An examination of Win10 (version 1803) ActivitiesCache.db database

Windows Timeline, is a new feature of Windows 10 introduced with version 1803.
It is part of the Connected Devices Platform

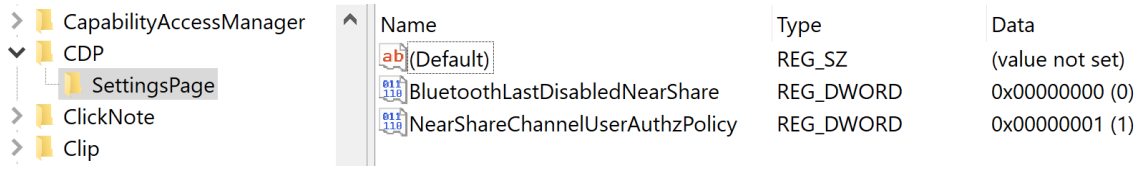
- *The Connected Devices Platform Service, is a Windows service that provides a way for devices such as PC's and smartphones to discover and send messages between each other.*
- *Connected Devices Platform Service (CDPSvc) Defaults in Windows 10.*

and the Microsoft Graph's Cross-device experience (Project Rome) .

The CDP settings for the Current User are stored in the registry at:
NTUSER.DAT -> 'Software\Microsoft\Windows\CurrentVersion\CDP'

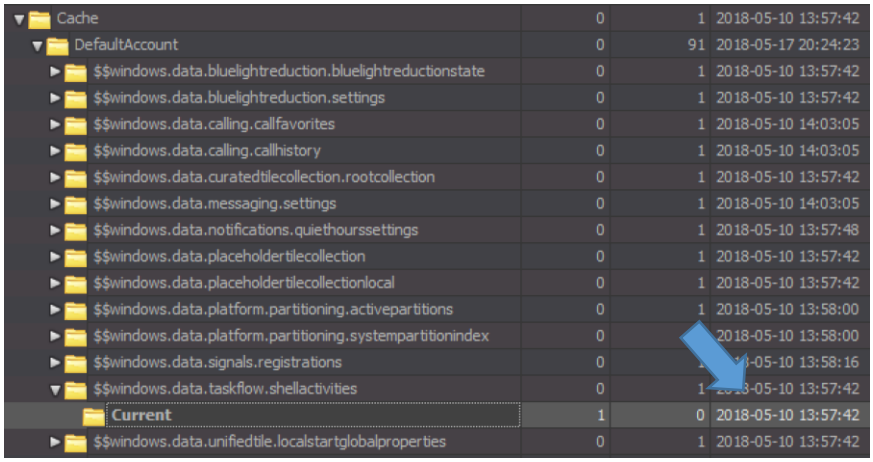
Name	Type	Data
(Default)	REG_SZ	(value not set)
CdpSessionUserAuthzPolicy	REG_DWORD	0x00000001 (1)
CdpUserSettingsVersion	REG_SZ	RS4
EnableRemoteLaunchToast	REG_DWORD	0x00000001 (1)
NearShareChannelUserAuthzPolicy	REG_DWORD	0x00000000 (0)
RomeSdkChannelUserAuthzPolicy	REG_DWORD	0x00000001 (1)

and

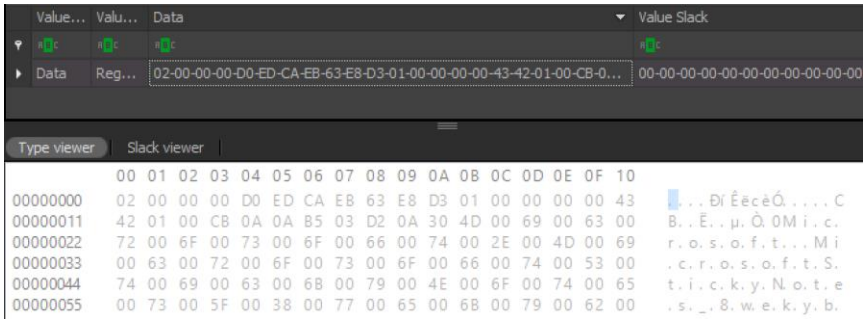


However, the service and the 'ActivitiesCache.db' database existed before the 1803 upgrade (May 2018), but with limited functionality. Another possibly (not sure yet) related activity store location is at:

'*Software\Microsoft\Windows\CurrentVersion\CloudStore\Store\Cache\DefaultAccount\$$windows.data.taskflow.shellactivities\Current*'



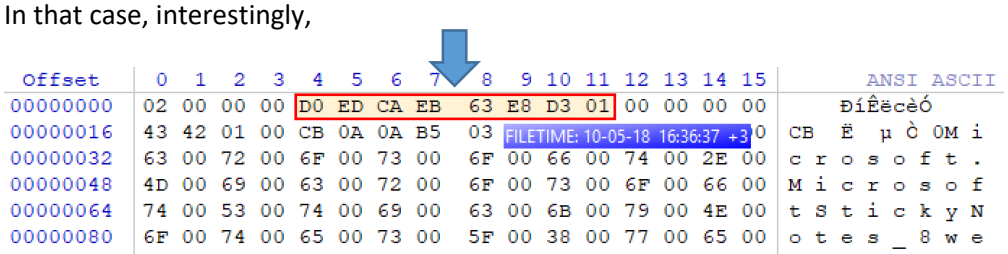
where there is a value named ‘Data’:



Its value is in hex and it seems to hold interesting information, including the Filetime of last update (which corresponds to the Last Time Timestamp of the registry key).

If Windows was updated to version 1803, this log stops being updated. This can be checked by looking in the SYSTEM hive at the Setup key like:

In that case, interestingly,

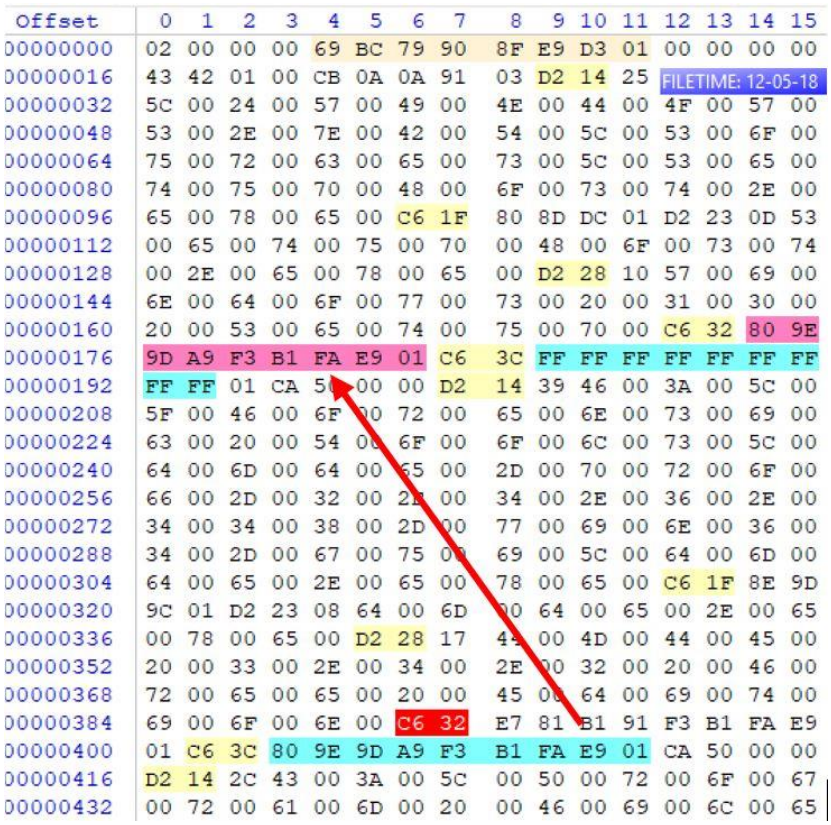


this Filetime is very close to the date of the ntuser.dat.LOG files (*which coincides with the date the 1803 update occurred*), and that can also be seen from the last entry above:

Further examination shows a consistent pattern:

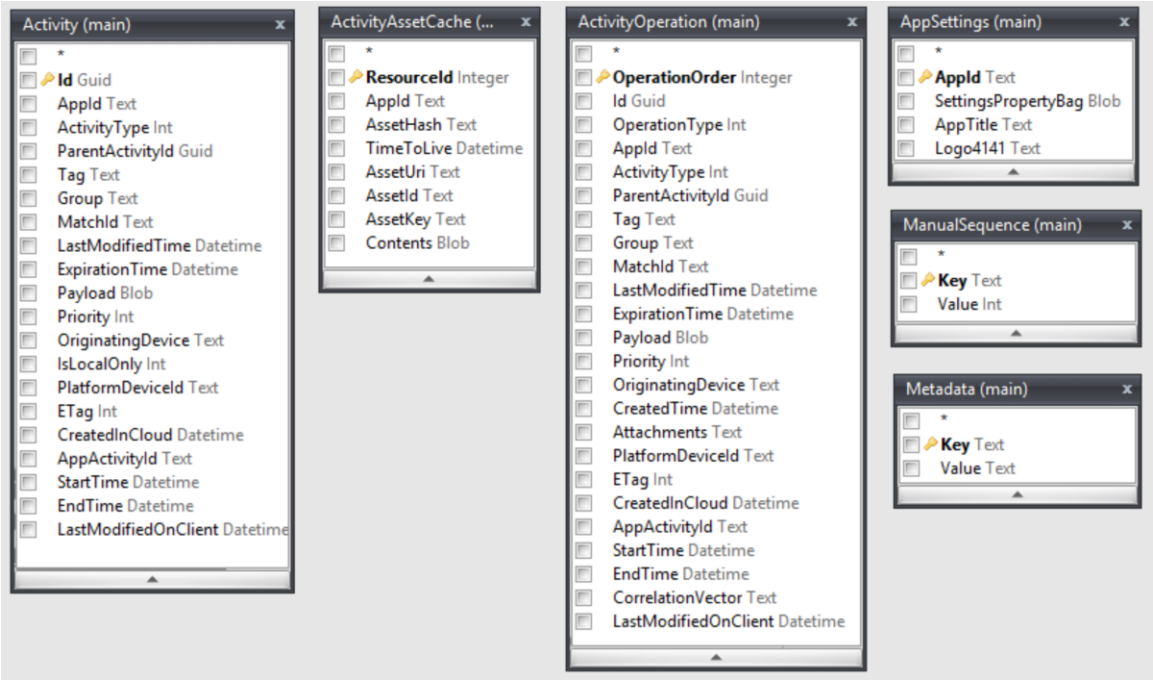
- 0xD2 14 = Start of Entry
- Next byte = length of block (x2)
- Start of path & executable
- 0xC6 1F = End of block
- Next 4 bytes = unknown
- 0xD2 23 = Executable Block
- Next byte = length of block (x2)
- Executable
- 0xD2 28 = Payload block
- Next byte = length of block (x2)
- Payload (eg email, URL etc.)
- 0xC6 32 = end of block
- Next 9 bytes = (A) is the same as (B) of the next entry (upwards)
- 0xC6 3C = Pointer to next entry
- Next 9 bytes = (B) is the same as (A) of the next entry (upwards) *
- 0x CA500000 = End of Entry

**Top most entry is the newest one, and these 9 bytes are all 0xFF*



With a bit of tweaking in Notepad++, it shows web page titles, email (used in accounts of Outlook), File Explorer paths followed and name of the remote devices accessed with Teamviewer among other.

Back to the 'ActivitiesCache.db' database. The old dB table structure was similar to the new dB, but it included 6 tables + the master table, with different fields:



The information held was also different:

E.g. the 'AppSettings' table in the old DB looked like this:

Table: AppSettings

	AppId	SettingsPropertyBag	AppTitle	Logo4141
	Filter	Filter	Filter	Filter
1	windows.immersivecontrolpanel_cw5n1h2txyewy!microsoft.windows.immersivecontrolpanel	*BLOB*	*NULL*	*NULL*
2	Microsoft.SkypeApp_kzf8qxf38zg5c!ppleae38af2e007f4358a809ac99a64a67c1	*BLOB*		*NULL*
3	Microsoft.SkypeApp_kzf8qxf38zg5c!App	*BLOB*	*NULL*	*NULL*
4	Microsoft.MicrosoftEdge_8wekyb3d8bbwe!MicrosoftEdge	*BLOB*	*NULL*	*NULL*
5	AD2F1837.HPPrinterControl_v10z8vjag6ke6!AD2F1837.HPPrinterControl	*BLOB*	*NULL*	*NULL*
6	Microsoft.BingNews_8wekyb3d8bbwe!AppexNews	*BLOB*	*NULL*	*NULL*
7	Microsoft.WindowsCamera_8wekyb3d8bbwe!App	*BLOB*	*NULL*	*NULL*
8	Microsoft.Windows.Photos_8wekyb3d8bbwe!App	*BLOB*	*NULL*	*NULL*
9	VirtualPulse.PlayerforMedia_nh7p8cqfc4t04!App	*BLOB*	*NULL*	*NULL*

And included UWP Apps only. The new DB does not have any entries in this table.

The 'Metadata' table was also different:

Table: Metadata

	Key	Value
	Filter	Filter
1	CurrentEtag	5b15de40-92c2-11e7-9117-01020305070d
2	CurrentSettings	{"ActivityTypes":[4,3,2,1,0],"Environment":"prod"}
3	PendingStrongAuth	{"pendingStrongAuth":true}

The new DB has different fields:

Table: Metadata

	Key	Value
	Filter	Filter
1	CurrentSettings	{"ActivityTypes":[4,3,2,1,0,5,13,6],"Environment":"prod"}
2	DatabaseInstanceId	38608
3	DatabaseInstanceIdUpdateTime	2018-05-10T14:00:55.845Z

After examination of various new 'ActivitiesCache.db' files, any entries seen in the new database have an 'ActivityType' (*seen above*) of 5 or 6, which appear to represent '*Open App/File/Url*' and '*App In Focus*' respectively – type 13 was not observed in any dB yet.
The 'DatabaseInstanceIdUpdateTime' date seen above, is the date & time (in UTC) that the current dB file was created.

A Microsoft Account related DB includes one more field, the 'CurrentEtag' but instead of ETAG it lists a GUID:

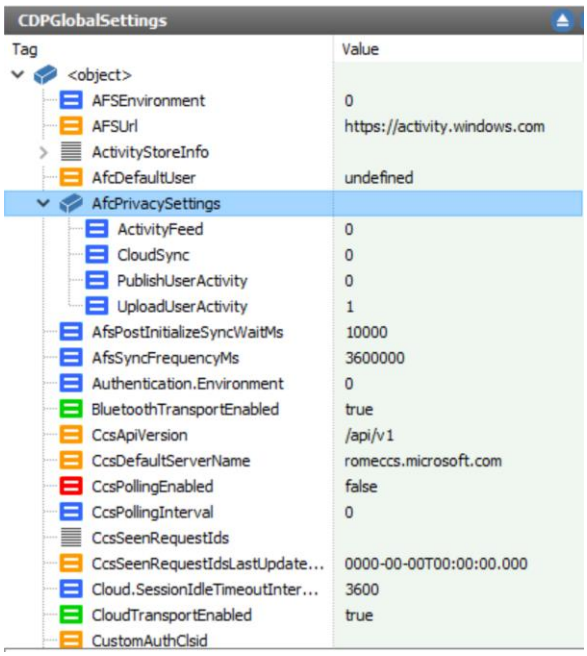
	Key	Value
	Filter	Filter
1	CurrentSettings	{"ActivityTypes":[4,3,2,1,0,5,13,6],"Environ...
2	DatabaseInstanceId	23919
3	DatabaseInstanceIdUpdateTime	2018-05-10T13:58:59.758Z
4	CurrentEtag	8df03b30-642c-11e8-9117-01020305070d

The ‘SmartLookUp’ view in the old DB was also different:

```
select
       [O].[Id],
       [O].[AppId],
       [O].[AppActivityId],
       [O].[ActivityType],
       [O].[ParentActivityId],
       [O].[Tag],
       [O].[Group],
       [O].[MatchId],
       [O].[LastModifiedTime],
       [O].[ExpirationTime],
       [O].[Payload],
       [O].[Priority],
       [O].[OriginatingDevice],
       [A].[IsLocalOnly],
       [A].[PlatformDeviceId],
       [A].[CreatedInCloud],
       [O].[StartTime],
       [O].[EndTime],
       [O].[LastModifiedOnClient],
       [O].[ETag]
from   [ActivityOperation] as [O]
       left outer join [Activity] as [A] on [O].[Id] = [A].[Id]
where  [O].[OperationType] <> 3
union
select
       [Id],
       [AppId],
       [AppActivityId],
       [ActivityType],
       [ParentActivityId],
       [Tag],
       [Group],
       [MatchId],
       [LastModifiedTime],
       [ExpirationTime],
       [Payload],
       [Priority],
       [OriginatingDevice],
       [IsLocalOnly],
       [PlatformDeviceId],
       [CreatedInCloud],
       [StartTime],
       [EndTime],
       [LastModifiedOnClient],
       [ETag]
from   [Activity]
where  [Id] not in (select [Id]
       from   [ActivityOperation])
```

The NEW (updated) database stores information for each user in the %LOCALAPPDATA% \ConnectedDevicesPlatform folder or more commonly familiar as: C:\users\username\appdata\local\ConnectedDevicesPlatform

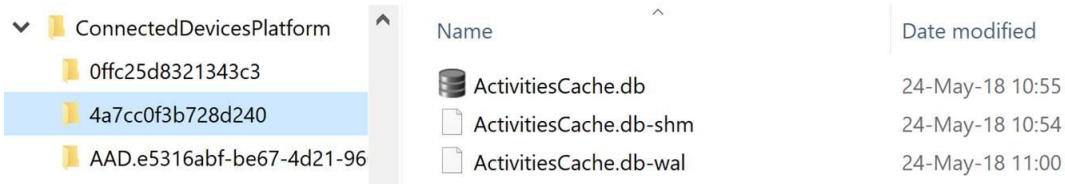
This folder has a file named ‘CDPGlobalSettings.cdp’ which is essentially a .json file



and includes information for the service including the store location for the database that holds the windows timeline entries for the current user.

The folder name may include one or more of:

- A Local account starting with an L.UserName, and/or
- folder(s) which appear to be name with random numbers but are the cid's of Microsoft accounts (e.g. Outlook.com, Live.com, Hotmail.com or MSN.com domains).



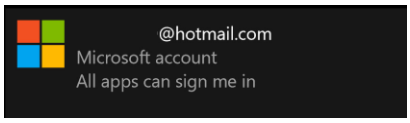
If a user logs in to Windows with a MS account, then the respective dB is populated.
Information as to which online account has this specific ID can be found at:
NTUSER.DAT -> '*Software\Microsoft\IdentityCRL\UserExtendedProperties*'
Under each online account (email) the 'cid' Value contains the respective value for each Microsoft Account & folder.

If a user logs in with a local account, then the L.UserName database is populated.

A Microsoft Exchange (Office365 etc) account starts with 'AAD' and is in the form of AAD.[sid]. AAD denotes a business account (as in Azure Active Directory) and information associated with this account 's [sid] can be found in NTUSER.dat at -> '*Software\Microsoft\MSOIdentityCRL\UserExtendedProperties'* and/or '*Software\Microsoft\OneDrive\Accounts\Business1'*

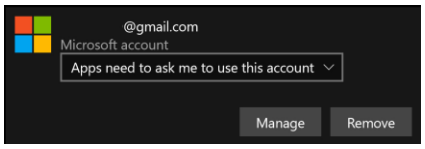
However, there are two cases when, even if a user continues to use a local account to sign in Windows 10 , the activities can still be synchronized across devices:

1. The User has set up a Microsoft Account at "Settings-> Accounts->Email&app accounts" and set it up so that all apps can sign in with this account:



   In this case the L.UserName folder is removed, and the active folder is the folder that corresponds to this account. All activities are synched across devices.

2. The User has set up a Microsoft Account at "Settings-> Accounts->Email&app accounts" and set it up so that Apps need permission to use this account:



   In this case, the local activities are stored in the L.Username folder, but any activity on other devices is stored in the respective folder that corresponds to this MS account.

The timeline settings ('Settings –> Privacy –> Activity History') enable or not activity to be tracked in the 'ActivitiesCache.db'. In order for the activity to be shared across devices (cloud) you need to login with a Microsoft account, enable Windows Hello (i.e. create pin and or biometric/photo login).
https://support.microsoft.com/en-us/help/4026102/windows-10-about-sync-settings

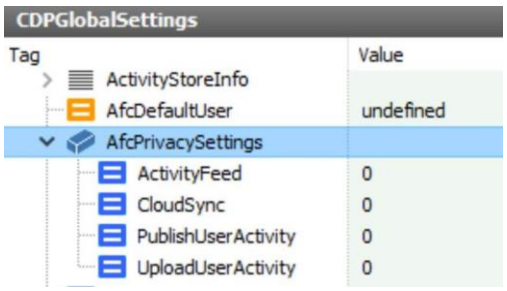
To enable cross-device experiences, your app users must login with either a Microsoft Account or an Azure Active Directory account.
https://github.com/Microsoft/project-rome/blob/master/cross-device_app_configuration.md

https://github.com/microsoftgraph/microsoft-graph-docs/blob/master/api-reference/beta/resources/intune_deviceconfig_windows10generalconfiguration.md

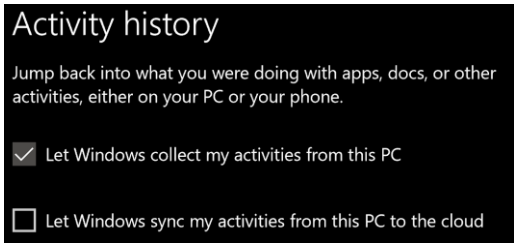
A list of settings that can be roamed or backed up (synced) across devices can be found here: https://docs.microsoft.com/en-us/azure/active-directory/active-directory-windows-enterprise-state-roaming-windows-settings-reference

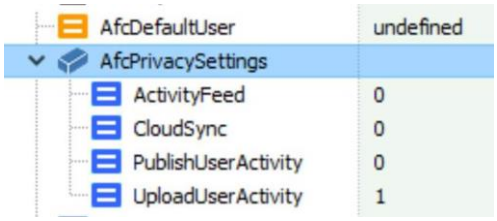
The 'AfcPrivacySettings' in 'CDPGlobalSettings.cdp' depicts the relative settings in Windows 10 'Settings –> Privacy- >Activity History', for example:

All settings ON:

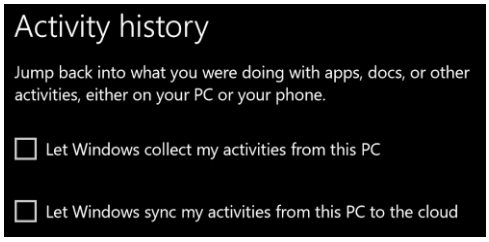


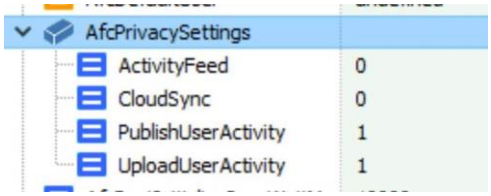
Upload to cloud off



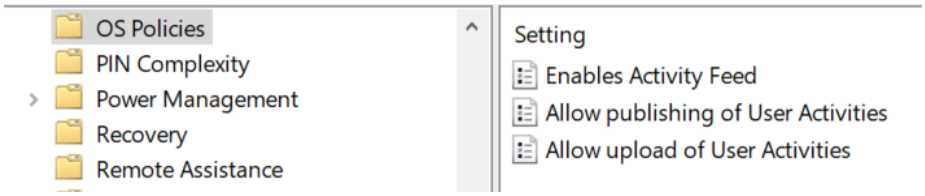


Publish off, cloud off



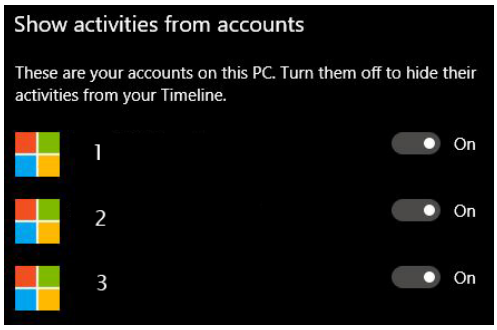


The same settings can also be configured through Group Policy at 'Computer Configuration -> Administrative Templates -> System ->OS Policies'



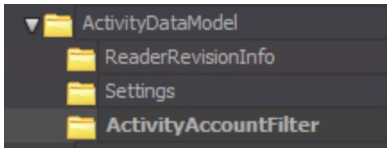
Opening the respective folder we can see the 'ActivitiesCache.db' (*and possibly also the ActivitiesCache.wal and ActivitiesCache.db-sdb files*) which is the storage of all timeline activity for the current user. When a user switches from a Local to a MS account, the 'L.UserName' is deleted and a new one is created (if it didn't exist before) under the respective MS account folder. Any previous entries in the 'L.UserName''s 'ActivitiesCache.db' now appear in the new 'ActivitiesCache.db' linked to the MS account, and sync to the cloud. Switching from a MS to a local account, the MS account related db is still available and holds timeline data.

When a user logs in Windows with a MS account, and the relevant check boxes in 'Settings –> Privacy- >Activity History' are set to On,



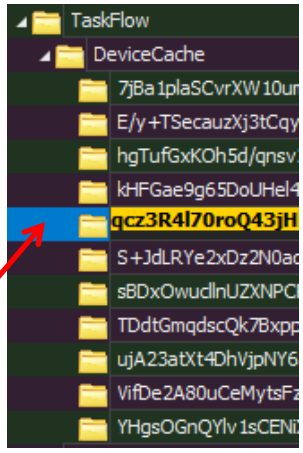
any activity created with his other MS account(s), is updated in the 'ActivitiesCache.db' in the respective '%LOCALAPPDATA% \ConnectedDevicesPlatform\' Folder, and can be seen in his timeline view .

Any excluded from sync accounts (above setting set to off) can be seen at the registry at: NTUSER.dat Software\Microsoft\Windows\CurrentVersion\ActivityDataModel\ActivityAccountFilter



The synchronized entries in a MS account related dB will show the 'Device ID' of the machine the entry originated from. The Device Name and Model of the originating machine can be seen in the NTUSER.dat's '*Software\Microsoft\Windows\CurrentVersion\TaskFlow\DeviceCache*':

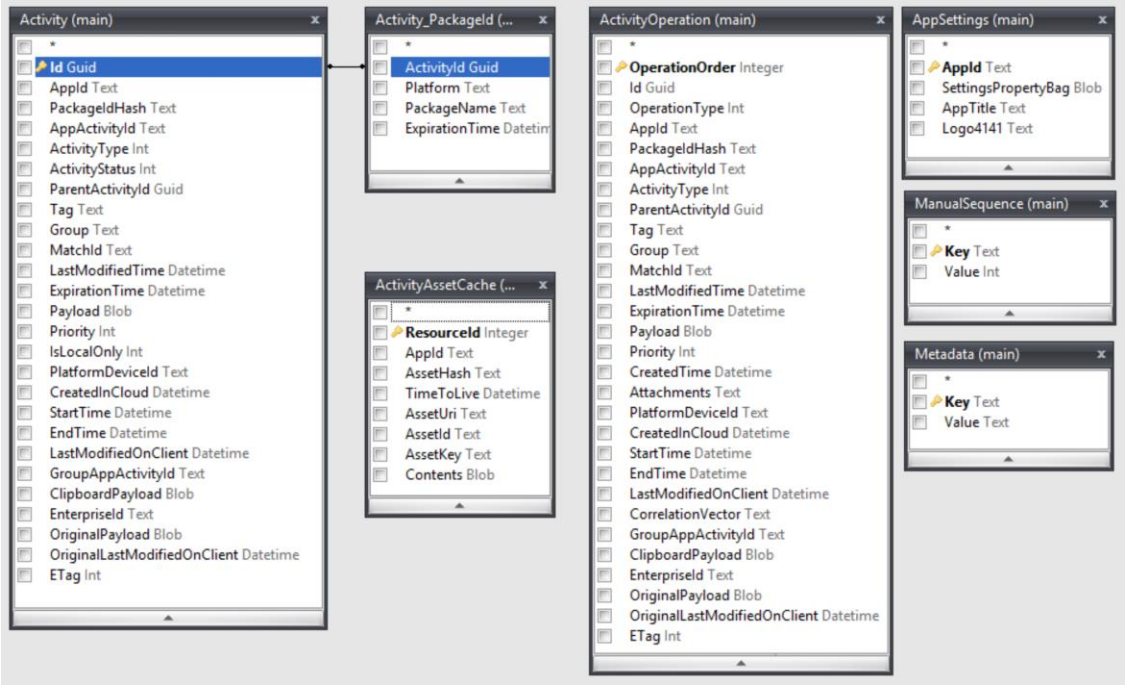
Device ID
qcz3R4I70roQ43jHkxaTglaAC2eQunmzl6IKwNi+KOc=
qcz3R4I70roQ43jHkxaTglaAC2eQunmzl6IKwNi+KOc=
qcz3R4I70roQ43jHkxaTglaAC2eQunmzl6IKwNi+KOc=



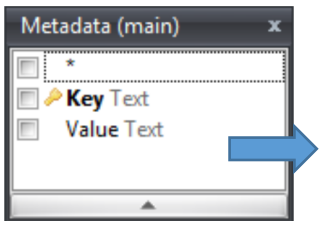
Even if a user logs in with a local account, when a MS Account was previously used for login, the entries of 'ActivitiesCache.db' of that account will also appear in the user's timeline (for the previous 7 days). So it may be prudent to examine all 'ActivitiesCache.db' databases in a system.

Following is the analysis of the new 'ActivitiesCache.db' file, using DB Browser for SQLite , SQLite Expert Professional, Notepad++ and JSON Viewer.

The database has the following structure (7 tables + the master table):



The 'Metadata' table includes the date/time the database was created by the system:



	Key	Value
	Filter	Filter
1	CurrentSettings	{"ActivityTypes":[4,3,2,1,0,5,13,6],"Environment":"prod"}
2	DatabaseInstanceId	3428
3	DatabaseInstanceIdUpdateTime	2018-05-24T08:16:50.365Z

Note: *see page 2 above for the differences with the old database*

The 'ManualSequence' table shows the last Activity (ETAG) recorded in the database:

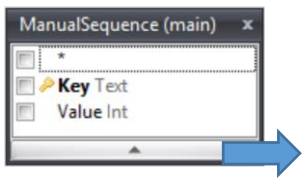
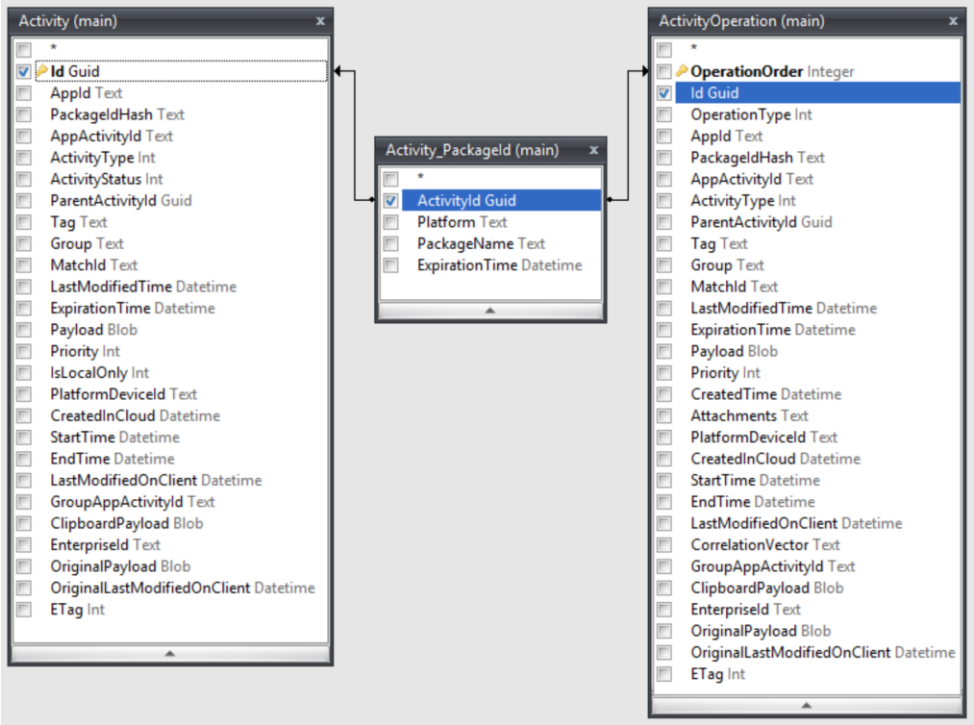


Table: ManualSequence

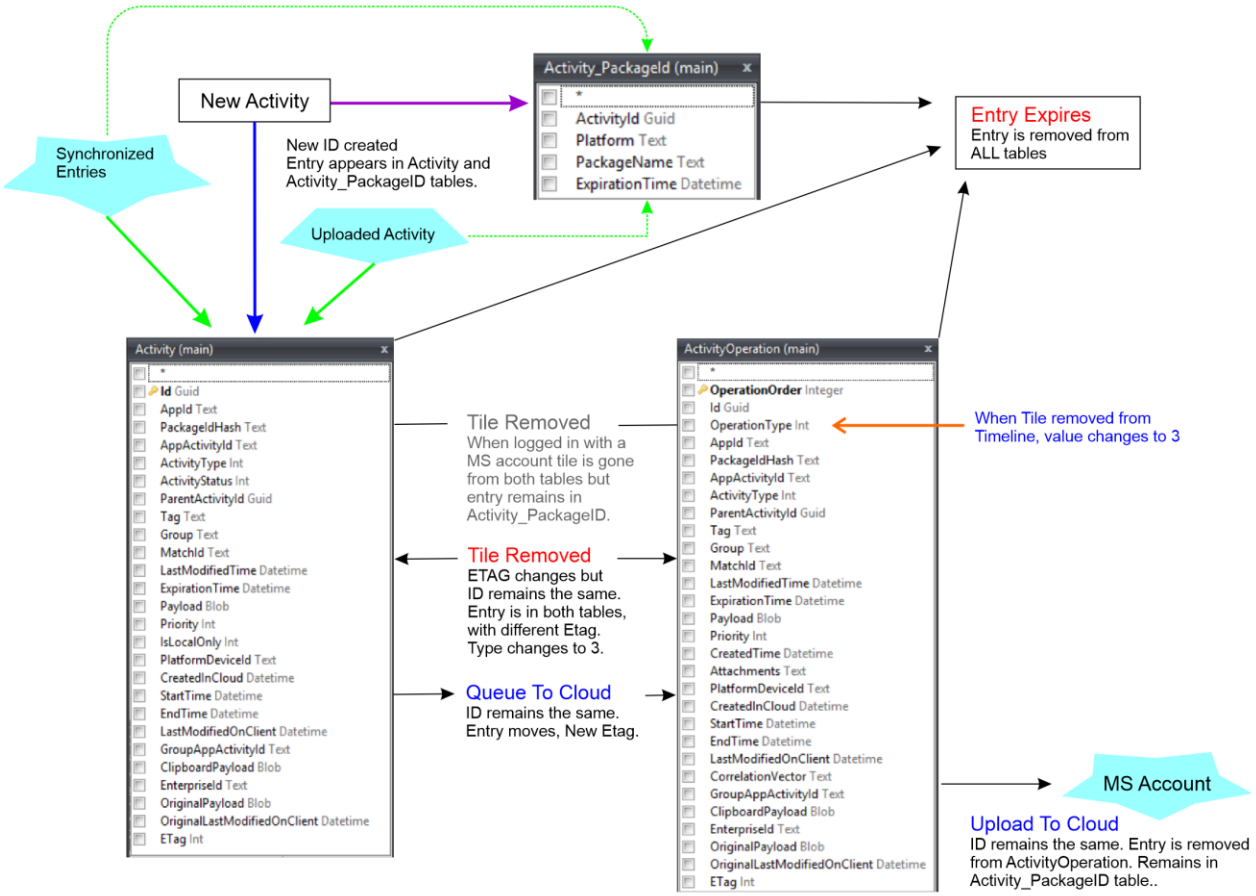
	Key	Value
	Filter	Filter
1	Activity	839

From the 7 tables above, only three hold information of user activities:

'Activity', 'ActivityOperation' and 'Activity_PackageID'



The Windows Timeline process looks like this:

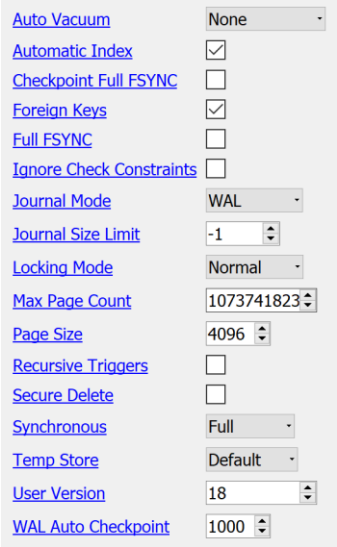


Any new application execution or focus change, triggers an entry to the 'Activity' table, and relevant entries in the 'Activity_PackageID' table.

Note: *It should be noted that any new entries/transactions are written first to the write ahead log ('ActivitiesCache.wal')*

Name	Date modified	Type	Size
ActivitiesCache.db	27-May-18 5:20 pm	DB File	4,196 KB
ActivitiesCache.db-shm	27-May-18 4:07 pm	DB-SHM File	32 KB
ActivitiesCache.db-wal	27-May-18 5:22 pm	DB-WAL File	528 KB

and the database is updated when the wal file reaches 1000 pages and commits the transactions. The default settings of the database show that each page is 4096 bytes long:



The query below (*which is in essence two united queries*) extracts all possible useful information from this database in a readable form. This query will not work properly in any SQLite browser that does not include support for the json1 extension.

```sql
SELECT -- This the ActivityOperation Table Query
    ActivityOperation.ETag as 'Etag',
    json_extract(ActivityOperation.Payload, '$.appDisplayName') as 'Program Name',
    case
        when json_extract(ActivityOperation.AppId, '$[0].application') = '308046B0AF4A39CB'
            then 'Firefox-308046B0AF4A39CB'
        when json_extract(ActivityOperation.AppId, '$[1].application') = '308046B0AF4A39CB'
            then 'Firefox-308046B0AF4A39CB'
        when length (json_extract(ActivityOperation.AppId, '$[1].application')) > 17
            and length (json_extract(ActivityOperation.AppId, '$[1].application')) < 22
            then
            replace(replace(replace(replace(replace
            (json_extract(ActivityOperation.AppId, '$[0].application'),
            '{'||'6D809377-6AF0-444B-8957-A3773F02200E'||'}', '* ProgramFilesX64 * '),
            '{'||'7C5A40EF-A0FB-4BFC-874A-C0F2E0B9FA8E'||'}', '* ProgramFilesX32 * '),
            '{'||'1AC14E77-02E7-4E5D-B744-2EB1AE5198B7'||'}', '* System * ' ),
            '{'||'F38BF404-1D43-42F2-9305-67DE0B28FC23'||'}', '* Windows * '),
            '{'||'D65231B0-B2F1-4857-A4CE-A8E7C6EA7D27'||'}', '* SystemX86 * ')
            else    replace(replace(replace(replace(replace(json_extract(ActivityOperation.AppId,
            '$[1].application'),
            '{'||'6D809377-6AF0-444B-8957-A3773F02200E'||'}', '* ProgramFilesX64 * ' ),
            '{'||'7C5A40EF-A0FB-4BFC-874A-C0F2E0B9FA8E'||'}', '* ProgramFilesX32 * '),
            '{'||'1AC14E77-02E7-4E5D-B744-2EB1AE5198B7'||'}', '* System * ' ) ,
            '{'||'F38BF404-1D43-42F2-9305-67DE0B28FC23'||'}', '* Windows * '),
            '{'||'D65231B0-B2F1-4857-A4CE-A8E7C6EA7D27'||'}', '* SystemX86 * ')
    end as Application,
   json_extract(ActivityOperation.Payload, '$.displayText') as 'File/title opened',
   json_extract(ActivityOperation.Payload, '$.description') as 'Full Path /Url',
   json_extract(ActivityOperation.Payload, '$.activationUri') as 'AppUriHandler',
   json_extract(ActivityOperation.Payload, '$.shellContentDescription') as 'FileShellLink (json)',
    case json_extract(ActivityOperation.AppId, '$[0].platform')
        when 'afs_crossplatform' then 'Yes'
        when 'host' then
            (case json_extract(ActivityOperation.AppId, '$[1].platform')
            when 'afs_crossplatform' then 'Yes' else null end)
        else null
    end as 'Synced',
    case
        when json_extract(ActivityOperation.AppId, '$[0].platform') = 'afs_crossplatform'
            then json_extract(ActivityOperation.AppId, '$[1].platform')
        else json_extract(ActivityOperation.AppId, '$[0].platform')
    end as 'Platform_id',
   ActivityOperation.PackageIdHash as 'Hash',
   ActivityOperation.OperationType as 'Status',
    case
        when ActivityOperation.Id in
            (select Activity.Id from Activity where Activity.Id = ActivityOperation.Id)
            then 'Removed'
    end as 'WasRemoved',
```

```
    case
        when ActivityOperation.Id
            in(select Activity.Id from Activity where Activity.Id = ActivityOperation.Id)
            then null else 'In Queue'
    end as 'UploadQueue',
    case ActivityOperation.ActivityType
        when 5 then 'Open App/File/Page' when 6 then 'App In Use/Focus'
        else 'Unknown yet'
    end as 'Activity_type',
   ActivityOperation.PlatformDeviceId as 'Device ID',
   json_extract(ActivityOperation.OriginalPayload, '$.type') as 'Type',
   json_extract(ActivityOperation.OriginalPayload, '$.appDisplayName') as 'Original Displayed Name',
   json_extract(ActivityOperation.OriginalPayload, '$.displayText') as 'Original File/title opened',
   json_extract(ActivityOperation.OriginalPayload, '$.description') as 'Original Full Path /Url',
   json_extract(ActivityOperation.OriginalPayload, '$.activationUri') as 'Original AppUriHandler',
   time(json_extract(ActivityOperation.Payload, '$.activeDurationSeconds'),'unixepoch') as 'Active Duration',
   time(json_extract(ActivityOperation.OriginalPayload, '$.activeDurationSeconds'),'unixepoch') as 'Orig.Duration',
   case
        when cast((ActivityOperation.EndTime - ActivityOperation.StartTime) as integer) < 0 then '-'
        else time(cast((ActivityOperation.EndTime - ActivityOperation.StartTime) as integer),'unixepoch')
   end as 'Calculated Duration',
   datetime(ActivityOperation.StartTime, 'unixepoch', 'localtime') as 'StartTime',
   datetime(ActivityOperation.LastModifiedTime, 'unixepoch', 'localtime') as 'LastModified',
    case
        when ActivityOperation.OriginalLastModifiedOnClient > 0
            then datetime(ActivityOperation.OriginalLastModifiedOnClient, 'unixepoch', 'localtime')
            else '  -  '
    end as 'LastModifiedOnClient',
    case
        when ActivityOperation.EndTime > 0
            then datetime(ActivityOperation.EndTime, 'unixepoch', 'localtime')
            else null
    end as 'EndTime',
    case
        when ActivityOperation.CreatedInCloud > 0
            then datetime(ActivityOperation.CreatedInCloud, 'unixepoch', 'localtime')
            else null
    end as 'CreatedInCloud',
   json_extract(ActivityOperation.OriginalPayload, '$.userTimezone') as 'TZone',
   cast((ActivityOperation.ExpirationTime - ActivityOperation.LastModifiedTime)
    as integer) / '86400' as 'Expires In days',
   datetime(Activity_PackageId.ExpirationTime, 'unixepoch', 'localtime') as 'Expiration on PackageID',
   datetime(ActivityOperation.ExpirationTime, 'unixepoch', 'localtime') as 'Expiration',
     '{' || substr(hex(Activity_PackageId.ActivityId), 1, 8) || '-' ||
            substr(hex(Activity_PackageId.ActivityId), 9, 4) || '-' ||
            substr(hex(Activity_PackageId.ActivityId), 13, 4) || '-' ||
            substr(hex(Activity_PackageId.ActivityId), 17, 4) || '-' ||
   substr(hex(Activity_PackageId.ActivityId), 21, 12) || '}' as 'ID'

from Activity_PackageId
```

```
join ActivityOperation on Activity_PackageId.ActivityId = ActivityOperation.Id
where   Activity_PackageId.Platform = json_extract(ActivityOperation.AppId, '$[0].platform')
    and Activity_PackageId.ActivityId = ActivityOperation.Id

union  -- Join Activity & ActivityOperation Queries to get results from both Tables

select -- This the Activity Table Query
   Activity.ETag as 'Etag',
   json_extract(Activity.Payload, '$.appDisplayName') AS 'Program Name',
    case
        when json_extract(Activity.AppId, '$[0].application') = '308046B0AF4A39CB'
            then 'Firefox-308046B0AF4A39CB'
        when json_extract(Activity.AppId, '$[1].application') = '308046B0AF4A39CB'
            then 'Firefox-308046B0AF4A39CB'
        when length (json_extract(Activity.AppId, '$[0].application')) > 17 and
                length(json_extract(Activity.AppId, '$[0].application')) < 22
            then replace(replace(replace(replace(replace(json_extract(Activity.AppId, '$[1].application'),
            '{'||'6D809377-6AF0-444B-8957-A3773F02200E'||'}', '* ProgramFilesX64 * ' ),
            '{'||'7C5A40EF-A0FB-4BFC-874A-C0F2E0B9FA8E'||'}', '* ProgramFilesX32 * '),
            '{'||'1AC14E77-02E7-4E5D-B744-2EB1AE5198B7'||'}', '* System * ' ),
            '{'||'F38BF404-1D43-42F2-9305-67DE0B28FC23'||'}', '* Windows * '),
            '{'||'D65231B0-B2F1-4857-A4CE-A8E7C6EA7D27'||'}', '* SystemX86 * ')
       when json_extract(Activity.AppId, '$[0].application') = '308046B0AF4A39CB'
            then 'Firefox-308046B0AF4A39CB'
            else  replace(replace(replace(replace(replace
            (json_extract(Activity.AppId, '$[0].application'),
            '{'||'6D809377-6AF0-444B-8957-A3773F02200E'||'}', '* ProgramFilesX64 * '),
            '{'||'7C5A40EF-A0FB-4BFC-874A-C0F2E0B9FA8E'||'}', '* ProgramFilesX32 * '),
            '{'||'1AC14E77-02E7-4E5D-B744-2EB1AE5198B7'||'}', '* System * '),
            '{'||'F38BF404-1D43-42F2-9305-67DE0B28FC23'||'}', '* Windows * '),
            '{'||'D65231B0-B2F1-4857-A4CE-A8E7C6EA7D27'||'}', '* SystemX86 * ')
    end as 'Application',
   json_extract(Activity.Payload, '$.displayText') as 'File/title opened',
   json_extract(Activity.Payload, '$.description') as 'Full Path /Url',
   json_extract(Activity.Payload, '$.activationUri') as 'AppUriHandler',
   json_extract(Activity.Payload, '$.shellContentDescription') as 'FileShellLink (json)',
    case json_extract(Activity.AppId, '$[0].platform')
        when 'afs_crossplatform' then 'Yes'
        when 'host' then (case json_extract(Activity.AppId, '$[1].platform')
        when 'afs_crossplatform' then'Yes' else null end) else null
    end as 'Synced',
    case
        when json_extract(Activity.AppId, '$[0].platform') = 'afs_crossplatform'
        then json_extract(Activity.AppId, '$[1].platform')
        else json_extract(Activity.AppId, '$[0].platform')
    end as 'Platform_id',
   Activity.PackageIdHash as 'Hash',
   Activity.ActivityStatus as 'Status',
   null as 'WasRemoved',
   'No' as 'UploadQueue',
```

```
    case Activity.ActivityType
        when 5 then 'Open App/File/Page' when 6 then 'App In Use/Focus'
        else 'Unknown yet'
    end as 'Activity_type',
    Activity.PlatformDeviceId as 'Device ID',
    json_extract(Activity.OriginalPayload, '$.type') as 'Type',
    json_extract(Activity.OriginalPayload, '$.appDisplayName') as 'Original Program Name',
    json_extract(Activity.OriginalPayload, '$.displayText') as 'Original File/title opened',
    json_extract(Activity.OriginalPayload, '$.description') as 'Original Full Path /Url',
    json_extract(Activity.OriginalPayload, '$.activationUri') as 'Original AppUriHandler',
    time(json_extract(Activity.Payload, '$.activeDurationSeconds'),'unixepoch') as 'Active Duration',
    time(json_extract(Activity.OriginalPayload, '$.activeDurationSeconds'),'unixepoch' ) as 'Orig.Duration',
    case
        when cast ((Activity.EndTime - Activity.StartTime) as integer) < 0 then '-'
        else time(cast((Activity.EndTime - Activity.StartTime) as integer),'unixepoch')
    end as 'Calculated Duration',
    datetime(Activity.StartTime, 'unixepoch', 'localtime') as 'StartTime',
    datetime(Activity.LastModifiedTime, 'unixepoch', 'localtime') as 'LastModified',
    case
        when Activity.OriginalLastModifiedOnClient > 0
            THEN datetime(Activity.OriginalLastModifiedOnClient, 'unixepoch', 'localtime')
            ELSE '  -  '
    end as 'LastModifiedOnClient',
    case
        when Activity.EndTime > 0
            then datetime(Activity.EndTime, 'unixepoch', 'localtime')
        else "-"
    end as 'EndTime',
    case
        when Activity.CreatedInCloud > 0
            then datetime(Activity.CreatedInCloud, 'unixepoch', 'localtime')
            else "-"
    end as 'CreatedInCloud',
    json_extract(Activity.OriginalPayload, '$.userTimezone') as 'TZone',
    cast((Activity.ExpirationTime - Activity.LastModifiedTime) as integer) / '86400' as 'Expires In days',
    datetime(Activity_PackageId.ExpirationTime, 'unixepoch', 'localtime') as 'Expiration on PackageID',
    datetime(Activity.ExpirationTime, 'unixepoch', 'localtime') as 'Expiration',
          '{' || substr(hex(Activity_PackageId.ActivityId), 1, 8) || '-' ||
                 substr(hex(Activity_PackageId.ActivityId), 9, 4) || '-' ||
                 substr(hex(Activity_PackageId.ActivityId), 13, 4) || '-' ||
                 substr(hex(Activity_PackageId.ActivityId), 17, 4) || '-' ||
                 substr(hex(Activity_PackageId.ActivityId), 21, 12) || '}' as 'ID'
from Activity_PackageId
join Activity on Activity_PackageId.ActivityId = Activity.Id
where   Activity_PackageId.Platform = json_extract(Activity.AppId, '$[0].platform')
    and Activity_PackageId.ActivityId = Activity.Id

order by Etag desc;  -- Edit this line to change the sorting

-- EOF
```

The above query can also be found online at:
https://kacos2000.github.io/WindowsTimeline/

The above query pulls information from the 'Activity' , 'ActivityOperation' and 'Activity_PackageId' tables. The 'Activity_PackageId' table is more or less a 'cache' of the unique IDs of each application execution and their expiration time.

Some field explanations provided by Microsoft:

Name	Type	Description
status	EnumType	Set by the server. A status code used to identify valid objects. Values: active, updated, deleted, ignored.
userTimezone	String	Optional. The timezone in which the user's device used to generate the activity was located at activity creation time. Values supplied as Olson IDs in order to support cross-platform representation.
createdDateTime	DateTimeOffset	Set by the server. DateTime in UTC when the object was created on the server.
lastModifiedDateTime	DateTimeOffset	Set by the server. DateTime in UTC when the object was modified on the server.
id	String	Required. Client-set GUID for the **historyItem** object.
startedDateTime	DateTimeOffset	Required. UTC DateTime when the **historyItem** (activity session) was started. Required for timeline history.
lastActiveDateTime	DateTimeOffset	Optional. UTC DateTime when the **historyItem** (activity session) was last understood as active or finished - if null, **historyItem** status should be Ongoing.
expirationDateTime	DateTimeOffset	Optional. UTC DateTime when the **historyItem** will undergo hard-delete. Can be set by the client.
activeDurationSeconds	int	Optional. The duration of active user engagement. if not supplied, this is calculated from the **startedDateTime** and **lastActiveDateTime**.

The Activity' table's 'ID' (*Primary key*) field is a unique value:
[Activity]([Id] GUID PRIMARY KEY NOT NULL

It is the unique ID of each entry/user activity. The same ID can be seen more than 2 times in the Activity_PackageID table depending on the 'Platform' field which has the following possible values (*Note: most of project Rome is still undocumented, so not all entries can be defined*):

"Host" *(is listed as many times as the number of entries in the Activity table),*
"packageid",
"x_exe_path" for Programs called from a specific path (e.g. Standalone executables)

plus one or more of the following (platform-specific application IDs):

"windows_win32" for Installed Software (non UWP) Applications
"windows_universal' for Windows UWP Apps
"msa",
"ios",
"android", and
"web"

As seen by this MS example:

Example:

```
{"platform":"windows_universal", "application":"Microsoft.Contoso_8wekyb3d8bbwe"},
{"platform":"windows_win32", "application":"DefaultBrowser_NOPUBLISHERID!Microsoft.Contoso.Default"},
{"platform":"android","application":"com.example.myapp"},
{"platform":"ios", "application":"com.example.myapp"},
{"platform":"web", "application":"https://contoso.com"},
{"platform":"web", "application":"https://chat.contoso.com"},
{"platform":"msa", "application":"00000000603E0BF"},
{"platform":"msa", "application":"48932b46-98b1-4020-9be4-cc7a65643c9e"},
]
```

If the Sync activity is enabled (user logs in with a MS Account) then another entry may also be seen: "afs_crossplatform" which shows a cloud sync enabled activity, and is an additional entry to the same ones seen when using a local account.

When using a MS account:

5251	���% c�...	afs_crossplatform	6082723978844556501	1529741627
5252	���% c�...	windows_win32	{7c5a40ef-a0fb-4bfc-874a-c0f2e0b9fa8e}\foobar2000\f...	1529741627
5253	���% c�...	packageid	{7c5a40ef-a0fb-4bfc-874a-c0f2e0b9fa8e}\foobar2000\f...	1529741627
5254	���% c�...	host		1529741627

When using a Local account there is NO 'afs_crossplatform' entry:

79	e% ❒ @❒I❒...	windows_win32	microsoft.window...	1529742843
80	e% ❒ @❒I❒...	packageid	microsoft.window...	1529742843
81	e% ❒ @❒I❒...	host		1529742843

As can also be seen below (*these are for a single ID of a Microsoft Word entry*):

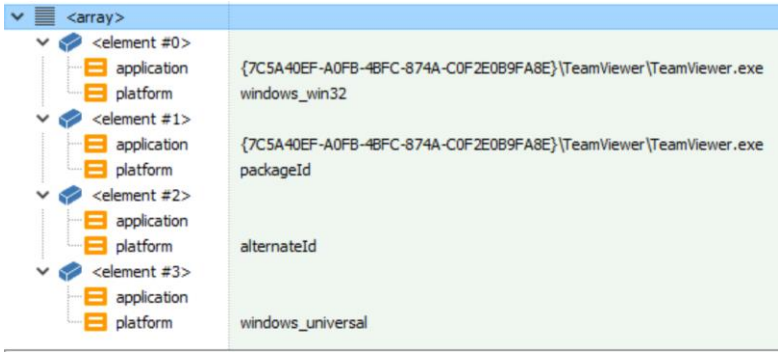
ActivityId	Platform	PackageName	ExpirationTime
CA06838559F87AE076659BF18C593CF9	host	word.activity.windows.com	2018-06-29 16:39:49
CA06838559F87AE076659BF18C593CF9	afs_crossplatform	16883319693718197224	2018-06-29 16:39:49
CA06838559F87AE076659BF18C593CF9	windows_universal	microsoft.office.word_8wekyb3d8bbwe	2018-06-29 16:39:49
CA06838559F87AE076659BF18C593CF9	windows_universal	microsoft.office.desktop_8wekyb3d8bbwe	2018-06-29 16:39:49
CA06838559F87AE076659BF18C593CF9	android	com.microsoft.office.word	2018-06-29 16:39:49
CA06838559F87AE076659BF18C593CF9	ios	com.microsoft.office.word	2018-06-29 16:39:49
CA06838559F87AE076659BF18C593CF9	windows_win32	microsoft.office.winword.exe.15	2018-06-29 16:39:49
CA06838559F87AE076659BF18C593CF9	windows_win32	{7c5a40ef-a0fb-4bfc-874a-c0f2e0b9fa8e}\microsoft office\office15\winword.exe	2018-06-29 16:39:49
CA06838559F87AE076659BF18C593CF9	windows_win32	{6d809377-6af0-444b-8957-a3773f02200e}\microsoft office\office15\winword.exe	2018-06-29 16:39:49
CA06838559F87AE076659BF18C593CF9	windows_win32	{7c5a40ef-a0fb-4bfc-874a-c0f2e0b9fa8e}\microsoft office\office14\winword.exe	2018-06-29 16:39:49
CA06838559F87AE076659BF18C593CF9	windows_win32	{6d809377-6af0-444b-8957-a3773f02200e}\microsoft office\office14\winword.exe	2018-06-29 16:39:49
CA06838559F87AE076659BF18C593CF9	msa	000000000003edf6	2018-06-29 16:39:49

The above can be seen by using this query:

```
Select
hex(Activity_PackageId.ActivityId) as 'ActivityId',
       Activity_PackageId.Platform ,
       Activity_PackageId.PackageName ,
       datetime (Activity_PackageId.ExpirationTime , 'unixepoch') as 'ExpirationTime'
FROM Activity_PackageId
join Activity on Activity_PackageId.ActivityId = Activity.Id
order by Activity.ActivityId
```

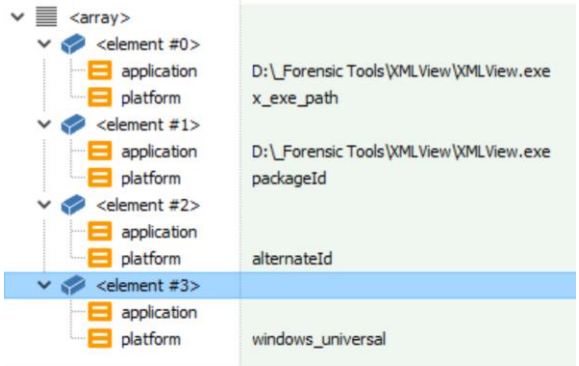
The 'Activity_PackageId' table seemingly has more information than the 'Activity' table. That is not exactly true though, as the 'Activity' table's 'AppID' field holds this information and a lot more. The 'Activity' and 'ActivityOperation' table 'AppID' fields are in fact Binary Large Objects (BLOB) in 'json array' format e.g.:

'AppID' entry for an Installed Application in Windows:

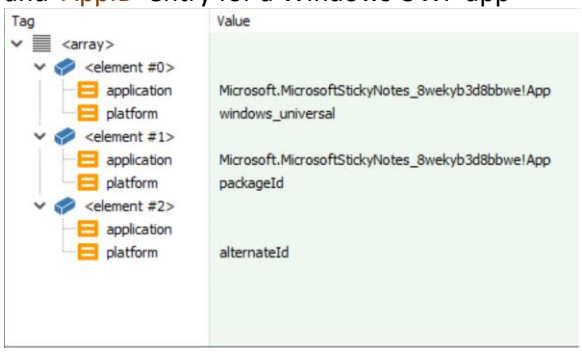


Note: *Notice that the path shown above is in the form of a GUID. It is a KNOWNFOLDERID used by windows for standard known folders like 'program files'. In the json file above, this is the GUID for ProgramFilesX86 (C:\Program Files (x86)\) folder. There are various lists of CLSIDs in the internet like https://msdn.microsoft.com/en-us/library/windows/desktop/dd378457(v=vs.85).aspx*

'AppID' entry for a standalone executable:



and 'AppID' entry for a Windows UWP app



Since this is a json array, we can do a query to see what values it holds in the 'application' and 'platform' fields. The `$[x].` depicts the number of the element in the array, and 'platform' or 'application' the respective field name:

```
-- List all entries in the AppID json array

select
hex(Activity.ID) as 'ID',
json_extract(Activity.AppId, '$[0].platform')  as 'p0' ,
json_extract(Activity.AppId, '$[0].application') as '0' ,
json_extract(Activity.AppId, '$[1].platform')  as 'p1' ,
json_extract(Activity.AppId, '$[1].application') as '1' ,
json_extract(Activity.AppId, '$[2].platform')  as 'p2' ,
json_extract(Activity.AppId, '$[2].application') as '2'  ,
json_extract(Activity.AppId, '$[3].platform')  as 'p3' ,
json_extract(Activity.AppId, '$[3].application') as '3' ,
json_extract(Activity.AppId, '$[4].platform')  as 'p4' ,
json_extract(Activity.AppId, '$[4].application') as '4',
json_extract(Activity.AppId, '$[5].platform')  as 'p5' ,
json_extract(Activity.AppId, '$[5].application') as '5' ,
json_extract(Activity.AppId, '$[6].platform')  as 'p6' ,
json_extract(Activity.AppId, '$[6].application') as '6' ,
json_extract(Activity.AppId, '$[7].platform')  as 'p7' ,
json_extract(Activity.AppId, '$[7].application') as '7' ,
json_extract(Activity.AppId, '$[8].platform')  as 'p8' ,
json_extract(Activity.AppId, '$[8].application') as '8'

from Activity
order by ID asc
```

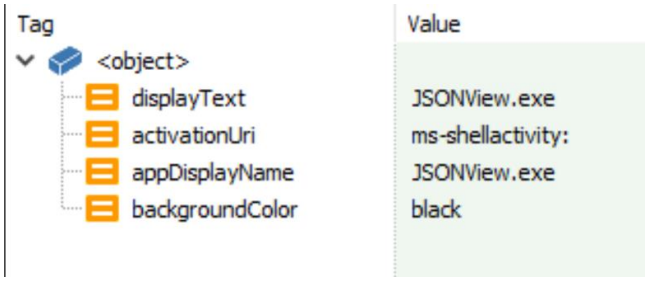
	A0	P0	A1	P1	A2	P2	A3	P3
19	{7C5A40EF-A0FB-4BFC-874A-C0F2E0B9FA8E}\Acronis\TrueImageHome\TrueImage.exe	x_exe_path	{7C5A40EF-A0FB-4BFC-874A-C0F2...	packageId		alternateId		windows_universal
20	Microsoft.Office.OUTLOOK.EXE.15	windows_win32	Microsoft.Office.OUTLOOK.EXE.15	packageId		alternateId		windows_universal
21	{6D809377-6AF0-444B-8957-A3773F02200E}\VOW Software\plist Editor Pro\plistEditor.exe	windows_win32	{6D809377-6AF0-444B-8957-A377...	packageId		alternateId		windows_universal
22	{7C5A40EF-A0FB-4BFC-874A-C0F2E0B9FA8E}\Acronis\TrueImageHome\TrueImage.exe	x_exe_path	{7C5A40EF-A0FB-4BFC-874A-C0F2...	packageId		alternateId		windows_universal
23	{6D809377-6AF0-444B-8957-A3773F02200E}\SQLite Expert\Professional 5\SQLiteExpertP...	windows_win32	{6D809377-6AF0-444B-8957-A377...	packageId		alternateId		windows_universal
24	Microsoft.Office.OUTLOOK.EXE.15	windows_win32	Microsoft.Office.OUTLOOK.EXE.15	packageId		alternateId		windows_universal

Or

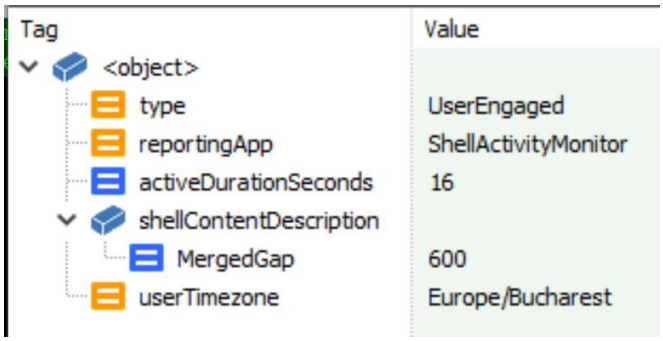
	2	p3	3	p4	4	p5	5	p6	6	p7	7	p8	8
1	osoftEdge_8wekyb3d8bbwe!MicrosoftEdge	msa	f44b1140-bc5e-48c6-8dc0-5cf5a53c0e34	msa	000000004C1BC462	android	com.microsoft.emmx.daily	android	com.microsoft.emmx.selfhost	android	com.microsoft.emmx	ios	com.microsoft.emmx.daily-df
2	osoftEdge_8wekyb3d8bbwe!MicrosoftEdge	msa	f44b1140-bc5e-48c6-8dc0-5cf5a53c0e34	msa	000000004C1BC462	android	com.microsoft.emmx.daily	android	com.microsoft.emmx.selfhost	android	com.microsoft.emmx	ios	com.microsoft.emmx.daily-df
3	osoftEdge_8wekyb3d8bbwe!MicrosoftEdge	msa	f44b1140-bc5e-48c6-8dc0-5cf5a53c0e34	msa	000000004C1BC462	android	com.microsoft.emmx.daily	android	com.microsoft.emmx.selfhost	android	com.microsoft.emmx	ios	com.microsoft.emmx.daily-df
4	osoftEdge_8wekyb3d8bbwe!MicrosoftEdge	msa	f44b1140-bc5e-48c6-8dc0-5cf5a53c0e34	msa	000000004C1BC462	android	com.microsoft.emmx.daily	android	com.microsoft.emmx.selfhost	android	com.microsoft.emmx	ios	com.microsoft.emmx.daily-df
5	osoftEdge_8wekyb3d8bbwe!MicrosoftEdge	msa	f44b1140-bc5e-48c6-8dc0-5cf5a53c0e34	msa	000000004C1BC462	android	com.microsoft.emmx.daily	android	com.microsoft.emmx.selfhost	android	com.microsoft.emmx	ios	com.microsoft.emmx.daily-df

Similarly, the ‘Activity’ and the ‘ActivityOperation’ table 'payload' fields (blob) are also in json format with different entries related to the different platform type as follows:

For "x_exe_path" , "windows_win32" and “windows_universal” it doesn't include duration when an app is first executed:



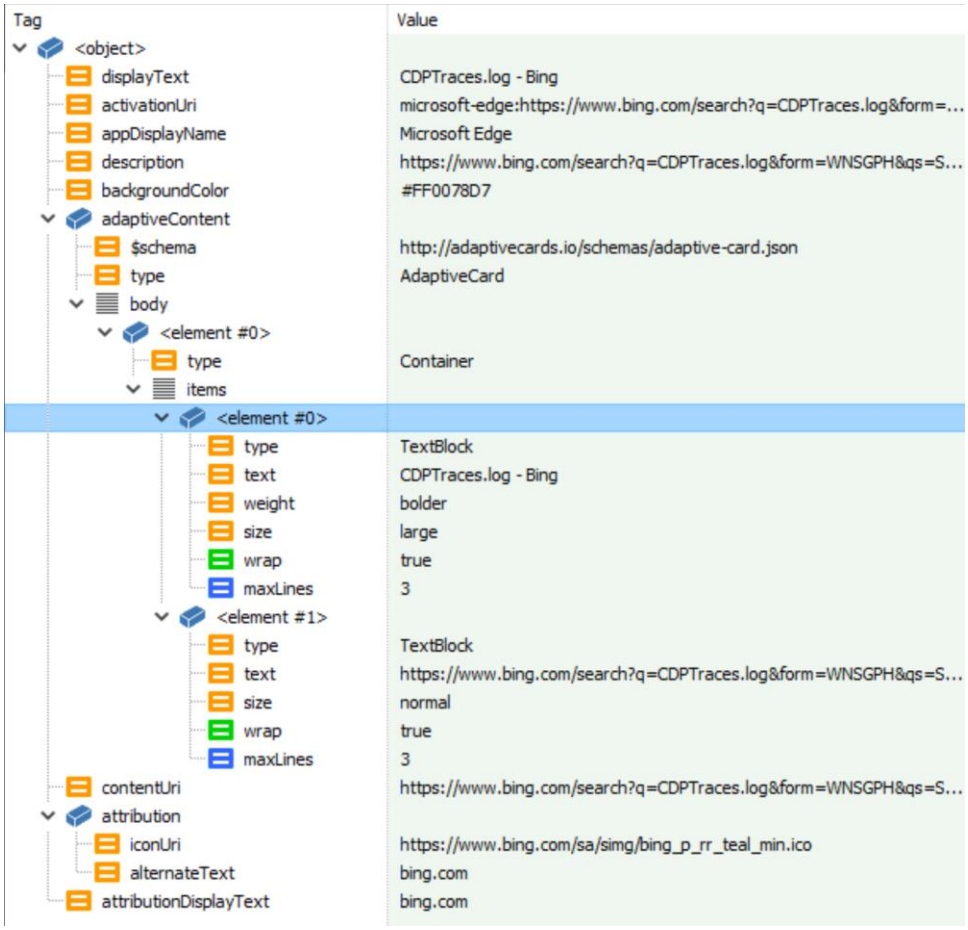
but includes it in subsequent entries:



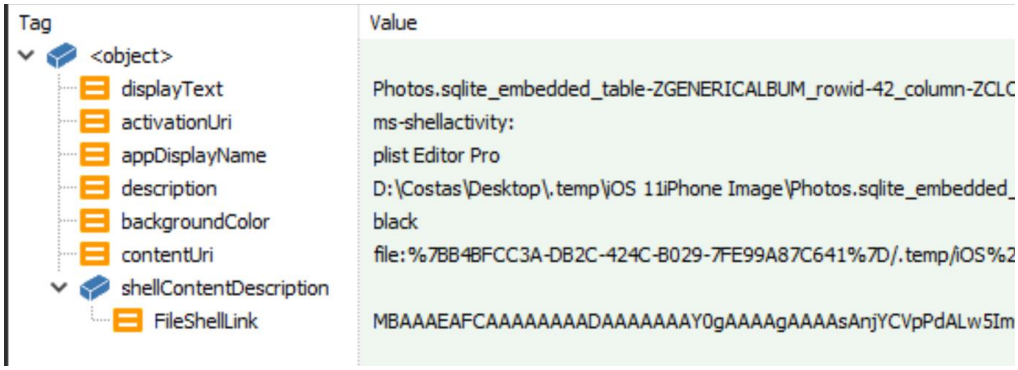
Having included my own duration calculation (calculated by subtracting the ‘StartTime’ from the ‘EndTime’), the duration is sometimes different than the one included in the json blob. A longer calculated duration could indicate that the app is in focus but idle i.e. the user does not interact with it.

When "windows_universal" or "windows_win32" applications open a file, the 'payload' field includes quite a bit of information.

"windows_universal" (In this case MS Edge - which uses the Adaptive Card framework - bing search for CDPTraces.log) can be seen below:



And 'windows_win32' here:



The 'ExpirationTime' fields in both Activity and Activity_PackageId tables are the same for each 'ID' and 'ActivityId' respectively (Unique field entry GUID).

All Dates/Times are stored by the database in UTC. To get the local times, we can adjust the query by using, 'localtime' e.g. :

```
datetime(Activity.StartTime , 'unixepoch', 'localtime'),
```

The ‘ExpirationTime’ is exactly 2592000 seconds or 30 Days from the relevant entry’s ‘LastModified’ time, and is easily seen by using the following:

```
Select
(Activity.ExpirationTime - Activity.LastModifiedTime) as 'Expires in'
From Activity
```

Sorting the query by the 'ETAG' field gives a historical list (sort of a timeline) according to the Events recorded in the db, but is not the same as if sorted by Start, End or Expiry datetimes. The ETAG can change for a particular entry’s GUID depending on the current action the database has recorded for that ID (e.g. user removed Tile from timeline).

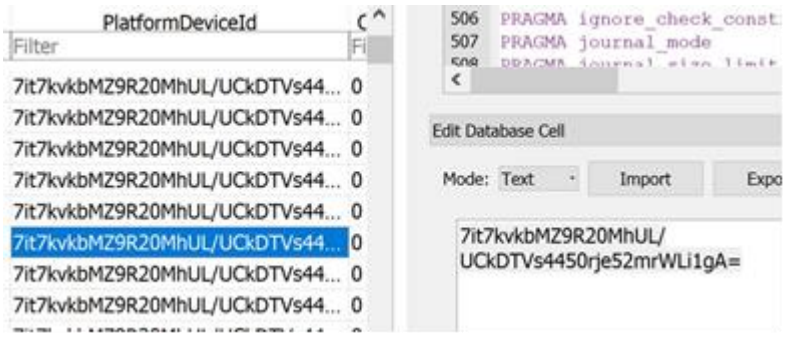
An example that an ID can have multiple ETAGs associated with it, is shown below:

	ID	Program Name	Nr_of_ETAGs_per_ID
1	14A18F08264CCBEEE1F87294159CC174	Microsoft.MicrosoftEdge_8wekyb3d8bbwe!MicrosoftEdge	12
2	E410C45C9351D584F5FF8771FB79EBE0	Microsoft.MicrosoftEdge_8wekyb3d8bbwe!MicrosoftEdge	12
3	3850384E31582BB4D7444A87238EBE71	Microsoft.MicrosoftEdge_8wekyb3d8bbwe!MicrosoftEdge	11
4	7FBCA86B1284531AB13714378DE462BF	Microsoft.MicrosoftEdge_8wekyb3d8bbwe!MicrosoftEdge	11
5	AF83A1D79E8018D88E34C6DA0367D8D2	Microsoft.MicrosoftEdge_8wekyb3d8bbwe!MicrosoftEdge	11
6	005A5B3C02AF18D3D743923C12A6A1A4	{1AC14E77-02E7-4E5D-B744-2EB1AE5198B7}\SnippingTool.exe	4
7	00AEEED61A3676A61C4683F4ADF956DC	{6D809377-6AF0-444B-8957-A3773F02200E}\Windows NT\Accessories\wordpad...	4
8	0197B6BDC361072CAE085484AF9E9D13	{6D809377-6AF0-444B-8957-A3773F02200E}\Windows NT\Accessories\wordpad...	4
9	019F2517AFD0B40206B668B4B979C5F3	Chrome	4
10	026D393D2A3C851046DD8296FFFFA0C1	Chrome	4

by running this query against a MS account’s ‘ActivitiesCache.db’:

```
select
hex(Activity.id) as 'ID',
json_extract(Activity.AppId, '$[2].application') as 'Program Name',
count(activity.etag) as 'Nr_of_ETAGs_per_ID'
from Activity
join Activity_PackageId on activity.id = Activity_PackageId.ActivityId
group by ID
order by Nr_of_ETAGs_per_ID desc
```

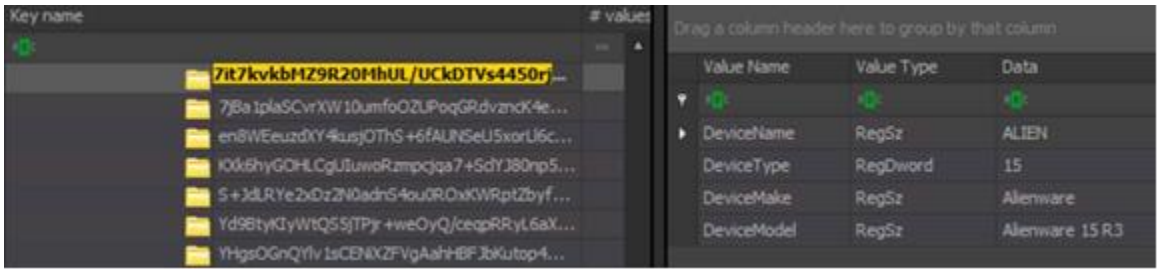
The ‘PlatformDeviceID’ seen in the ‘Activity’ & ‘ActivityOperations’ tables:



can be found in the user's NTUSER.dat at

"Software\Microsoft\Windows\CurrentVersion\TaskFlow\DeviceCache\".

As seen below:

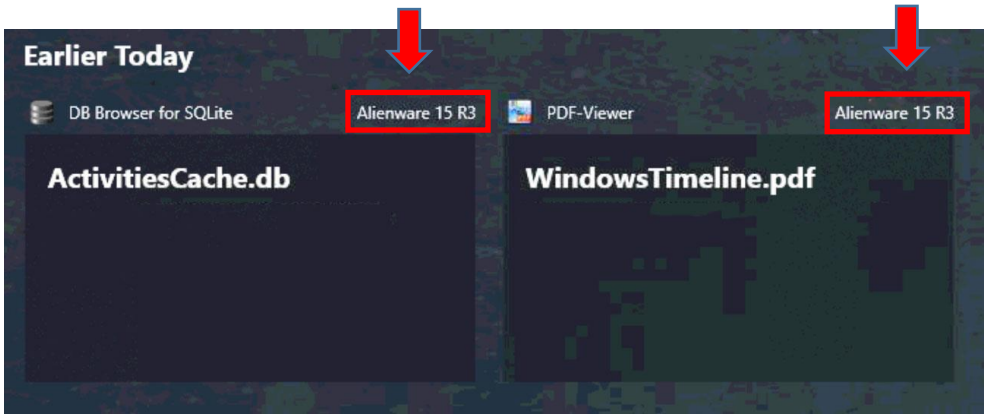


But it is not fixed - it looks encoded with the QuickXorHash Algorithm, similar to the 'FileShellLink' in the 'Package' blob - and it changes occasionally,

	Device ID
190	TDdtGmqdscQk7BxppyGhb2yzThGYW8Ldm8tsq+GJ9n4=
191	TDdtGmqdscQk7BxppyGhb2yzThGYW8Ldm8tsq+GJ9n4=
192	TDdtGmqdscQk7BxppyGhb2yzThGYW8Ldm8tsq+GJ9n4=
193	TDdtGmqdscQk7BxppyGhb2yzThGYW8Ldm8tsq+GJ9n4=
194	Yd9BtyKIyWtQS5jTPjr+weOyQ/ceqpRRyL6aX0XAiro=
195	Yd9BtyKIyWtQS5jTPjr+weOyQ/ceqpRRyL6aX0XAiro=
196	Yd9BtyKIyWtQS5jTPjr+weOyQ/ceqpRRyL6aX0XAiro=

still depicting the same Machine (*as seen at the relevant registry entries*). I have not yet found the pattern as to why or when it changes when a user logs in with a local account.

It is used to show the Device origin of a Tile in the Timeline view, such as:



The device name seen above is derived from the "DeviceModel" entry in the registry for the specific DeviceID. That information originates from the OEM information value in 'SystemProductName' at "HKLM\SYSTEM\ControlSet001\Control\SystemInformation" which is usually filled by the device manufacturer, or is blank when the computer is user assembled.

The ‘AppActivityId’ field has 3 possible observed values:

- ECB32AF3-1440-4086-94E3-5311F97F89C4 which looks like a GUID but having checked 4 different computers of different users except my own ones, this GUID remains the same in all of them .
- An alphanumeric identifier (Hash?) following the above GUID, which is specific to a file opened, or
- A URL opened by MS Edge (on MS Account related ‘ActivitiesCache.db’ databases).

This can be seen by running the following:

```
select
case when Activity.AppActivityId not like '%-%-%-%-%\%' then Activity.AppActivityId
when Activity.AppActivityId not like '%-%-%-%-%' then Activity.AppActivityId
else substr(Activity.AppActivityId , 38)
end as 'DeviceID',
datetime(Activity.StartTime, 'unixepoch', 'localtime') AS 'StartTime',
json_extract(Activity.Payload, '$.displayText') as 'File_Name'
from Activity
join Activity_PackageId on activity.id = Activity_PackageId.ActivityId
group by File_Name
order by StartTime asc
```

	DeviceID	StartTime	File_Name
13	ECB32AF3-1440-4086-94E3-5311F97F89C4	2018-05-01 07:20:10	7-Zip File Manager
14	ECB32AF3-1440-4086-94E3-5311F97F89C4	2018-05-01 07:24:15	SystemPropertiesAdvanced.exe
15	ECB32AF3-1440-4086-94E3-5311F97F89C4	2018-05-01 07:28:48	Opera Browser
16	ECB32AF3-1440-4086-94E3-5311F97F89C4	2018-05-01 07:35:45	ESEDatabaseView.exe
17	ECB32AF3-1440-4086-94E3-5311F97F89C4	2018-05-01 07:45:59	jre-8u171-windows-x64.exe
18	4cc59f0fe7ab8e7280b2c4228bcd76193b3e5f3f	2018-05-01 07:58:11	backup.tar
19	fc1379ad2d57897a2013f92a6e5d53e0dbd1e0f8	2018-05-01 08:01:38	tealium.db
20	b6e7e75f16f5d0fddf4260be2a2e1058bcfb619a	2018-05-01 08:05:13	flixster_dc2.db

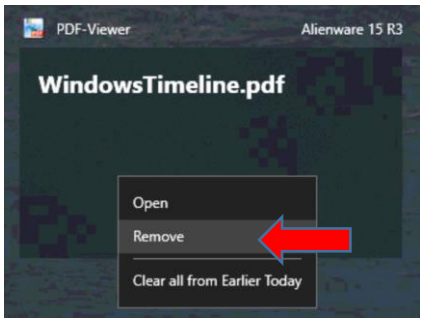
The ‘PackageIDHash’ field is a unique value related to each application. It appears to be the QuickXorHash of the executable – with this query:

```
select
        Activity.PackageIdHash as 'Hash',
        Activity_PackageId.PackageName as 'Name',
        datetime(Activity.StartTime, 'unixepoch', 'localtime') AS 'StartTime'
        from Activity
        join Activity_PackageId on activity.id = Activity_PackageId.ActivityId
group by Hash
order by StartTime desc
```

we can see that different versions of the same application have a different Hash value:

	Hash	Name	StartTime
1	LpMbiNsaIvhwz9RKScqy+21v9qKArm8vsmGRxfqVh8k=	d:_forensic tools\dmde-3.2.0.692-win32-gui\dmde.exe	2018-05-28 19:44:18
2	83TnlWHEKIgYDl4QLfHkrWYjNrlRwf7OP4R6Djg6TXI=	d:_forensic tools\dmde-3.0.6.648-win32-gui\dmde.exe	2018-05-28 19:44:00

The ‘Activity’ table’s ‘ActivityStatus’ and ‘ActivityOperation’ tables’s ‘OperationType’ have 3 observed values ranging from 1 to 3 and depict the related Tile Status as:
1 -> Active (means it is still an open application)
2-> Updated (means the entry is an update for a previously Active entry)
3-> Deleted (means the Tile is removed from the Timeline)
And there could be a 4th entry with a value of **4**, meaning that the entry is ‘Ignored’.
When a Tile is removed from the Timeline by the user,



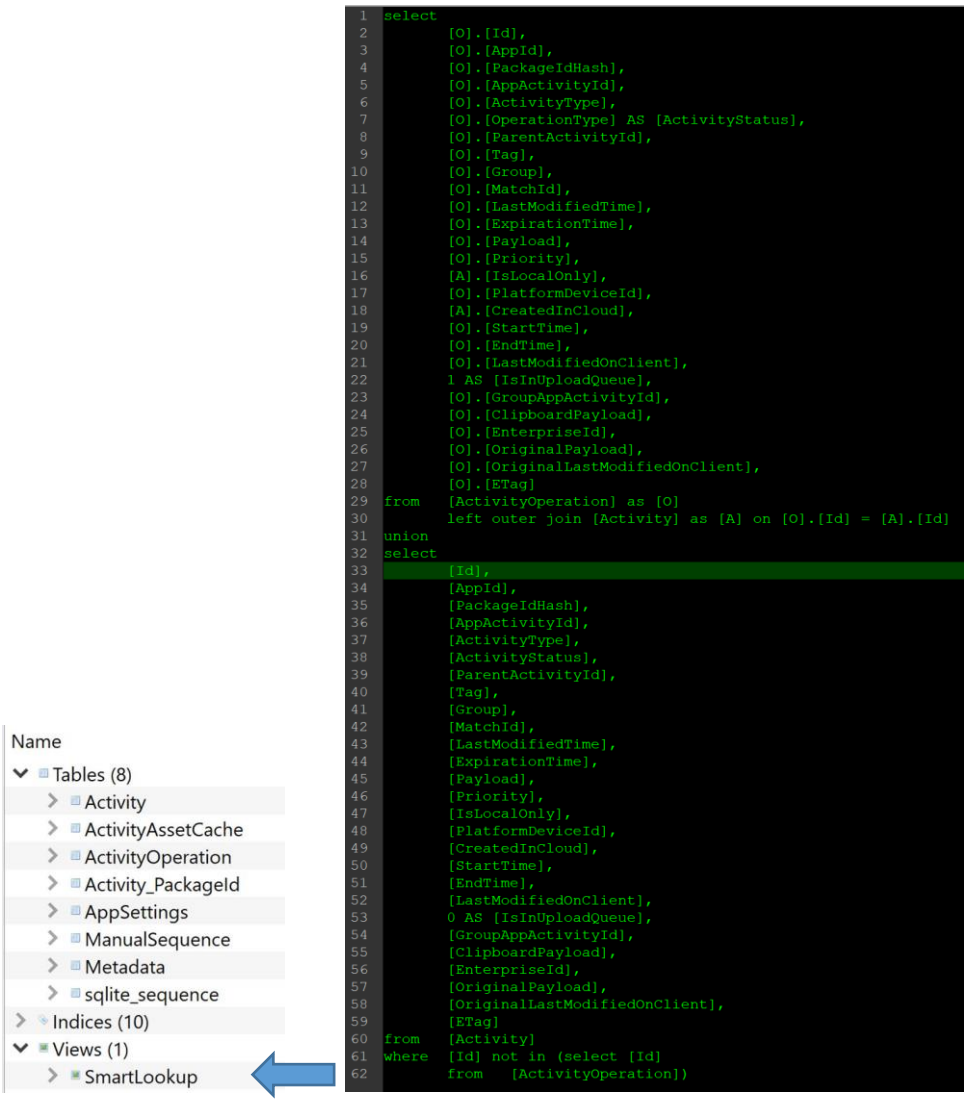
the associated ID(s) to this Tile remain the same, but the entries are copied from the ‘Activity’ to the ‘ActivityOperation’ table with a NEW ‘Etag’ and new type/status value:

```
1 select
2 activity.etag as 'Activity Etag',
3 hex(activity.id) as 'Activity ID',
4 Activity.ActivityStatus as 'Type',
5 ActivityOperation.ETag as 'ActivityOperation Etag',
6 hex(ActivityOperation.Id) as 'ActivityOperation ID',
7 ActivityOperation.OperationType as 'Type'
8 from Activity
9 join ActivityOperation on Activity.Id = ActivityOperation.Id
```

	Activity Etag	Activity ID	Type	ActivityOperation Etag	ActivityOperation ID	Type
1	19255	942964941C4A1FC75A837A1CFC8CFF92	1	20098	942964941C4A1FC75A837A1CFC8CFF92	3
2	19264	0DE4C1CFA7AD59E7B9F9A3A0509D21CA	1	20099	0DE4C1CFA7AD59E7B9F9A3A0509D21CA	3
3	19267	4B8DDEC337073252145BE8B622D2A619	1	20100	4B8DDEC337073252145BE8B622D2A619	3
4	19270	3704910DA81C809DB04126F03E300092	1	20101	3704910DA81C809DB04126F03E300092	3
5	19410	289D11B903A8BC64FE6305B96BC46750	1	20102	289D11B903A8BC64FE6305B96BC46750	3

When the user logs in with a local account, and a tile is removed from the Timeline, the associated entries for that are copied from the ‘Activity’ to the ‘ActivityOperation’ table with a different ‘ETAG’ value, and stay there until expiration time.

An easy way to figure out which entries are User Removed can be derived from viewing the ‘SmartLookup’ view (*automatically created query included in the database*) which, in essence, is the source of the actual timeline a user observes:



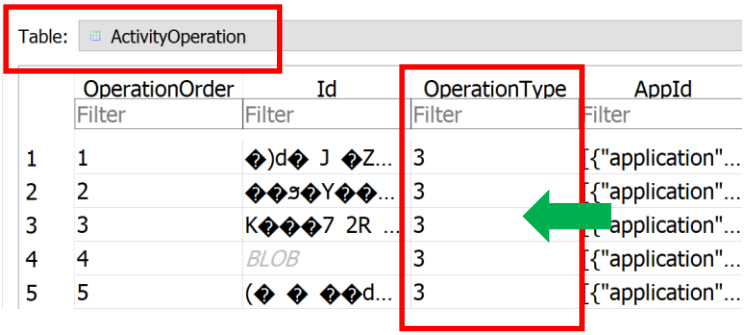
The key here is the line: FROM [Activity] WHERE [Id] NOT IN (SELECT [Id] FROM [ActivityOperation])

If we check the 'SmartLookup' query data view, it lists all 'Activity' table entries plus all 'ActivityOperation' table entries minus the 'Activity' table entries that appear in both tables with the same ID. Why? Because when the ActivityOperation.Id is also found in the Activity.Id then the entry has been (user) removed from the timeline.

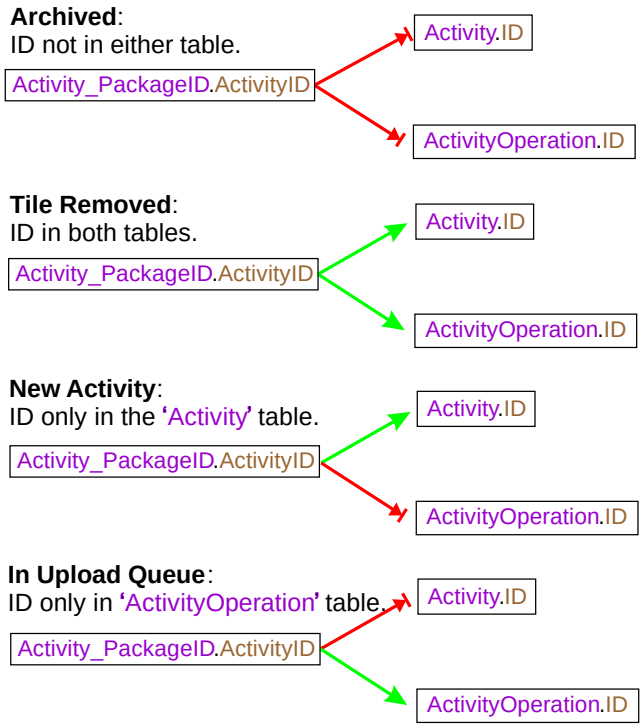
What this query also does, is to mark (and list) all entries in the 'Activity' table with a value of 0 in the field [IsInUploadQueue], and all entries in the 'ActivityOperations with a value of 1 in the [IsInUploadQueue]. Which this means is that the entries in the 'ActivityOperations' table are in the Upload to Cloud Queue.

However, when a user is logged in with a MS Account, and the user removes a tile from the Timeline, the respective entries are copied to the 'ActivityOperation' table with a new 'Etag' value and a new type/status of 3 (*Deleted*) momentarily until they are synced, and then they are

moved to the 'Activity' table until they expire. The type/status value 3 tells the other synchronized devices not to display this tile in their Timeline. This may be confusing when examining an offline system:



The following query checks every ID on 'Activity_PackageID' against both 'Activity' and 'ActivityOperation' tables and provides the relevant results:




```
select
case
        when
                Activity_PackageId.ActivityId in (select Activity.Id from activity) and
                Activity_PackageId.ActivityId in (select ActivityOperation.Id
        from ActivityOperation)
        then 'Tile Removed - '||hex(Activity_PackageId.ActivityId)
        else
                case
                when     Activity_PackageId.ActivityId  not in
                                (select Activity.Id from activity) and
                                Activity_PackageId.ActivityId  not in
                                (select ActivityOperation.Id from ActivityOperation)
                then 'Is Archived - '||hex(Activity_PackageId.ActivityId)
                else
```

```
                        case
                        when Activity_PackageId.ActivityId in
                        (select Activity.Id from activity) and
                        Activity_PackageId.ActivityId not in
                        (select ActivityOperation.Id from ActivityOperation)
                                then 'New Activity - '||hex(Activity_PackageId.ActivityId)
                                else
                                case
                                        when Activity_PackageId.ActivityId
                                        not in (select Activity.Id from Activity)
                                        and  Activity_PackageId.ActivityId
                                        in (select ActivityOperation.Id from ActivityOperation)
                                        then '
                                        In Upload Queue - '||hex(Activity_PackageId.ActivityId)
                end
                end
        end
end as 'Status_ID', -- This field includes both the Status and the unique ID of the associated activity
Activity_PackageId.PackageName as 'PackageName', -- The program/ associated with the above ID
datetime(Activity_PackageId.ExpirationTime, 'unixepoch', 'localtime') as 'ExpirationTime',
Activity_PackageId.Platform

from Activity_PackageId
where Activity_PackageId.Platform in ('windows_win32', 'windows_universal', 'x_exe_path')
group by Status_ID
order by ExpirationTime asc
```

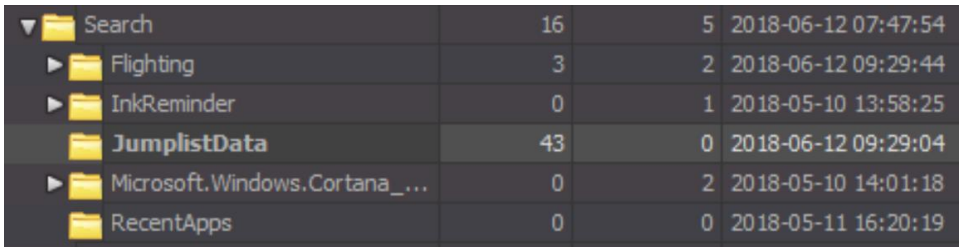
We can see which ID's entry is a:

- 'New activity'
- 'Had the associated timeline tile removed'
- 'In the Upload Queue'
- Archived' *(until expiration time)*

	Status_ID	PackageName	ExpirationTime	Platform
2464	New Activity - FAC3BF8F86CC8D737E2912333C1B480B	microsoft.office.outlook.exe.15	2018-08-21 18:29:32	windows_win32
2465	New Activity - A082487B90E1920C1E794CCE485D46D6	microsoft.office.outlook.exe.15	2018-08-21 18:29:54	windows_win32
2466	Is Archived - AA02B8BF0A259DEAE764099B3E5BCB8B	{6d809377-6af0-444b-8957-a3773f02200e}\db browser for sqlite\db browser for sqlite.exe	2018-08-24 10:30:25	windows_win32
2467	Is Archived - 5D82FFF1A179FFEAC9A59CE52FC48ACA	windows.immersivecontrolpanel_cw5n1h2txyewy	2018-08-24 10:30:44	windows_universal
2468	New Activity - E4D4D3EE8FF161D4D330C2C5FAAEECF2	{6d809377-6af0-444b-8957-a3773f02200e}\db browser for sqlite\db browser for sqlite.exe	2018-08-24 10:30:44	windows_win32
2469	Is Archived - F48D10A0C52199D9D35BDB1EE68CBC4E	windows.immersivecontrolpanel_cw5n1h2txyewy	2018-09-25 23:06:50	windows_universal
2470	Is Archived - 65D2974DC6A859499B8B20FA926146BE	microsoft.office.outlook.exe.15	2018-11-21 17:29:58	windows_win32
2471	Is Archived - 732C5C66B18DBB39B19AAE2D48AA8C65	windows.immersivecontrolpanel_cw5n1h2txyewy	2018-11-21 17:29:59	windows_universal

---

As a final note, and not yet sure if this is related, there appears to be a new Jumplist in windows 1803, located at NTUSER.dat "*Software\Microsoft\Windows\CurrentVersion\Search\JumplistData*":



Which in the value field has a list of applications each

Value Name	Value Type	Data
E7CF176E110C211B	RegQword	131732693443306503
{6D809377-6AF0-444B-8957-A3773F02200E}\Emsisoft Anti-Malware\a2start.exe	RegQword	131732664857040045
windows.immersivecontrolpanel_cw5n1h2txyewy!microsoft.windows.immersivecontrolpanel	RegQword	131732661999990249
OperaSoftware.OperaWebBrowser.1482616678	RegQword	131732632253640937
{6D809377-6AF0-444B-8957-A3773F02200E}\DB Browser for SQLite\DB Browser for SQLite.exe	RegQword	131732229857044091
{6D809377-6AF0-444B-8957-A3773F02200E}\Windows NT\Accessories\wordpad.exe	RegQword	131732218473490864
D:_Forensic Tools\RegistryExplorer_RECmd\RegistryExplorer.exe	RegQword	131732192428203938
Microsoft.Office.EXCEL.EXE.15	RegQword	131732189855753289
{1AC14E77-02E7-4E5D-B744-2EB1AE5198B7}\WindowsPowerShell\v1.0\powershell.exe	RegQword	131732037370491875
Microsoft.AutoGenerated.{4386C36E-24C8-1079-C3DA-17FF7E718F1E}	RegQword	131731995196161952
{6D809377-6AF0-444B-8957-A3773F02200E}\Tracker Software\PDF Viewer\PDFXCview.exe	RegQword	131731867951745237
Corel.DrawHome.20	RegQword	131731845581242218
Microsoft.Office.WINWORD.EXE.15	RegQword	131731121065717654
D:\Costas\Downloads\XWFIM\XWFIM.exe	RegQword	131731056023994705
{1AC14E77-02E7-4E5D-B744-2EB1AE5198B7}\notepad.exe	RegQword	131731055155095654
{6D809377-6AF0-444B-8957-A3773F02200E}\WinRAR\WinRAR.exe	RegQword	131731054662301052
D:\Costas\Dropbox\IACIS_QA\JSONView\JSONView.exe	RegQword	131731044186574237
Microsoft.Windows.RemoteDesktop	RegQword	131730894892470341

With a Data value (in Windows FileTime) which appears to be the date/time an application was launched through the search (Cortana box), or the taskbar, but certain apps only appear if you open a file (e.g. MS Word opens a .doc file for editing or viewing).

Source date/times

131732693443306503

Original	date	format
131732693443306503	12 Jun 18 9:29:04 am	FileTime

What's interesting with the JumpList is that while these entries do not appear (in my testing) in the normal Custom or Automatic jumplists, but they appear with the same dates in ActivitiesCache.db. Further testing of this is needed though.