间。
  + **尝试禁用压缩：** 设置 -is\_compress\_backup\_file false，看是否能成功备份。如果可以，问题可能出在压缩环节。
  + **清理旧备份：** 使用 -delete\_all\_backup\_file 命令或手动删除旧备份，释放空间。

**4. 错误：无法创建快捷方式**

* **日志表现：** [error] Failed to create shortcut for <target>. Error: ...
* **原因分析：**
  + **权限不足：** 在桌面或开始菜单等公共位置创建快捷方式通常需要管理员权限。
  + **目标路径无效：** 快捷方式指向的目标文件路径不正确。
* **解决方案：**
  + **确保管理员权限：** 以管理员身份运行更新器。
  + **检查目标路径：** 确认更新后的应用程序可执行文件路径是正确的。

**5. 错误：上传更新记录失败**

* **日志表现：**
  + [error] Failed to resolve server address <address>.
  + [error] Failed to connect to server <address>:<port>. Error: ...
  + [error] Failed to send update record. HTTP status code: <status\_code>.
  + (Win7 TLS 问题) [error] Failed to send HTTPS request. Error: ... (可能与 SSL/TLS 握手失败相关)
* **原因分析：**
  + **网络连接问题：** 目标机器无法访问互联网或指定的服务器。
  + **服务器地址或端口错误：** -server\_address 或 -server\_port 配置错误。
  + **服务器未运行或拒绝连接：** 目标服务器宕机、防火墙阻止或应用程序未监听指定端口。
  + **DNS 解析失败：** 无法将服务器域名解析为 IP 地址。
  + **SSL/TLS 问题：** 如果服务器使用 HTTPS，客户端（尤其是旧系统如 Win7）可能不支持服务器要求的加密协议 (如 TLS 1.2/1.3)。
  + **日志过大：** 上传的日志超出了 -upload\_update\_log\_max\_size 限制（服务器端也可能有大小限制）。
* **解决方案：**
  + **检查网络连接：** 确保机器网络通畅，可以 ping 通服务器地址（如果允许 ICMP）。
  + **核对服务器配置：** 确认 -server\_address 和 -server\_port 正确无误。
  + **检查服务器状态和防火墙：** 确保服务器正在运行，监听指定端口，并且网络防火墙允许来自客户端的连接。
  + **检查 DNS：** 尝试在客户端机器上解析服务器域名。
  + **处理 TLS 问题：**
    - 确保客户端操作系统（尤其是 Win7）已打全补丁，并启用了必要的 TLS 版本支持。
    - 考虑服务器端是否可以降低 TLS 要求（有安全风险）。
    - 更新器使用的网络库（如 libcurl）版本是否支持所需的 TLS 版本。
  + **调整日志上传策略：**
    - 尝试设置 -upload\_update\_record\_flag 1（不上传日志），看是否成功。
    - 如果必须上传日志，尝试减小 -upload\_update\_log\_max\_size 的值，或优化日志记录以减少体积。
    - 检查服务器端是否有接收大小限制。

**6. 更新器窗口闪退或无响应**

* **原因分析：**
  + **参数错误：** 传递了无效或冲突的命令行参数，导致更新器在早期初始化阶段崩溃。
  + **环境问题：** 缺少必要的运行时库（如 VC++ Redistributable），或与其他软件冲突。
  + **代码 Bug：** 更新器本身可能存在未处理的异常或逻辑错误。
* **解决方案：**
  + **简化命令行：** 从最简单的参数组合开始（例如只提供核心路径和进程名），逐步添加参数，找出导致问题的参数。
  + **检查依赖：** 确保目标机器安装了更新器所需的运行时环境。
  + **查看系统事件日志：** Windows 事件查看器中的应用程序日志可能记录了更新器崩溃的相关信息。
  + **联系开发者：** 如果怀疑是代码 Bug，需要将问题报告给开发者。

## 8.3 如何报告问题

当您遇到无法自行解决的更新问题，需要向技术支持或开发团队报告时，提供详尽的信息将大大有助于问题的诊断和解决。请尽量收集并提供以下信息：

1. **完整的日志文件：** 这是最重要的信息来源。找到对应更新操作的 .log 文件（位于 install\_log 目录），并将其完整地提供出来。
2. **使用的命令行参数：** 提供您调用 imp\_updater.exe 时使用的完整命令行字符串。
3. **更新器版本：** 如果可能，提供您使用的 imp\_updater.exe 的版本号（可以通过文件属性查看，或询问提供者）。
4. **应用程序信息：**
   * 需要更新的应用程序名称和版本号（更新前和更新后）。
   * 应用程序的安装路径 (-destination\_path)。
5. **环境信息：**
   * 操作系统版本（例如 Windows 10 Pro 64-bit, Version 22H2）。
   * 是否以管理员权限运行更新器。
   * 是否有特殊的安全软件（杀毒软件、防火墙）正在运行。
6. **问题描述：**
   * 清晰地描述您遇到的问题（例如，“更新失败，提示文件被占用”，“动画窗口卡在 50% 不动”，“更新成功但快捷方式未创建”）。
   * 问题发生在哪一步（参照更新流程）？
   * 是否有错误提示框？如有，请提供截图或准确的错误信息文本。
7. **复现步骤：** 如果可能，提供能够稳定复现问题的步骤。
8. **更新包信息（可选）：** 更新包（源路径）的大小、文件数量等信息有时也有帮助。

将这些信息整理清晰后，通过指定的渠道（如内部 Bug 跟踪系统、邮件、技术支持平台）提交给相关人员。提供的信息越全面、准确，问题解决的速度通常越快。

# 第8章：高级主题

在前几章中，我们已经涵盖了应用程序软件更新器的基本用法、核心流程和常见配置。本章将深入探讨一些更高级的功能和使用场景，帮助您更灵活、更深入地利用更新器来满足特定的部署需求。我们将讨论如何处理可能被安全软件拦截的“敏感”DLL，理解现有的备份策略及其局限性，详细配置与服务器的记录集成，以及实现真正的静默更新的最佳实践。

## 9.1 处理敏感DLL

在某些复杂的软件环境中，特定的动态链接库（DLL）文件在复制或加载过程中可能会被某些安全软件（如杀毒软件、主机入侵防御系统 HIPS）误报或拦截，导致更新失败。更新器提供了一个机制来尝试处理这种情况，即通过 -sensitive\_dll 参数指定这些文件。

**参数：-sensitive\_dll**

* **注解：** 敏感的动态链接库（将会绕过杀毒软件拦截，格式（a.dll;b.dll））
* **说明：** 您可以通过此参数提供一个或多个被认为是“敏感”的 DLL 文件名列表，文件名之间用分号 (;) 分隔。例如：-sensitive\_dll Qt5Core.dll;libssl-3.dll;my\_custom\_hook.dll。
* **内部列表：** 除了通过参数指定外，更新器的代码 (utility.cpp) 中还包含一个硬编码的列表 often\_intercepted\_dll\_name，其中预定义了一些常见的、可能被拦截的 DLL（例如 Qt 相关的库、OpenSSL 库、一些 MSVC 运行时库以及特定应用的插件库如 mes\_plugin.dll）。这些内部列表中的 DLL 可能会被自动视为敏感，即使没有在命令行中明确指定。
* **处理机制（推测）：** 注解中提到“将会绕过杀毒软件拦截”，但源代码中并未明确展示具体的“绕过”技术细节。可能的实现方式包括但不限于：
  + 在复制这些特定文件时，使用不同的文件 I/O API 或标志，尝试减少触发安全软件检测的可能性。
  + 在复制前后执行特定的操作，例如临时修改文件属性或权限。
  + **需要强调的是，“绕过”安全软件通常是高风险行为，可能违反安全策略，也可能无法保证在所有安全软件和配置下都有效。** 其确切机制和效果需要通过实际测试来验证。

**何时使用：**

* 当您在更新过程中反复遇到特定 DLL 文件被安全软件隔离、删除或阻止复制，导致更新失败时。
* 在尝试了将更新器和应用程序目录添加到安全软件的信任列表/排除列表但仍然无效后。

**使用建议与风险：**

1. **谨慎使用：** 这是最后的手段，而非首选方案。优先考虑与安全软件供应商沟通或调整安全策略。
2. **精确指定：** 只将确实遇到问题的 DLL 文件名添加到 -sensitive\_dll 参数中，避免不必要的指定。
3. **充分测试：** 在将此参数用于生产环境之前，务必在包含目标安全软件的测试环境中进行充分测试，验证其有效性以及是否引入了其他副作用。
4. **理解风险：** 任何尝试绕过安全机制的行为都可能带来未知的安全风险，需要仔细评估。
5. **查阅内部列表：** 可以使用 -default\_sensitive\_dll 辅助命令查看更新器内部已知的敏感 DLL 列表，了解哪些文件可能已被特殊处理。

**示例：**

imp\_updater.exe ... -sensitive\_dll problematic.dll;another\_one.dll ...

## 9.2 自定义备份策略

更新器提供了基础的文件备份功能，由 -is\_backup\_file 和 -is\_compress\_backup\_file 参数控制。虽然称为“自定义备份策略”，但目前更新器提供的定制化选项相对有限，主要集中在“是否备份”和“是否压缩”两个维度。理解其工作方式和局限性对于制定符合您需求的备份方案很重要。

**现有备份机制回顾：**

* **启用/禁用：** -is\_backup\_file true (默认启用) / false (禁用)。
* **备份内容：** 根据版本 1.0.0.3 的优化，只备份目标路径下那些在源路径中也存在对应新版本的文件。
* **备份位置：** LOCALAPPDATA\imp\_updater\_backup\<进程名>\<版本号>\_backup。
* **压缩选项：** -is\_compress\_backup\_file true (压缩备份) / false (默认不压缩，直接复制)。
* **旧备份清理：** -is\_delete\_old\_version\_backup\_file true (尝试删除旧版本备份) / false (默认不删除)。

**策略考量与局限性：**

1. **备份范围：** 更新器只备份将被覆盖的文件。如果更新包中不包含某个旧文件，那么这个旧文件即使存在于目标目录中，也不会被备份。同样，如果更新包中新增了某个文件，目标目录中没有对应的旧文件，自然也谈不上备份。
2. **无法自定义备份路径：** 备份位置是硬编码在用户 LOCALAPPDATA 目录下的，无法通过参数指定到其他位置（例如网络共享、特定备份分区等）。
3. **无法选择性备份：** 不能指定只备份某些特定类型的文件或排除某些文件不备份（例如大型数据文件、日志文件）。备份是基于源路径和目标路径的文件对比进行的。
4. **备份保留策略有限：** -is\_delete\_old\_version\_backup\_file 只能控制是否删除“旧版本”的备份，但无法实现更精细的策略，例如“保留最近N个版本的备份”或“保留X天内的备份”。
5. **回滚机制：** 更新器本身似乎没有提供一个自动化的“一键回滚”功能（rollback\_file 函数存在但可能未完全集成或暴露）。备份的主要目的是在更新失败或新版本出现严重问题时，提供**手动恢复**的可能性。管理员或用户需要手动访问备份目录，并将文件复制回目标应用程序目录。

**可能的策略组合与建议：**

* **标准策略（默认）：** 启用备份 (-is\_backup\_file true)，不压缩 (-is\_compress\_backup\_file false)，不自动删除旧备份 (-is\_delete\_old\_version\_backup\_file false)。这提供了最直接的回滚可能性，但可能占用较多磁盘空间，且需要手动管理旧备份。
* **节省空间策略：** 启用备份 (-is\_backup\_file true)，启用压缩 (-is\_compress\_backup\_file true)，启用删除旧备份 (-is\_delete\_old\_version\_backup\_file true)。这样可以显著减少备份占用的空间，并自动清理过时的备份。但回滚时需要先解压缩，且只保留了上一个版本的备份。
* **无备份策略（高风险）：** 禁用备份 (-is\_backup\_file false)。这可以加快更新速度并节省空间，但一旦更新失败或新版本有问题，将无法轻易恢复到更新前的状态。**通常不推荐在生产环境中使用此策略，除非有其他可靠的回滚机制（如系统快照、虚拟机快照）。**

**如果需要更高级的备份策略：**

如果更新器内置的备份功能无法满足您的需求（例如需要备份到网络位置、需要更精细的备份范围或保留策略），您可能需要考虑：

* **在调用更新器之前执行自定义备份：** 使用专门的备份工具或脚本，在启动 imp\_updater.exe 之前，按照您的策略备份目标目录。然后，在调用更新器时设置 -is\_backup\_file false 禁用其内置备份。
* **修改更新器源代码：** 如果有源代码访问权限和开发能力，可以修改 backup\_file 函数或相关逻辑，实现更灵活的备份功能。

## 9.3 服务器记录集成

更新器可以将每次更新操作的结果（成功或失败）以及可选的详细日志上传到指定的 HTTP 服务器。这对于集中监控大量客户端的更新状态、收集错误信息、分析更新成功率等非常有用。

**相关参数：**

* **-server\_address:** 服务器的域名或 IP 地址 (默认: www.invomes.com)。
* **-server\_port:** 服务器的端口号 (默认: 60682)。
* **-upload\_update\_record\_flag:** 控制上传行为 (默认: 1)。
  + 0: 不上传。
  + 1: 只上传结果 (成功/失败, 耗时)。
  + 2: 失败时上传结果和日志。
  + 3: 总是上传结果和日志。
* **-upload\_update\_log\_max\_size:** 上传日志的最大大小 (MB, 默认: 5)。

**工作机制：**

1. **数据收集：** 在更新流程结束时 (update\_finish 函数内部)，更新器会收集以下信息：
   * 更新是否成功 (success 布尔值)。
   * 更新过程的总耗时 (elapsed\_time 毫秒)。
   * 客户端标识信息：可能包括主机名 (get\_host\_name)、GUID (get\_guid)、CPUID (get\_cpuid) 或 UUID (get\_uuid)，用于识别发起更新的机器。
   * 应用程序信息：可能包括进程名 (cmdline\_option.old\_version\_process\_name) 和新版本号 (cmdline\_option.latest\_version)。
   * 日志文本 (log\_filetext)：如果 upload\_update\_record\_flag 为 2 或 3，并且日志大小未超过限制。
2. **数据格式化：** 将收集到的信息格式化为特定格式（可能是 JSON、XML 或自定义格式），准备通过 HTTP 发送。
3. **HTTP 请求：** 使用内置的网络库（代码中可能使用了 libcurl 或 Windows Sockets API）向指定的 -server\_address 和 -server\_port 发起 HTTP POST 请求。
   * 请求的 URL 路径可能是一个固定的端点（例如 /api/updatelog），或者在代码中定义。
   * 格式化后的数据作为请求体 (Request Body) 发送。
   * 请求可能包含特定的 HTTP 头（例如 Content-Type）。
4. **服务器响应处理：** 更新器会等待服务器的响应。通常只关心 HTTP 状态码是否表示成功（如 200 OK）。上传操作的结果（成功或失败）会被记录到本地日志文件中。

**服务器端要求：**

您需要在指定的 -server\_address 和 -server\_port 上部署一个 HTTP 服务来接收这些更新记录。该服务需要：

* 监听指定的端口。
* 能够处理来自更新器的 HTTP POST 请求。
* 能够解析请求体中的数据（根据更新器发送的格式）。
* 将接收到的数据存储到数据库、日志文件或其他存储系统中，以供后续分析。
* 返回一个表示接收成功的 HTTP 响应（例如 200 OK）。

**配置建议：**

* **选择合适的 upload\_update\_record\_flag：**
  + 对于大规模部署，建议至少使用 1 来收集基本的成功/失败统计信息。
  + 如果需要详细排查失败原因，使用 2 或 3 上传日志。但要注意日志上传可能带来的网络流量和服务器存储压力。
* **设置合理的 upload\_update\_log\_max\_size：** 根据网络带宽和服务器存储能力设置一个合适的日志大小限制，避免因日志过大导致上传失败或服务器过载。
* **确保服务器可达性：** 确保客户端能够通过网络访问到指定的服务器地址和端口，注意防火墙配置。
* **安全性考虑：**
  + 默认配置下，数据是通过 HTTP 发送的，可能在网络中被窃听。如果传输的数据包含敏感信息，应考虑让服务器支持 HTTPS，并确保更新器使用的网络库支持 HTTPS 连接（版本 1.0.0.2 日志提到修复了 Win7 下的 HTTPS 问题，暗示支持 HTTPS）。
  + 服务器端应考虑实施认证或授权机制，防止未经授权的请求。

## 9.4 静默更新

静默更新是指在用户无感知或干扰最小的情况下完成应用程序的更新。这对于需要保持系统持续运行或避免打扰用户的场景非常重要。更新器通过 -is\_hide\_window 和 -is\_prompt\_if\_error 参数提供了实现静默更新的基础。

**实现静默更新的关键参数：**

1. **-is\_hide\_window true:** 这是实现静默更新的核心参数。它会阻止更新器显示任何窗口。
2. **-is\_prompt\_if\_error false:** 这是实现**完全**静默的关键。默认情况下 (true)，即使窗口隐藏，更新失败时仍会弹出错误消息框。将其设置为 false 可以阻止这个弹窗，使得即使更新失败，用户也不会收到任何直接通知。

**静默更新流程中的注意事项：**

* **触发时机：** 静默更新通常由后台服务、计划任务或应用程序在空闲时自动触发，而不是由用户交互启动。
* **权限问题：** 静默更新通常需要以管理员权限或系统权限运行，以确保能够执行所有必要的操作（终止进程、写入保护目录等）而无需用户 UAC 确认。
* **旧进程处理：** 由于没有用户交互，需要谨慎处理旧进程。通常会使用 -is\_wait\_old\_version\_exit false 配合 -kill\_old\_version\_process\_flag 来强制终止旧进程，或者确保在用户不使用应用程序的时段进行更新。
* **错误处理与监控：** 由于用户不会收到失败通知，依赖日志文件和服务器记录来监控静默更新的成功率和失败原因变得至关重要。
  + 确保日志能够可靠生成（检查权限）。
  + 强烈建议配置服务器记录上传 (-upload\_update\_record\_flag 设为 1, 2 或 3），以便集中监控。
* **更新后行为：** 静默更新后，通常不会自动启动新版本（除非明确配置了 -start\_process\_file\_path），用户将在下次手动启动应用程序时使用新版本。

**完全静默更新配置示例：**

imp\_updater.exe ^  
 -old\_version\_process\_name MyApp.exe ^  
 -source\_path C:\Updates\MyApp\_v2 ^  
 -destination\_path "C:\Program Files\MyApp" ^  
 -latest\_version 2.0.0.0 ^  
 -is\_backup\_file true ^  
 -is\_create\_shortcut false ^  
 -is\_delete\_update\_file true ^  
 -is\_hide\_window true ^  
 -is\_prompt\_if\_error false ^  
 -is\_wait\_old\_version\_exit false ^  
 -kill\_old\_version\_process\_flag 1 ^  
 -upload\_update\_record\_flag 3 ^  
 -server\_address logserver.internal

这个配置将尝试在后台完全静默地更新 MyApp.exe，强制终止旧进程，不创建快捷方式，失败时不弹窗，并将详细结果和日志上传到服务器。

通过掌握这些高级主题，您可以更有效地将更新器集成到复杂的部署场景中，处理特殊情况，并实现更自动化、更可控的软件更新体验。

# 附录A：更新日志

本附录记录了应用程序软件更新器 (imp\_updater.exe) 的主要版本变更历史和修复内容。了解更新历史有助于理解功能的演进和已知问题的修复情况。

**版本：1.0.0.3 (发布日期：2025年2月21日)**

1. **修复：** 解决了在尝试终止进程时，如果 GetLastError 返回 ERROR\_INVALID\_PARAMETER 或 ERROR\_ACCESS\_DENIED 可能导致更新安装失败的问题。增强了进程终止逻辑的健壮性。
2. **优化：** 改进了文件备份逻辑。现在只备份那些在源路径中有对应新版本、确实需要被覆盖的文件，而不是备份整个目标应用程序目录。这显著减少了备份所需的时间和空间，尤其是在应用程序目录包含大量未更改文件时，避免了升级过慢的问题。
3. **优化：** 改进了检查目标文件是否被占用的逻辑。现在仅检查目标路径的直接父目录中的文件，不再递归检查子文件夹中的文件是否被占用。这提高了检查效率，并可能减少因检查范围过广而导致的误判或性能问题。
4. **变更：** 删除了未使用的或已废弃的命令行参数 -backup\_file\_filter\_suffix。

**版本：1.0.0.2 (发布日期：2024年12月31日)**

1. **修复：** 解决了在终止需要更新的主进程后，其加载的某些 DLL 文件仍然被系统或其他进程占用，导致后续复制新版本 DLL 文件失败的 Bug。
2. **优化：** 改进了错误信息的捕获和记录机制，使得日志中的错误信息更准确、更详细。
3. **修复：** 解决了当日志消息中包含百分号 (%) 时，内部日志格式化函数 (\_vsprintf\_p) 可能因格式字符串解析错误而导致更新器程序崩溃的 Bug。
4. **新增：** 增加了更新完成后将更新结果（成功/失败状态）和可选的详细更新日志上传到指定服务器的功能。这有助于集中监控更新部署情况和分析失败原因。
5. **新增：** 增加了动画风格的更新中界面 (-window\_style 1)，用于显示更新进度百分比，提升用户体验。
6. **新增：** 增加了多个命令行参数以支持新功能：
   * -server\_address：指定接收更新记录的服务器地址。
   * -server\_port：指定接收更新记录的服务器端口。
   * -upload\_update\_record\_flag：控制上传记录的行为（是否上传、是否包含日志）。
   * -window\_style：选择更新器窗口样式（0: 标准日志窗口, 1: 动画进度窗口）。
   * -is\_delete\_old\_version\_backup\_file：控制是否在更新成功后删除旧版本的备份文件。
7. **修复：** 解决了在标准窗口模式 (-window\_style 0) 下，窗口创建时 (WM\_CREATE) 可能无法正确显示通过命令行参数传递的信息的 Bug。
8. **新增（内部逻辑）：** 增加了对按时间段推送更新的支持（具体实现细节未在日志中详述，可能需要结合服务器端逻辑）。
9. **修复：** 解决了在 Windows 7 操作系统下，由于系统默认可能不支持 TLS 1.1/1.2 或配置不当，导致向使用 HTTPS 的服务器发送更新记录失败的问题。
10. **优化：** 提高了压缩备份旧文件时的处理速度。
11. **新增（界面）：** 在更新过程中（包括备份和复制文件时）增加了更新进度的计算和显示（对应动画窗口模式下的百分比）。
12. **修复：** 解决了在终止旧进程后，如果操作系统回收进程资源（如句柄）较慢，更新器立即再次尝试检查或操作相关资源可能导致报错的 Bug。
13. **修复：** 解决了更新器内部文件操作逻辑可能无法正确处理超过 2GB 大小文件的 Bug。
14. **优化：** 提高了在不使用压缩备份 (-is\_compress\_backup\_file false) 时，备份大量文件的速度。
15. **依赖更新：** 更新了使用的第三方库版本：libcurl 7.66.0, zlib 1.3.1, libzip 1.11.2。
16. **修复：** 解决了如果在更新器正在运行时，用户再次启动了旧版本的应用程序，可能导致后续复制更新文件时因文件占用而失败的 Bug。
17. **修复：** 解决了在更新完成并尝试删除源更新文件 (-is\_delete\_update\_file true) 时，如果这些文件被其他进程（如杀毒软件扫描）临时占用，可能导致删除失败的 Bug。

**版本：1.0.0.1 (发布日期：2024年4月12日)**

1. **初始版本：** 第一个公开发行的版本。

# 附录B：命令行参数速查表

本附录提供了一个应用程序软件更新器所有命令行参数的快速参考表，便于您查找和使用。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名称 | 注解 (用途) | 值类型 | 默认值 | 章节参考 |
| **核心参数** |  |  |  |  |
| -old\_version\_process\_name | 旧版本进程名 | 字符串 | "" | 5.2 |
| -old\_version\_process\_id | 旧版本进程ID | 整数 (DWORD) | 0 | 5.2 |
| -source\_path | 源路径 (更新文件所在文件夹) | 字符串 (路径) | "" | 5.2 |
| -destination\_path | 目标路径 (应用程序安装文件夹) | 字符串 (路径) | "" | 5.2 |
| -latest\_version | 最新版本号 (用于备份命名等) | 字符串 | "0.0.0.0" | 5.2 |
| **行为控制参数** |  |  |  |  |
| -is\_create\_shortcut | 是否创建快捷方式 | 布尔 (true/false) | true | 5.3 |
| -is\_recursion\_replace\_shortcut | 是否递归替换桌面快捷方式 | 布尔 (true/false) | false | 5.3 |
| -is\_delete\_update\_file | 是否删除源路径更新文件 | 布尔 (true/false) | true | 5.3 |
| -is\_backup\_file | 是否备份旧文件 | 布尔 (true/false) | true | 5.3 |
| -is\_compress\_backup\_file | 是否压缩备份文件 (仅当 -is\_backup\_file true) | 布尔 (true/false) | false | 5.3 |
| -is\_delete\_old\_version\_backup\_file | 是否删除旧版本备份文件 | 布尔 (true/false) | false | 5.3 |
| -is\_wait\_old\_version\_exit | 是否等待旧进程退出 (而不是直接终止) | 布尔 (true/false) | false | 5.3 |
| -wait\_old\_version\_exit\_timeout | 等待旧进程退出超时时间 (毫秒) | 整数 | 3000 (3秒) | 5.3 |
| -kill\_old\_version\_process\_flag | 终止旧进程方式 (0: 按ID, 1: 按名称) | 整数 (0 或 1) | 0 | 5.3 |
| -start\_process\_file\_path | 更新后/仅启动模式下要启动的程序路径 | 字符串 (路径) | "" | 5.3 |
| -start\_process\_command\_line | 启动新进程时传递的命令行参数 | 字符串 | "" | 5.3 |
| -start\_process\_working\_directory | 启动新进程时的工作目录 | 字符串 (路径) | "" | 5.3 |
| -is\_only\_start\_process | 是否仅启动进程 (跳过更新流程) | 布尔 (true/false) | false | 5.3 |
| **界面与交互参数** |  |  |  |  |
| -is\_hide\_window | 是否隐藏窗口 (静默模式) | 布尔 (true/false) | false | 5.4 |
| -is\_show\_topmost | 是否将窗口置顶 | 布尔 (true/false) | false | 5.4 |
| -window\_title | 窗口标题 (仅当 -window\_style 0) | 字符串 | (来自资源文件) | 5.4 |
| -window\_width | 窗口宽度 (像素, 仅当 -window\_style 0) | 整数 | 500 | 5.4 |
| -window\_height | 窗口高度 (像素, 仅当 -window\_style 0) | 整数 | 350 | 5.4 |
| -window\_style | 窗口样式 (0: 标准日志, 1: 动画进度) | 整数 (0 或 1) | 1 | 5.4 |
| -is\_prompt\_if\_error | 错误时是否弹窗提示 (隐藏/动画模式下) | 布尔 (true/false) | true | 5.4 |
| **高级参数** |  |  |  |  |
| -sensitive\_dll | 敏感DLL列表 (分号分隔, 用于尝试绕过拦截) | 字符串 | "" | 5.5, 9.1 |
| -server\_address | 上传更新记录的服务器地址 | 字符串 | "www.invomes.com" | 5.5, 9.3 |
| -server\_port | 上传更新记录的服务器端口 | 整数 | 60682 | 5.5, 9.3 |
| -upload\_update\_record\_flag | 上传记录标志 (0:不传, 1:结果, 2:失败传日志, 3:都传) | 整数 (0-3) | 1 | 5.5, 9.3 |
| -upload\_update\_log\_max\_size | 上传日志最大大小 (MB) | 整数 | 5 | 5.5, 9.3 |
| **辅助命令** | (通常单独使用) |  |  |  |
| -help | 显示帮助信息 | N/A | N/A | 5.6 |
| -option | 显示所有参数及注解 | N/A | N/A | 5.6 |
| -default\_sensitive\_dll | 显示内部默认敏感DLL列表 | N/A | N/A | 5.6 |
| -delete\_all\_backup\_file | 删除所有备份文件 | N/A | N/A | 5.6 |

**注意:** \* 参数值包含空格时，请使用双引号括起来。 \* 布尔参数建议明确使用 true 或 false。 \* 参数名称通常是大小写敏感的，请按照表格中的格式使用。 \* “章节参考”指向本手册中对该参数进行详细解释的章节编号。

# 附录C：术语表

本术语表解释了在本手册中使用的特定术语和缩写，以帮助读者更好地理解相关概念。

* **API (Application Programming Interface)** 应用程序编程接口。一组定义软件组件之间如何交互的规则和协议。例如，更新器使用 Windows API 来执行文件操作和进程管理。
* **备份 (Backup)** 在执行可能导致数据丢失或更改的操作（如软件更新）之前，复制并存储原始文件或数据的过程。用于在发生问题时恢复到先前的状态。
* **命令行参数 (Command Line Argument/Parameter)** 在通过命令行启动程序时，附加在程序名称后面的文本字符串，用于向程序传递信息或指令。更新器通过命令行参数接收其所有配置。
* **DLL (Dynamic Link Library)** 动态链接库。Windows 操作系统中的一种文件格式，包含可被多个程序同时使用的代码和数据。应用程序在运行时加载所需的 DLL。
* **DNS (Domain Name System)** 域名系统。互联网的一项服务，用于将人类可读的域名（如 www.example.com）转换为机器可读的 IP 地址（如 192.168.1.1）。
* **GUI (Graphical User Interface)** 图形用户界面。允许用户通过图形图标、视觉指示器和指向设备（如鼠标）与电子设备进行交互的用户界面类型。更新器的标准窗口和动画窗口属于 GUI。
* **GUID (Globally Unique Identifier)** 全局唯一标识符。一个 128 位的数字，用于在计算中产生唯一的引用标识符。更新器可能使用 GUID 来标识客户端机器。
* **HTTP (Hypertext Transfer Protocol)** 超文本传输协议。用于在万维网上传输文件的应用层协议，是客户端（如浏览器或更新器）和服务器之间通信的基础。
* **HTTPS (Hypertext Transfer Protocol Secure)** 安全超文本传输协议。HTTP 的安全版本，通过 SSL/TLS 协议对通信进行加密，确保数据传输的机密性和完整性。
* **进程 (Process)** 操作系统中正在运行的程序的实例。每个进程都有自己独立的内存空间和系统资源。
* **进程 ID (Process ID / PID)** 操作系统为每个正在运行的进程分配的唯一数字标识符。
* **权限 (Permissions)** 控制用户或进程可以对文件、目录或其他系统资源执行哪些操作（如读取、写入、执行）的规则。
* **快捷方式 (Shortcut)** 指向文件、文件夹或应用程序的链接文件（通常是 .lnk 文件），允许用户快速访问目标项。
* **静默模式 (Silent Mode)** 软件安装或更新的一种模式，在后台运行，不显示任何用户界面，也不需要用户交互。
* **路径 (Path)** 在文件系统中唯一标识文件或目录位置的字符串。可以是绝对路径（从根目录开始）或相对路径（相对于当前工作目录）。
* **日志 (Log)** 按时间顺序记录系统或应用程序运行期间发生的事件、操作、错误或状态信息的文件或数据流。
* **目标路径 (Destination Path)** 在文件复制或软件更新操作中，文件将被复制到或应用程序将被安装到的目录。
* **源路径 (Source Path)** 在文件复制或软件更新操作中，包含要复制或安装的原始文件的目录。
* **TLS (Transport Layer Security)** 传输层安全性协议。一种加密协议，用于在计算机网络上提供通信安全，是 HTTPS 的基础。取代了早期的 SSL (Secure Sockets Layer)。
* **UAC (User Account Control)** 用户帐户控制。Windows Vista 及更高版本中的一项安全功能，有助于防止未经授权的更改。当程序尝试执行需要管理员权限的操作时，会弹出提示要求用户确认。
* **UI (User Interface)** 用户界面。用户与计算机、软件或设备交互的方式、手段或界面。
* **UUID (Universally Unique Identifier)** 通用唯一标识符。与 GUID 类似，是一个旨在全球范围内唯一的标识符。更新器可能使用 UUID 来标识客户端机器。
* **工作目录 (Working Directory)** 进程当前操作的默认目录。当程序引用没有指定完整路径的文件时，通常会在工作目录中查找。