Windows Updates using forward and reverse differentials

A technique to produce compact software updates optimized for any origin/destination revision pair

## Microsoft

Windows 10 monthly Quality Updates are cumulative, containing all previously released fixes to ensure consistency and simplicity. For an OS platform like Windows, which stays in support for multiple years, the size of monthly quality updates can quickly grow large, thus directly impacting network bandwidth consumption.

Today, this problem is addressed by utilizing express downloads, where differential downloads for every changed file in the update are generated based on selected historical revisions plus the base version. In this paper, we introduce a new technique to build compact software update packages that are applicable to any base revision, and then describe how Windows 10 Quality Updates utilize this technique.

### General Terms

Base version: A major software release with significant changes e.g. Windows 10 October 2018 Update (OS Build 17763.1)

Revision: Minor releases in between the major version releases e.g. KB4464330 (OS Build 17763.55)

Baseless Patch Storage Files (Baseless PSF): PSF that contains full binaries or files

## Introduction

In this paper, we introduce a new technique that can produce compact software updates optimized for any origin/destination revision pair. It does this by calculating forward the differential of a changed file from the base version and its reverse differential back to the base version. Both forward and reverse differentials are then packaged as an update and distributed to the endpoints running the software to be updated.

∆0→N

∆N→0

#### Figure 1: Update package contents

The endpoints that have the base version of the file (V0) hydrate the target revision (VN) by applying a simple transformation:

V0 + ∆0→N = VN

The endpoints that have revision N of the file (VN), hydrate the target revision (VR) by applying the following set of transformations:

VN + ∆N→0 = V0

V0 + ∆0→R = VR

The endpoints retain the reverse differentials for the software revision they are on, so that it can be used for hydrating and applying next revision update.

By using a common baseline, this technique produces a single update package with numerous advantages:

* Compact in size
* Applicable to all baselines
* Simple to build
* Efficient to install
* Redistributable

Historically, download sizes of Windows 10 Quality Updates (Windows 10 1803 and down-level supported versions of Windows 10) are optimized by using express download. Express download is optimized such that updating Windows 10 systems will download the minimum number of bytes. This is achieved by generating differentials for every updated file based on selected historical base revisions of the same file + it’s base/RTM version.

For example, if the Oct monthly Quality Update has updated notepad.exe, differentials for notepad.exe file changes from Sept to Oct, Aug to Oct, July to Oct, June to Oct and from the original feature release to Oct are generated. All these differentials are stored in a Patch Storage File (PSF, also referred to as express download files) and hosted/cached on Windows Update (WU) or other update management/distribution servers (e.g. Windows Server Update Services (WSUS), System Center Configuration Manager, or a third-party update manager/distribution server that supports express updates). A device leveraging express updates uses network protocol to determine optimal differentials, then download only what is needed from the update distribution endpoints.

The flipside of express download is that the size of PSF files can be very large depending on the number of historical baselines against which differentials were calculated. Downloading and caching large PSF files to on-premise or remote update distribution servers is problematic for most organizations and hence they are unable to leverage express updates to keep their fleet of Windows systems 10 up-to-date. Secondly, due to the complexity of generating differentials and size of the express files that need to be cached on update distribution servers, it is only feasible to generate express download files for the most common baselines, thus express updates are only applicable to selected baselines. Finally, calculation of optimal differentials is expensive in terms of system memory utilization, especially for low cost systems impacting their ability to download and apply an update seamlessly.

In the following sections, we describe how Windows 10 Quality Updates will leverage this technique based on forward and reverse differentials for newer releases of Windows 10 and Windows Server to overcome the challenges with express.

## High-level Design

### Update Packaging

### Windows 10 Quality Update packages will contain forward differentials from Feature Update RTM baselines (∆RTM→N) and reverse differentials back to RTM (∆N→RTM) for each file that has changed since RTM. By using RTM as the baseline, we ensure that all devices will have an identical payload. Update package metadata, content manifests, and forward/reverse differentials will be packaged into a Cabinet file (.cab). This .cab, and the applicability logic, will also be wrapped in Microsoft Standalone Update (.msu) format.

There can be cases where new files are added to the system during servicing. These files will not have RTM baselines, thus forward and reverse differentials cannot be used. In these scenarios, null differentials will be used to handle servicing. Null differentials are the slightly compressed and optimized version of the full binaries. It should be noted that update packages can have either forward or reverse differentials, or null differential of any given binary in them.

.msu

Applicability Logic

.cab

Update Metadata

Content Manifests

∆RTM→N (File1, File2,..)

∆N→RTM (File1, File2,..)

#### Figure 2. Windows 10 Quality Update Standalone Installer

### Hydration and Installation

Once the usual applicability checks are performed on the update package and is determined to be applicable, Windows Component Servicing Infrastructure (CSI) will hydrate the full files during pre-installation and then proceed with the usual installation process.

Below is a high-level sequence of activities that CSI will run in a transaction to complete installation of the update:

* Identify all files that are required to install the update.
* Hydrate each of necessary files using current version (VN) of the file, reverse differential (VN->RTM) of the file back to Feature Update RTM/base version and forward differential (VRTM->R) from Feature Update RTM/base version to the target version. Also, use null differential hydration to hydrate null compressed files.
* Stage the hydrated files (full file), forward differentials (under ‘f’ folder) and reverse differentials (under ‘r’ folder) or null compressed files (under ‘n’ folder) in the Component Store (%windir%\WinSxS folder).
* Resolve any dependencies and install components.
* Cleanup older state (VN-1), previous state VN is retained for uninstall and restore/repair.

### Resilient Hydration

To ensure resiliency against Component Store corruption or missing files that could occur due to susceptibility of certain types of hardware to file system corruption, Corruption Repair service has been traditionally used to recover the Component Store automatically (Automatic Corruption Repair (ACR)) or on demand (Manual Corruption Repair (MSR)) using an online or local repair source. This service will continue to offer the ability to repair and recover content for hydration and successfully install an update, if at all needed.

When corruption is detected during update operations, ACR will kick in as usual and use Baseless Patch Storage File (Baseless PSF) published to WU for each update to fix corrupted manifests, binary differentials or hydrated/full files. Baseless PSF will contain reverse and forward differentials, and full files for each updated component. Integrity of the repair files will be hash verified.

Corruption Repair will use the component manifest to detect missing files and get hashes for corruption detection. During update installation, new registry flags for each differential staged on the machine will be set. When ACR runs, it will scan hydrated files using the manifest and differential files using the flags. If the differential cannot be found or verified, it will be added to the list of corruptions to repair.

### Lazy ACR

Lazy ACR runs during update operation to detect corrupted binaries and differentials. While applying an update, if hydration of any file fails, Lazy ACR automatically kicks in and figures out the corrupted binary or differential, and then adds it to the corruption list. Later, update operation continues as far as it can go, so that Lazy ACR can collect as many corrupted files to fix. At the end of the hydration section, update fails and ACR kicks in. ACR runs as usual and at the end of its operation, adds Lazy ACR’s corruption list on top of the new list to repair. ACR will repair the files on the corruption list and installation of the update will succeed on the next attempt.