Jabra Integration Service

A deployment guide to the Jabra Integration Service solution

Deployment Guide

Revision history

Rev.	Date	Description
1.0	2018-12-18	The initial version of this document.
1.1	2021-02-24	Document migrated to a new template, minor corrections to wording.
1.2	2021-03-01	Updates for release 1.9. Product list updated.
1.3	2021-04-22	Updated event documentation. Removed Lync/Skype for Business support
1.4	2021-08-31	Added support for Basic Authentication
1.5	2021-10-05	Moved list of supported devices to separate document. Added support for devices from other vendors

Table of Contents

Figures, Tables, and Examples

1 Introduction

Jabra Integration Service provides high-level functionality for:

- Event logging of Jabra device and system information
- Firmware updates for Jabra devices
- Settings configuration of Jabra devices.

The purpose of this document is to provide an overview of the Jabra Integration Service solution.

1.1 Target audience

The target audience for this document is Jabra integration partners and IT administrators responsible for Jabra device management.

1.2 About this guide

This guide is divided into sections, each with a specific topic. Each section can be read as a stand-alone text, but we encourage you to look through all sections. When applicable, notes, examples, and tips will be provided.

2 Overview

Jabra Integration Service is a Windows Desktop .NET application, it functions as a client, connecting to integration partner endpoints, see Figure 1 below:

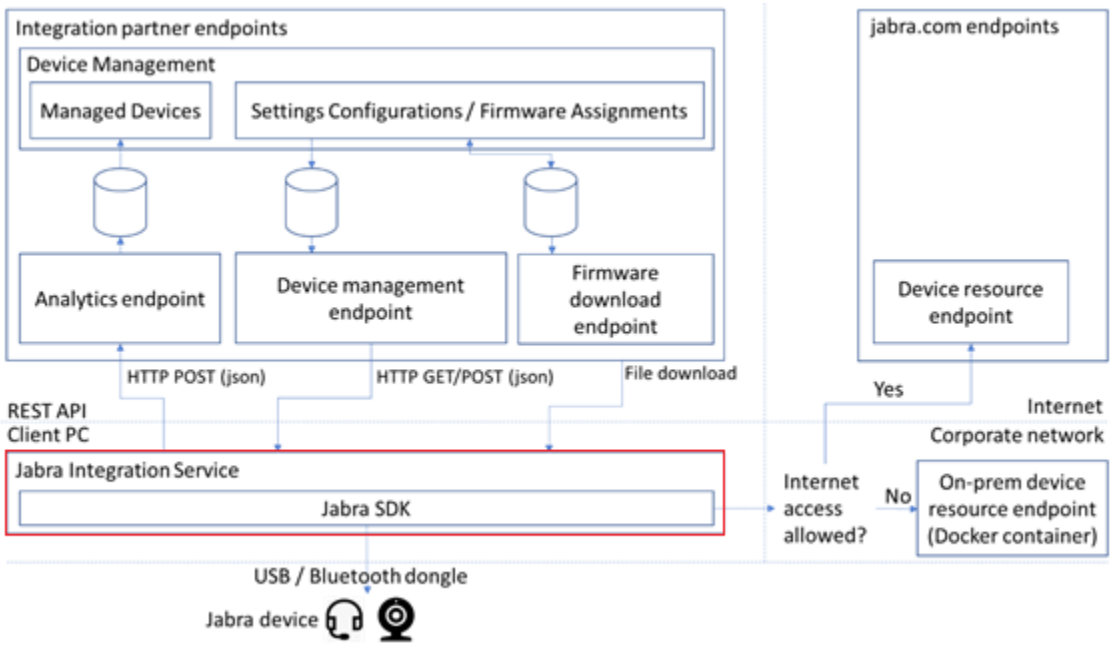


Figure 1 Overview, Jabra Integration Service

Jabra Integration Service connects to an analytics endpoint where it posts device and system information, it also connects to the device management endpoint where it gets configurations for the managed devices connected to the client PC.

The integration partner should provide the device management backend system and provide the necessary JSON documents (configurations) for the client.

Both endpoint URLs are partner-specific and shall be configured during the installation of the client application using MSI properties. The endpoints shall provide a RESTful API as described in the following sections.

3 Requirements

Client platform support:

- 32-bit Windows 7, or newer
- 64-bit Windows 7, or newer
- .NET Framework 4.7.2 or newer

Client installer:

- Windows Installer (.MSI file-based)
- Non-UI-installer
- Customized with MSI Properties.

Table 1 Requirements

4 Deployment

The Jabra Integration Service software component can be mass deployed, mass upgraded, or mass uninstalled using Microsoft SCCM or similar tools.

The following can be customized:

- Analytics backend endpoint (a URL).
  Default is empty, i.e. analytics is disabled. Can be either ‘http://’ or ‘https://’. Multiple analytics endpoints can be configured by separating each endpoint with a ‘|’ character.
  Installer property: ANALYTICSENDPOINT.

- Device management backend endpoint (a URL).
  Default is empty, i.e. device management is disabled. Can be either ‘http://’ or ‘https://’.
  Installer property: DEVICEMANAGEMENTENDPOINT.

- Username and Password
  If Username and Password is specified, they will be used to setup Basic Authentication against both of the above endpoints.
  Installer properties: USERNAME and PASSWORD

- Show/hide tray icon.
  Default is show (YES).
  Installer property: TRAYICON.

- Check for updates interval in minutes.
  Default is 10080 (7 days).
  Installer property: CHECKFORUPDATESINTERVALMINUTES.

- Partner-specific backend type.
  Default is (Default).
  Installer property: BACKENDTYPE.

- Partner-specific backend configuration options.
  Default is empty.
  Installer property: BACKENDCONFIGURATION.

- Jabra SDK backend configuration option.
  Default is empty.
  Installer property: DEVICECAPABILITIESENDPOINT.

4.1 Deployment Location

On a 64-bit machine Jabra Integration Service is installed in “`C:\Program Files (x86)\Jabra\Jabra Integration Service`”.

The client executable is called `JabraLoggingClient.exe`

4.2 Deployment example

To clarify deployment further we will use an example where three settings are customized:

1. Hide tray icon
2. Set the analytics and device management endpoints to https://myserver/api/myendpoint
3. Set the check for updates interval to 1 hour.

This example will look like this:

```
msiexec.exe /i JabraIntegrationServiceX64Setup.msi TRAYICON="NO"
ANALYTICSENDPOINT="https://myserver/api/myendpoint1|
https://myserver/api/myendpoint2"
DEVICEMANAGEMENTENDPOINT="https://myserver/api/myendpoint"
CHECKFORUPDATESINTERVALMINUTES=”60”
```

4.3 Jabra device SDK

The Jabra Integration Service client uses the Jabra SDK[1] as a library for Jabra device detection, firmware update, and device settings configuration.

By default, the Jabra SDK requires internet access to retrieve device metadata and device resources, however, from release 1.4 and onwards, the client supports the configuration of the Jabra SDK backend endpoint. This makes it possible to host the Jabra SDK backend on-premises using the Docker container technology[2].
With this deployment configuration, the Jabra SDK will not access the internet. Jabra makes regular updates of the Jabra Xpress and Jabra SDK backend Docker image on Docker Hub[3].

Please refer to section 10.6 and associated subsections starting on page 29 for further information.

[1] https://developer.jabra.com/site/global/sdks/windows/index.gsp
[2] https://www.docker.com/
[3] https://hub.docker.com/r/gnaudio/jabra-xpress-frontend

5 Reporting data

One of the purposes of the Jabra Integration Service component is to send data (‘http’ or ‘https’ post) to a backend endpoint after certain events.
The backend identifies the client’s POST request for the analytics endpoint by the User-Agent string: `"JabraLogging”`.

Data is posted when:

A device is attached (or the client is started)	Event = “DeviceAdded”
	Event = “CameraDeviceAdded”
	Event = “OtherDeviceAdded”
A device is removed	Event = “DeviceRemoved”
	Event = “CameraDeviceRemoved”
	Event = “OtherDeviceRemoved”
A call is started, and again when the call is ended	Event = “CallStatus”
A camera starts streaming	Event = “CameraStreamingStarted”
A camera stops streaming	Event = “CameraStreamingEnded”
People count changes (both when the camera is streaming and when idle)	Event = “PeopleCount”
Firmware update has completed	Event = “FirmwareUpdateCompleted”

Table 2 Reporting data posts

The data is formatted as JSON. The data is an HTTP POST.

An example of reporting data is shown below:

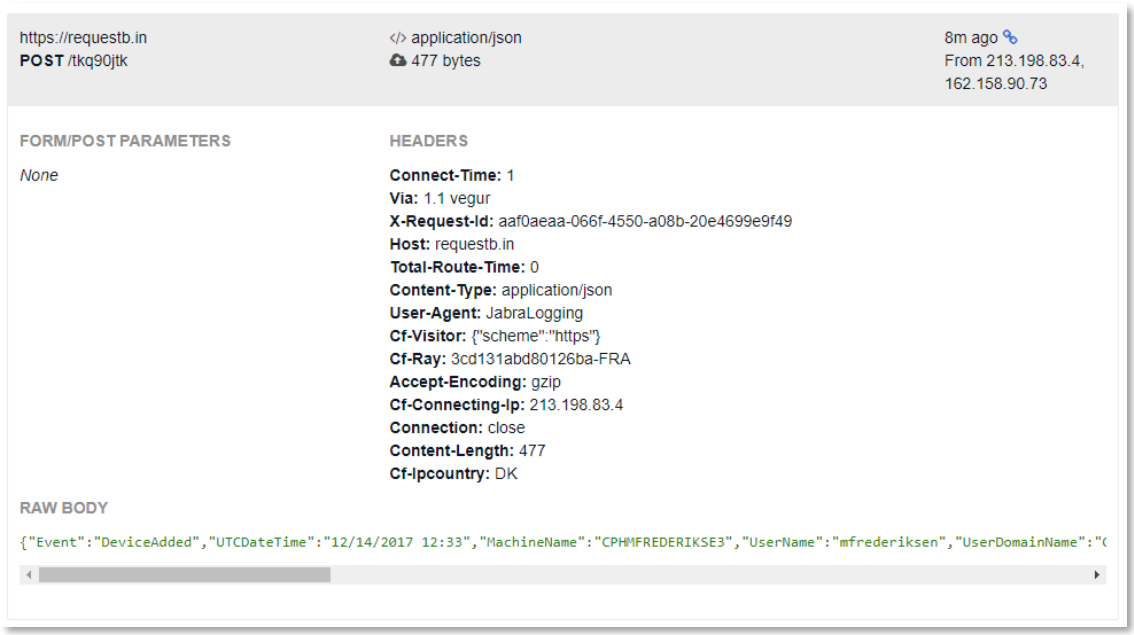


Figure 2 Reporting data example

5.1 Data format

The data format used for events is described in detail in an OpenAPI specification. This can be found in the document `Events.OpenApi.yml` that is part of the Jabra Integration Service release package.

The list of possible events described here may be extended in future releases of the Jabra Integration Service. The data included in each event may also be extended with new properties in the future. An endpoint that receives events should be able to handle new and unknown events as well as new properties for each event.

The subsections below list the data each SDK provides**.**

5.1.1 Jabra SDK

A list of attached Jabra devices with:

- Device name
- Firmware version
- Electronic serial number
- Connection Type
- For a wireless device:
    - Battery level (0-100)
    - Battery low (true/false)
    - Battery charging (true/false).
- Firmware update completed info
    - Firmware update completed status (Success/Failure)

- Firmware update failure description (in case of Failure).
- Basic call status (for all supported Jabra devices)
  - Call started
  - Call ended
    - Duration of the call measured in minutes.
    - The number of ‘mute’ button presses on the device during the call.
    - The number of ‘volume up’ button presses on the device during the call.
    - The number of ‘volume down’ button presses on the device during the call.
- Advanced call ended status with aggregated call data:
  - Average audio exposure during a call, measured in dB
  - Maximum audio exposure during a call, measured in dB
  - Average background noise during a call, measured in dB
  - Maximum audio exposure during a call, measured in dB
  - How many times the boom arm was badly positioned during a call
  - Malfunctioning microphone detected during a call.

5.1.2 Jabra PanaCast PeopleCount SDK

- Device name
- Firmware version
- Electronic serial number
- Streaming information (CameraStreamingStarted and CameraStreamingEnded events)
- Basic PeopleCount when counting changes (PeopleCount event)
- Aggregated PeopleCount information included in CameraStreamingEnded event
  - Number of faces detected at the beginning of the session
  - Number of faces detected at the end of the session
  - Minimum number of faces detected during streaming
  - Maximum number of faces detected during streaming
  - The average number of faces detected during streaming
  - Duration of the video streaming session (in minutes).

Please note, that the PeopleCount SDK only supports one connected Jabra PanaCast.

5.1.3 Devices from Other Vendors

The Jabra Integration Service as limited support for devices from a selected group of other vendors. The data published for these devices contains:

- Device name
- Windows instance ID
- Vendor ID
- Product ID

5.1.4 Client PC

- Computer name[4]
- Domain/User[5,6]
- Windows details[7]
- OS version[8]
- List of PC IP addresses[9]
- Jabra Integration Service version number.

[4] https://msdn.microsoft.com/en-us/library/system.environment.machinename(v=vs.110).aspx
[5] https://msdn.microsoft.com/en-us/library/system.environment.username(v=vs.110).aspx
[6] https://msdn.microsoft.com/en-us/library/system.environment.userdomainname(v=vs.110).aspx
[7] https://stackoverflow.com/a/39888998/600559
[8] https://msdn.microsoft.com/en-us/library/system.environment.osversion(v=vs.110).aspx
[9] https://stackoverflow.com/a/21155473/600559

5.2 Data examples

5.2.1 DeviceAdded and DeviceRemoved


```
Event: "DeviceAdded"
UTCDateTime: "2020-05-25 10:50:11Z"
MachineName: "CPHAPF10T4YX"
UserName: "lkjensen"
UserDomainName: "CORP"
OSFullName: "Microsoft Windows 10 Enterprise"
OSVersion: "6.2.9200.0"
JabraLoggingClientVersion: "1.0.0.0"
› IPAddresses: Array
⌄ JabraDevice: Object
    Name: "Jabra Evolve2 40"
    Firmware: "1.15.0"
    ElectronicSerialNumber: "A0000670E741"
    VendorId: 2830
    ProductId: 3648
    VariantType: "01-5E"
    BatteryLow: false
    Charging: false
    LevelInPercent: 0
    FirmwareUpdateCompletedStatus: null
    FirmwareUpdateFailureDecription: null
    CallStatus: null
    CallDurationInMinutes: null
    AudioExposureAvgDuringCall: null
    BackgroundNoiseAvgDuringCall: null
    MaxAudioExposureDuringCall: null
    MaxBackgroundNoiseDuringCall: null
    TimesBoomArmBadlyPositionedDur…: null
    MalfunctioningMicDetectedDurin…: null
    TimesMuteButtonPressedDuringCa…: null
    TimesVolumeUpButtonPressedDuri…: null
    TimesVolumeDownButtonPressedDu…: null
```


Example 1 DeviceAdded


```
Event: "DeviceRemoved"
UTCDateTime: "2020-05-25 10:53:01Z"
MachineName: "CPHAPF10T4YX"
UserName: "lkjensen"
UserDomainName: "CORP"
OSFullName: "Microsoft Windows 10 Enterprise"
OSVersion: "6.2.9200.0"
JabraLoggingClientVersion: "1.0.0.0"
› IPAddresses: Array
⌄ JabraDevice: Object
    Name: "Jabra Evolve2 40"
    Firmware: "1.15.0"
    ElectronicSerialNumber: "A0000670E741"
    VendorId: 2830
    ProductId: 3648
    VariantType: "01-5E"
    BatteryLow: false
    Charging: false
    LevelInPercent: 0
    FirmwareUpdateCompletedStatus: null
    FirmwareUpdateFailureDecription: null
    CallStatus: null
    CallDurationInMinutes: null
    AudioExposureAvgDuringCall: null
    BackgroundNoiseAvgDuringCall: null
    MaxAudioExposureDuringCall: null
    MaxBackgroundNoiseDuringCall: null
    TimesBoomArmBadlyPositionedDur…: null
    MalfunctioningMicDetectedDurin…: null
    TimesMuteButtonPressedDuringCa…: null
    TimesVolumeUpButtonPressedDuri…: null
    TimesVolumeDownButtonPressedDu…: null
```

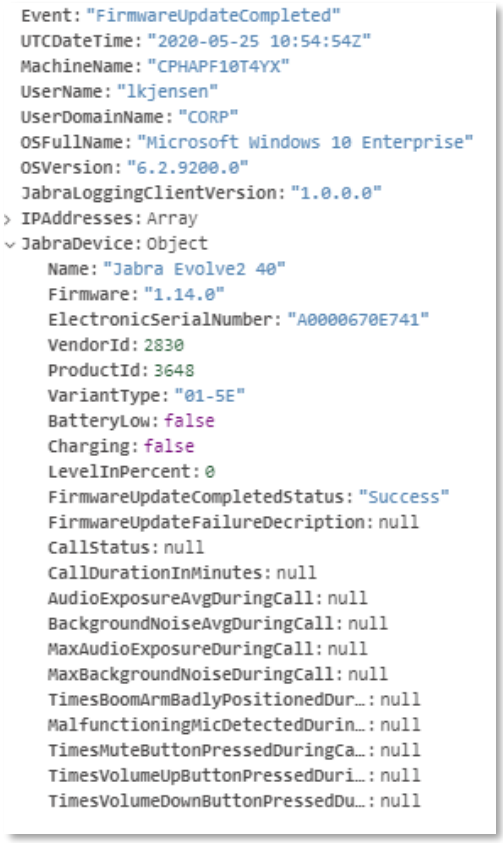

Example 2 DeviceRemoved

5.2.2 ConversationStarted

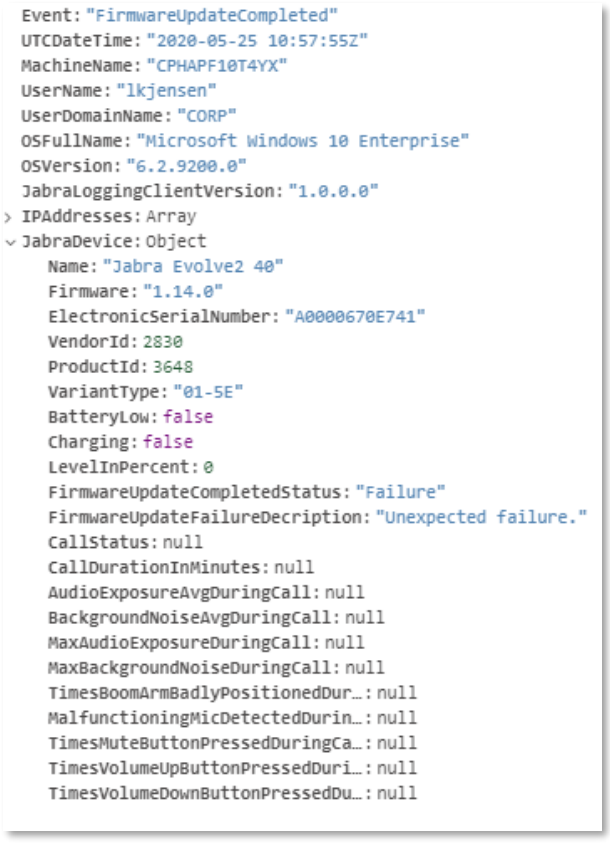

```
Event: "ConversationStarted"
UTCDateTime: "2019-09-02 09:02:23Z"
MachineName: "CPHAPF10T4YX"
UserName: "lkjensen"
UserDomainName: "CORP"
OSFullName: "Microsoft Windows 10 Enterprise"
OSVersion: "6.2.9200.0"
JabraLoggingClientVersion: "1.0.0.0"
› IPAddresses: Array
› Conversation: Object
ActiveAudioDevice: "Speakers (Jabra Engage 50)"
IsActiveAudioDeviceCertified: false
⌄ AudioDevices: Array
    0: "Speakers (Jabra Engage 50)"
    1: "Speaker/HP (Realtek High Definition Audio)/Microphone Array (Realtek H..."
```


Example 3 ConversationStarted

5.2.3 FirmwareUpdateCompleted - Success and Failure



Example 4 FirmwareUpdateCompleted (Success)



Example 5 FirmwareUpdateCompleted (Failure)

5.2.4 CallStatus - CallStarted and CallEnded


```
Event: "CallStatus"
UTCDateTime: "2020-05-25 12:07:27Z"
MachineName: "CPHAPF10T4YX"
UserName: "lkjensen"
UserDomainName: "CORP"
OSFullName: "Microsoft Windows 10 Enterprise"
OSVersion: "6.2.9200.0"
JabraLoggingClientVersion: "1.0.0.0"
› IPAddresses: Array
⌄ JabraDevice: Object
    Name: "Jabra Evolve2 40"
    Firmware: "1.15.0"
    ElectronicSerialNumber: "A0000670E741"
    VendorId: 2830
    ProductId: 3648
    VariantType: "01-5E"
    BatteryLow: false
    Charging: false
    LevelInPercent: 0
    FirmwareUpdateCompletedStatus: null
    FirmwareUpdateFailureDecription: null
    CallStatus: "CallStarted"
    CallDurationInMinutes: null
    AudioExposureAvgDuringCall: null
    BackgroundNoiseAvgDuringCall: null
    MaxAudioExposureDuringCall: null
    MaxBackgroundNoiseDuringCall: null
    TimesBoomArmBadlyPositionedDur…: null
    MalfunctioningMicDetectedDurin…: null
    TimesMuteButtonPressedDuringCa…: null
    TimesVolumeUpButtonPressedDuri…: null
    TimesVolumeDownButtonPressedDu…: null
```


Example 6 CallStatus (CallStarted)


```
Event: "CallStatus"
UTCDateTime: "2020-05-25 12:07:33Z"
MachineName: "CPHAPF10T4YX"
UserName: "lkjensen"
UserDomainName: "CORP"
OSFullName: "Microsoft Windows 10 Enterprise"
OSVersion: "6.2.9200.0"
JabraLoggingClientVersion: "1.0.0.0"
› IPAddresses: Array
⌄ JabraDevice: Object
    Name: "Jabra Evolve2 40"
    Firmware: "1.15.0"
    ElectronicSerialNumber: "A0000670E741"
    VendorId: 2830
    ProductId: 3648
    VariantType: "01-5E"
    BatteryLow: false
    Charging: false
    LevelInPercent: 0
    FirmwareUpdateCompletedStatus: null
    FirmwareUpdateFailureDecription: null
    CallStatus: "CallEnded"
    CallDurationInMinutes: 0.17
    AudioExposureAvgDuringCall: 0
    BackgroundNoiseAvgDuringCall: 0
    MaxAudioExposureDuringCall: 0
    MaxBackgroundNoiseDuringCall: 0
    TimesBoomArmBadlyPositionedDur…: 0
    MalfunctioningMicDetectedDurin…: false
    TimesMuteButtonPressedDuringCa…: 0
    TimesVolumeUpButtonPressedDuri…: 0
    TimesVolumeDownButtonPressedDu…: 0
```


Example 7 CallStatus (CallEnded)

5.2.5 CameraDeviceAdded and CameraDeviceRemoved


```
Event: "CameraDeviceAdded"
UTCDateTime: "2019-08-28 11:51:54Z"
MachineName: "CPHAPF10T4YX"
UserName: "lkjensen"
UserDomainName: "CORP"
OSFullName: "Microsoft Windows 10 Enterprise"
OSVersion: "6.2.9200.0"
JabraLoggingClientVersion: "1.0.0.0"
IPAddresses: Array
    0: "10.145.44.82"
    1: "192.168.56.1"
JabraCameraDevice: Object
    Name: "Jabra PanaCast"
    Firmware: "C0.3.7.18"
    ElectronicSerialNumber: "P31910760PC3"
    VendorId: 11155
    ProductId: 3
    PeopleCount: null
    IsStreaming: null
    FirstPeopleCountDuringStreaming: null
    LastPeopleCountDuringStreaming: null
    MaxPeopleCountDuringStreaming: null
    MinPeopleCountDuringStreaming: null
    AvgPeopleCountDuringStreaming: null
    StreamingDurationInMinutes: null
```


Example 8 CameraDeviceAdded


```
Event: "CameraDeviceRemoved"
UTCDateTime: "2019-08-28 11:53:46Z"
MachineName: "CPHAPF10T4YX"
UserName: "lkjensen"
UserDomainName: "CORP"
OSFullName: "Microsoft Windows 10 Enterprise"
OSVersion: "6.2.9200.0"
JabraLoggingClientVersion: "1.0.0.0"
IPAddresses: Array
    0: "10.145.44.82"
    1: "192.168.56.1"
JabraCameraDevice: Object
    Name: "Jabra PanaCast"
    Firmware: "C0.3.7.18"
    ElectronicSerialNumber: "P31910760PC3"
    VendorId: 11155
    ProductId: 3
    PeopleCount: null
    IsStreaming: null
    FirstPeopleCountDuringStreaming: null
    LastPeopleCountDuringStreaming: null
    MaxPeopleCountDuringStreaming: null
    MinPeopleCountDuringStreaming: null
    AvgPeopleCountDuringStreaming: null
    StreamingDurationInMinutes: null
```


Example 9 CameraDeviceRemoved

5.2.6 CameraStreamingStarted and CameraStreamingEnded


```
Event: "CameraStreamingStarted"
UTCDateTime: "2019-08-28 11:52:19Z"
MachineName: "CPHAPF10T4YX"
UserName: "lkjensen"
UserDomainName: "CORP"
OSFullName: "Microsoft Windows 10 Enterprise"
OSVersion: "6.2.9200.0"
JabraLoggingClientVersion: "1.0.0.0"
IPAddresses: Array
    0: "10.145.44.82"
    1: "192.168.56.1"
JabraCameraDevice: Object
    Name: "Jabra PanaCast"
    Firmware: "C0.3.7.18"
    ElectronicSerialNumber: "P31910760PC3"
    VendorId: 11155
    ProductId: 3
    PeopleCount: null
    IsStreaming: null
    FirstPeopleCountDuringStreaming: null
    LastPeopleCountDuringStreaming: null
    MaxPeopleCountDuringStreaming: null
    MinPeopleCountDuringStreaming: null
    AvgPeopleCountDuringStreaming: null
    StreamingDurationInMinutes: null
```


Example 10 CameraStreamingStarted


```
Event: "CameraStreamingEnded"
UTCDateTime: "2019-08-28 11:53:41Z"
MachineName: "CPHAPF10T4YX"
UserName: "lkjensen"
UserDomainName: "CORP"
OSFullName: "Microsoft Windows 10 Enterprise"
OSVersion: "6.2.9200.0"
JabraLoggingClientVersion: "1.0.0.0"
IPAddresses: Array
    0: "10.145.44.82"
    1: "192.168.56.1"
JabraCameraDevice: Object
    Name: "Jabra PanaCast"
    Firmware: "C0.3.7.18"
    ElectronicSerialNumber: "P31910760PC3"
    VendorId: 11155
    ProductId: 3
    PeopleCount: null
    IsStreaming: null
    FirstPeopleCountDuringStreaming: 5
    LastPeopleCountDuringStreaming: 5
    MaxPeopleCountDuringStreaming: 5
    MinPeopleCountDuringStreaming: 1
    AvgPeopleCountDuringStreaming: 2.54
    StreamingDurationInMinutes: 1
```


Example 11 CameraStreamingEnded

5.2.7 PeopleCount

Example 12 PeopleCount

6 Firmware Update Completed event

From release 1.7 and onwards, Jabra Integration Service posts an event when the firmware update procedure is finalized.
If the firmware update was successful the `FirmwareUpdateCompletedStatus` value will be `Success`. If the update failed, the value will be `Failure`. For all other event types, the value will be `null`.

In case of failure, the value of `FirmwareUpdateFailureDescription` provides additional information about the failure. In all other cases, the value will be `null`.
Possible failure descriptions are listed below:

- 'Firmware file does not match the device'
- 'Headset needs to be docked'
- 'Bluetooth headset not connected to dongle'
- 'Region not found in the firmware file'
- 'Language not found in firmware file'
- 'Downgrade is not allowed for this product'
- 'Battery level too low for firmware update'
- 'Unexpected failure'.

7 Call Status events

As described in the previous sections, Jabra Integration Service posts call status events to the analytics endpoint when a call is started and when the call is ended. The call status event reflects a state change in the Jabra device between an idle state and an active call state as illustrated in Figure 3 below:

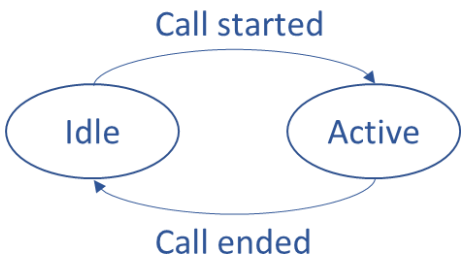


Figure 3 Call status event reflects a change of state

The change of state can e.g. be triggered by a mobile phone call, a desk phone call, or by any softphone that integrates with the Jabra device; either through Jabra Direct, a 3rd party softphone integration or via standard HID telephony signaling e.g. Skype for Business and Microsoft Teams.

It is important to note, that call status events are triggered by the device and that the payload included in an event only contains device-specific information and general system information such as Windows version, username, and IP addresses - i.e. no information about the source of the call.

Correlating call status events from a Jabra device to an actual softphone call, as an example, is a server-side responsibility.

Values in the CallStatus event will be `null` if they are either not applicable or if no events were received. As an example, the `TimesMuteButtonPressedDuringCall` is `null` if no 'mute' button events have been detected.

7.1 Aggregated call data

For all Jabra devices with mute and volume control, the following data is included as part of the CallStatus -> CallEnded event:

- Duration of the call, in minutes
- The number of 'mute' button presses during a call
- The number of 'volume up' button presses during a call
- The number of 'volume down' button presses during a call.

7.2 Advanced call data

It is possible to include additional call data in the CallStatus -> CallEnded event if you use one of the following headsets:

- Jabra Engage 50
- Jabra Engage 65 / 75 (latest firmware version)
- Jabra Evolve2 series

The headset can send a lot of events (background noise, audio exposure, badly positioned boom arm, and malfunctioning microphone detection) during a call, but to reduce network traffic and backend storage, the Jabra Integration Service aggregates the data during the call and only sends one event with all the aggregated data at the end of the call.

The data format is described in section 5.1 and associated subsections, go to page 11 for further information.

8 Camera events

The Jabra PanaCast is a plug-and-play device, designed to improve meetings in small meeting rooms by using three 13-megapixel cameras and real-time video stitching to give a full 180° view.

Jabra Integration Service version 1.5 and onwards supports the PeopleCount feature of the Jabra PanaCast. Once a Jabra PanaCast is connected to the Jabra Integration Service the system starts detecting faces in the meeting room (up to 19 feet).

PeopleCount events are posted to the backend whenever the count changes or otherwise every 10 minutes (heartbeat interval). PeopleCount events are sent both when the Jabra PanaCast is idle and when it is streaming video.

8.1 Meeting room utilization

The basic PeopleCount event is an integer representing the number of detected faces in the room. Storing these events in a backend system allows for historical analysis of meeting room utilization.

Please note, that this use case requires a permanent setup with a dedicated PC and a connected Jabra PanaCast.

8.2 Meeting participants count

During a video streaming session, the Jabra Integration Service aggregates PeopleCount data and includes the following information in the CameraStreamingEnded event:

- Number of people detected at the beginning of the streaming session
- Number of people detected at the end of the streaming session
- Maximum number of people detected during the streaming session
- Minimum number of people detected during the streaming session
- The average number of people detected during the streaming session
- Duration of the streaming session in minutes.

Knowing the number of participants in individual meetings can help optimize the floor layout in the office space.

9 Firmware update and settings configuration

Once installed, the client polls the device management endpoint for configuration updates. This is done whenever a Jabra device is connected or whenever the check for updates interval elapses.

The backend identifies the client's GET/POST request for the device management endpoint and the file download request for the firmware file endpoint by the User-Agent string.
An example is shown below:

```
"JabraClient/1.0 (Vendor=0B0E; Product=2465; Variant=01-3D;
ESN=501AA5D6337F; Firmware=2.0.0; Version=1.4.4735.0)"
```

The `JabraClient/1.0` part of the User-Agent string refers to the client/backend protocol (API) version. This protocol/API is defined by the format of the User-Agent string and the format of the configuration JSON.
If the User-Agent format changes or if the configuration JSON format changes in a non-backward compatible way, the major component of the version number is incremented. If a backward-compatible change is made to the configuration JSON, such as adding a new element, then the minor component is incremented.

The exact Jabra device model/type is determined by `Vendor=0B0E`, `Product=2465`, and `Variant=01-3D` information.

The connected devices are uniquely identified by their electronic serial number `ESN=501AA5D6337F`.

The firmware version of the connected Jabra device is determined by `Firmware=2.0.0`.

The `Version=1.4.4735.0` part of the User-Agent string refers to the version of the JabraLoggingClient.exe executable (identical to the Jabra Integration Service version number).

In response to the GET/POST request, the endpoint returns a configuration (JSON document) containing the managed firmware version, a download URL for that firmware, and a list of managed settings.

```
{
  "firmware": {
    "version": "2.0.0",
    "downloadUrl": "Firmware/Jabra-Evolve-75-2.0.0.zip"
  },
  "settings": {
    "AUDIO_PROTECTION": "5",
    "WIRELESS_RANGE": "3"
  }
}
```

The download URL can be relative as shown in the example above or an absolute URL starting with either ‘http:’, ‘https:’, or ‘ftp:’.

Assuming that the device management endpoint is configured to `https://myserver/api/myendpoint` then the complete download URL in this example will be `https://myserver/api/myendpoint/Firmware/Jabra-Evolve-75-2.0.0.zip`

The `firmware` and `settings` elements are mandatory, but they can be left empty to indicate that there are no elements to manage.

The firmware file is downloaded to the PC before the firmware update procedure is initiated. The device settings are not persisted on the PC as they are applied immediately.

9.1 API documentation

As part of the client release, two .xml files (documents) provide the details of supported devices and supported settings: ‘SettingsDefinition-Jabra.xml‘ and ‘Mapping.xml’, see below:

9.1.1 SettingsDefinition-Jabra.xml

The SettingsDefinition-Jabra.xml file contains the complete list of device settings configuration options in the given release. An example entry is shown below:

```
    <SettingDefinition Target="Jabra" Name="WIRELESS_RANGE" UserFriendlyName="Wireless
signal range" ValueType="String" ValueConstraintType="Enumeration" DefaultValue="1"
RequiresReboot="true" AllowsNull="false">
        <Description>Select the wireless range between the headset and base. If you
experience signal interference from other wireless devices, select Medium or
Short.</Description>
        <UserFriendlyDescription>Select the wireless range between the headset and base. If
you experience signal interference from other wireless devices, select Medium or
Short.</UserFriendlyDescription>
        <SupportedValues>
            <SupportedValue Value="1" UserFriendlyValue="Long" />
            <SupportedValue Value="2" UserFriendlyValue="Medium" />
            <SupportedValue Value="3" UserFriendlyValue="Short" />
        </SupportedValues>
    </SettingDefinition>
```

The “Name” of the setting is the unique identifier for a setting. The identifier along with the desired “Value” should be provided in the “settings” list of the configuration JSON as shown below:

```
"settings": {
  "WIRELESS_RANGE": "2"
}
```

The user-friendly names are intended to be used by the integration partner device management system when presenting the configuration options in a UI. When creating the device management configurations, it is important to provide the actual “Value” values and not the “UserFriendlyValue” values.

9.1.2 Mapping.xml

The mapping.xml file contains the complete list of devices supported by the given client release. The settings entry indicates which settings are supported by a given device. An example is shown below:

```
<device name="Jabra Evolve 30 II Stereo">
  <variants>
    <variant vid="2830" pid="42249" variantType="01-40" />
    <variant vid="2830" pid="786" variantType="01-40" />
    <variant vid="2830" pid="787" variantType="01-40" />
  </variants>
  <settings>
    <setting name="AUDIO_PROTECTION_1" />
    <setting name="RINGTONE_IN_HEADSET_SOFTPHONE" />
  </settings>
</device>
```

The Jabra USB vendor id (vid) is 2830 (0x0B0E), the device USB product id (pid) and the variant (VariantType) are model specific. Variants of the same Jabra product are listed under variants.

When creating a device management UI, it is not necessary to present any of the variants separately as they all share the same firmware and settings.

9.2 Region and language selection

Some settings are used to select the desired display language pack, voice announcements language pack, or simply the desired language as these are (for some devices) binary files used during the firmware update process.

For these settings to take any effect it is required to also provide a firmware file URL as part of the configuration JSON i.e., the firmware element of the JSON configuration cannot be empty.
The relevant settings are listed below:

```
SCREEN_REGION
SCREEN_REGION_1
VOICE_ANNOUNCEMENTS_LANGUAGE_1
VOICE_ANNOUNCEMENTS_LANGUAGE_2
VOICE_ANNOUNCEMENTS_LANGUAGE_4
VOICE_ANNOUNCEMENTS_REGION.
```

9.3 Configuration of device events

Device events are enabled for all devices by default. For example every time a call is started and ended on a Jabra device a CallStatus event is posted to the analytics endpoint.

Device events can be disabled for individual devices by adding the settings like the below to the settings JSON element:

```
"settings": {
    "EVENT_CALL_STATUS": "0"
  }
```

The following event settings are supported as part of the device configuration:

```
EVENT_CALL_STATUS
EVENT_DEVICE_DETECTION
EVENT_DEVICE_CONFIGURATION
EVENT_PEOPLE_COUNT
EVENT_CAMERA_STREAMING_STATUS
```

9.4 Allow firmware downgrade

Firmware downgrade is not allowed by default, however, the default behavior can be changed by including the below setting to the settings JSON element:

```
"settings": {
    "ALLOW_DOWNGRADE": "1"
  }
```

10 Deployment and backend configuration

The Jabra Integration Service can be mass deployed/upgraded/uninstalled using Microsoft SCCM or similar tools. The default out-of-the-box deployment configures the client to run with the default behavior.
However, for selected Jabra partners the Jabra Integration Service provides specific customizations that slightly change the behavior of the client to accommodate easy integration with existing device management systems.

10.1 Deployment for default backend

The most basic usage of the Jabra Integration Service MSI package is to enable the analytics part by only providing an analytics endpoint as shown below:

```
msiexec.exe /i JabraIntegrationServiceX64Setup.msi TRAYICON="NO"
ANALYTICSENDPOINT="https://myserver/api/myendpoint"
```

In this example, the device management features (firmware update and device settings configuration) are completely disabled. This basic deployment method is useful for getting an overview of all Jabra devices and installed firmware versions.

Adding a device management endpoint as shown below enables pushing out the latest device firmware and configuring the devices as desired:

```
msiexec.exe /i JabraIntegrationServiceX64Setup.msi TRAYICON="NO"
ANALYTICSENDPOINT="https://myserver/api/myendpoint"
DEVICEMANAGEMENTENDPOINT="https://myserver/api/myendpoint"
```

10.2 Deployment in a PowerSuite environment

PowerSuite[10] by Unify Square offers Jabra device management features such as device overview and firmware and device configuration management.

Deployment of Jabra Integration Service in a PowerSuite environment requires configuration of the analytics endpoint as a minimum. This will enable the device management overview to be updated with known Jabra devices. To enable firmware update and device configuration the device management endpoint also needs to be configured as shown below:

```
msiexec.exe /i JabraIntegrationServiceX64Setup.msi TRAYICON="NO"
ANALYTICSENDPOINT="https://myserver/DeviceMgmt"
DEVICEMANAGEMENTENDPOINT="https://myserver/DeviceMgmt"
```

The actual endpoints are typically one of the **Listen Addresses** configured in PowerSuite under Configuration -> General -> Device Management.

The MSI property `BACKENDTYPE` can be set to `PowerSuite` but this is not required as the PowerSuite behavior is identical to the default behavior.

[10] https://www.unifysquare.com/unified-communications-solutions

10.3 Deployment in an OVOC environment

One Voice Operations Center (OVOC[11]) by AudioCodes offers Jabra device management features such as device overview and firmware and device configuration management.

Deployment of Jabra Integration Service in an OVOC environment only requires configuration of the device management endpoint. This will enable the Monitor -> Devices Status view to be updated with known Jabra devices as well as enable firmware update and device configuration. An example is shown below:

```
msiexec.exe /i JabraIntegrationServiceX64Setup.msi TRAYICON="NO"
DEVICEMANAGEMENTENDPOINT="http://OVOC_IP/" BACKENDTYPE=”Ovoc”
CECKFORUPDATESINTERVALMINUTES="10"
```

Please note: The MSI property `BACKENDTYPE` must be configured to `Ovoc`.

10.4 Query strings for endpoint URLs

Jabra Integration Service version 1.8 and onwards makes it possible to configure all endpoints with URLs that contain query strings. This is useful for simple token-based authentication. As an example, it is possible to configure the device management endpoint as shown below:

```
msiexec.exe /i JabraIntegrationServiceX64Setup.msi TRAYICON="NO"
ANALYTICSENDPOINT="https://myserver/api/myendpoint"
DEVICEMANAGEMENTENDPOINT="https://myserver/api/myendpoint/?Token=XXXX"
```

The query string is appended to the firmware download URL provided in the JSON configuration. Assuming that the downloadUrl is configured as
`"downloadUrl": "Firmware/Jabra-Evolve-75-2.0.0.zip"`
then the resulting download URL would look like this:

`https://myserver/api/myendpoint/Firmware/Jabra-Evolve-75-2.0.0.zip?Token=XXXX`

10.5 Endpoint Security

Jabra Integration Service supports authentication against all endpoints using API tokens as described in section 10.4. From version 1.11 and onwards it also supports Basic Authentication. Basic Authentication can be configured during installation with the `USERNAME` and `PASSWORD` properties:

```
msiexec.exe /i JabraIntegrationServiceX64Setup.msi
ANALYTICSENDPOINT="https://myserver/api/myendpoint" USERNAME="XXX"
PASSWORD="YYY"
```

When Basic Authentication is used the specified credentials will be used against all configured endpoints – both analytics and device management.

[11] https://www.audiocodes.com/solutions-products/products/management-products-solutions/one-voice-operations-center

10.6 Self-hosting of the Jabra SDK backend

The Jabra SDK backend provides Jabra device metadata required by the Jabra SDK. By default, the Jabra SDK will access the Jabra-hosted SDK backend on the internet, but if an integration partner or a company wants to self-host the Jabra SDK backend it is possible using the Docker container technology. This section assumes that the reader is familiar with the concepts of Docker images, Docker containers, and docker-compose.

Jabra has adopted the Docker (or rather Linux Container) technology[12] for the Jabra Xpress device management solution and all the Xpress backend services are published as Docker images on Docker Hub[13]. Setting up a self-hosted Jabra SDK backend is much easier than setting up the entire Xpress backend as the Jabra SDK backend is a single service that needs to be configured.

10.6.1 docker-compose.yml

Assuming that a Docker Engine is installed and running on e.g. an Ubuntu Linux distribution, the following boilerplate docker-compose.yml file defines the configuration of the Jabra SDK backend:

```
version: '3.4'

services:

  sdkbackend:
    image: gnaudio/jabra-xpress-sdkbackend:v4.0.4337
    environment:
    - "Storage:SDKCopyUrl=http://myserver:82"
    ports:
    - "82:80"
```

In a terminal window `cd` to the directory containing the docker-compose.yml file and execute the following command:

```
$ docker-compose up
```

The Jabra SDK backend can now be accessed with a browser from http://myserver/.

Explore the Web APIs using Swagger. The only API needed is the /v4/deviceconfiguration. Append this part to the base URL to get the absolute device capabilities endpoint:

```
"http://myserver/v4/deviceconfiguration"
```

It is recommended to experiment with the Docker container deployment on a simple Linux desktop system configuring the server URL to http and localhost, before deploying on a larger scale.

Please note: The SDK backend image will be updated on regular basis with the latest Jabra device metadata (device capabilities descriptions) and thus the version in the above example needs to be updated to the latest version. Consult the Jabra Xpress documentation on Docker Hub for the latest released version.

[12] https://www.docker.com/

[13] https://hub.docker.com/r/gnaudio/jabra-xpress-frontend

10.6.2 Configuring the Jabra SDK backend endpoint

Once the Jabra SDK backend server is accessible, the Jabra Integration Service needs to be deployed with the `DEVICECAPABILITIESENDPOINT` MSI property set to the URL from the docker-compose.yml file as shown below:

```
msiexec.exe /i JabraIntegrationServiceX64Setup.msi TRAYICON="NO"
ANALYTICSENDPOINT="https://myserver/api/myendpoint"
DEVICEMANAGEMENTENDPOINT="https://myserver/api/myendpoint"
DEVICECAPABILITIESENDPOINT =https://myserver/v4/deviceconfiguration
```

11 Log files for troubleshooting

Log-files for the client is created in this folder:

`C:\Users\<username>\AppData\Roaming\Jabra\JabraIntegrationService`

Log-files for the firmware updater is created in this folder:

`C:\Users\<username>\AppData\Local\Jabra\JabraCmdlineFwUpdater`

When requesting Jabra support these log files are essential.

12 Jabra resources

The purpose of this section is to explain how to retrieve relevant Jabra resources when adding support for Jabra devices in a device management system.
The target audience for this section is Jabra integration partners, however, it can also be useful for IT administrators.

The Jabra resources are:

- Jabra firmware files
  - Type: .zip files containing an info.xml and relevant binary files
- Jabra firmware release notes
  - Type: string
- Jabra product images
  - Type: .png
- Jabra device identification information
  - Device name
    - Type: string
  - Vendor id (vid)
    - Type: integer
    - USB vendor id (as defined by usb.org)
    - 2830 (0x0B0E) for Jabra devices.
  - Product id (pid)
    - Type: integer
    - USB product id (as defined by usb.org)
    - The value is specific to the device model but can be shared between device variants of the same model. As an example, an MS variant would have a dedicated pid.
  - Variant type
    - Type: string
    - Jabra proprietary variant type
    - The first part of the string represents a device type. The relevant types are:
      - “00” = base (e.g. Jabra Pro 9450 base)
      - “01” = headset (e.g. Jabra Evolve 75)
      - “04” = dongle (Bluetooth adapter, e.g. Jabra Link 370)
      - “07” = USB (USB adapter, e.g. Jabra Link 860)
      - “08” = speakerphone (e.g. Jabra Speak 710).

- The second part represents a specific Jabra model variant within the scope of the device type e.g.:
  - “01-03” is the mono variant of the Jabra BIZ 2400 headset
  - “08-03” is a Jabra Speak 710 speakerphone.

Please note: vid, pid, and variant type are all required to uniquely identify the specific model type/variant of a Jabra device.

12.1 Retrieving resources

The recommendation and intended use are to download the relevant resources and then add/integrate these resources into the relevant device management system as depicted in Figure 1 on page 7.

For the time being, this is recommended as a manual step that should be considered whenever a new Jabra Integration Service version is released.

12.2 Resource-mapping.xml

To download the appropriate firmware file for a given Jabra device model, it is important to consult the **resource-mapping.xml** file (part of the Jabra Integration Service release package). This file lists all the supported Jabra devices and variants as well as relevant firmware download URLs and firmware release notes.

An example entry from the resource-mapping.xml is shown below:

```
  <device name="Jabra Engage 50 Stereo"
imageUrl="https://sdkbackend.jabra.com/v4/product/PID_4006/image?type=01-4E"
thumbnailUrl="https://sdkbackend.jabra.com/v4/product/PID_4006/thumbnail80?type=01-4E">
    <variants>
      <variant vid="2830" pid="16385" variantType="01-4E" />
      <variant vid="2830" pid="16386" variantType="01-4E" />
      <variant vid="2830" pid="16390" variantType="01-4E" />
    </variants>
    <fwVersions>
      <fwVersion version="1.22.0"
firmwareFile="https://sdkbackend.jabra.com/v3/download/4006/1.22.0">
        <releaseNote />
      </fwVersion>
      <fwVersion version="1.24.0"
firmwareFile="https://sdkbackend.jabra.com/v3/download/4006/1.24.0">
        <releaseNote>• New feature: Stereo in conversation mode*
• New feature: Balanced voice
• Updated: Busylight function can be customized using Jabra SDK
• Updated: Sidetone adjustments
• Performance and stability improvements

*Dependent on softphone support
</releaseNote>
      </fwVersion>
    </fwVersions>
  </device>
```

12.2.1 Variants

The combination of Vendor Id (vid), Product Id (pid), and variant type uniquely identifies the type of device. In the example above the pids are different between variants. In other cases, the VariantType changes but the pids are the same.

All variants within a device element share the same resources; the same name, the same images (the above example is a stereo/duo headset), and the same firmware and firmware release notes.

12.3 Firmware file download

It is strongly recommended to use PowerShell, cURL or wget to download the firmware files. Below is an example of how to download a firmware file using PowerShell:

1. Download the firmware file to the `$download` variable. The highlighted URL below is obtained from the resource-mapping.xml:

   ```
   PS C:\Temp>$download = Invoke-WebRequest -Uri
   'https://firmware.jabra.com/v3/download/4006/1.24.0' -Headers
   @{"accept"="application/octet-
   stream";"ClientId"="C6raG/RknoKJ8PnFw6SLTc/AYv6yqzobKiWpq3yn6JQ="}
   ```

2. Write variable content to a file with the appropriate filename:

   ```
   PS C:\Temp>[System.IO.File]::WriteAllBytes((Get-Location -
   PSProvider FileSystem).ProviderPath+"\"+$download.headers["Content-
   Disposition"].Split("=")[1],$download.content)
   ```

Jabra device firmware files can be downloaded through a Web API. There is a Swagger UI for testing purposes at this site:
https://firmware.jabra.com/swagger/index.html#!/Firmware/V4DownloadByProductIdByVersionGet

However, the Swagger UI should NOT be used for actual file download.

12.3.1 ClientId

The ClientId in the above PowerShell example is the Jabra SDK test ClientId which is available from the Swagger site.
Obtain a dedicated Jabra SDK ClientID by registering at https://developer.jabra.com/

12.4 Product image download

Jabra product images are downloaded slightly differently from firmware files and it does not require a ClientId. However, the download URL is still obtained from the resource-mapping.xml file, see the example below where an image is downloaded to a file with the corresponding product name:

```
PS C:\Temp>Invoke-WebRequest
'https://devicecapabilities.jabra.com/v4/product/PID_4006/image?type=01-
4E' -OutFile Jabra_Engage_50_Stereo.png
```

12.5 Using the resources

The device management backend can obtain Jabra device identification info from the User-Agent string transmitted in the GET/POST requests. An example is shown here:

```
"JabraClient/1.0 (Vendor=0B0E; Product=2465; Variant=01-3D;
ESN=501AA5D6337F; Firmware=2.0.0; Version=1.4.4735.0)"
```

Based on the mapping.xml, the resource-mapping.xml, and the User-Agent string the backend can provide the correct Jabra resources for the relevant device.

13 FAQ

What is the Jabra Integration Service software and what does it do?
Jabra Integration Service enables you to manage Jabra devices (change settings and firmware). Moreover, the software provides data relevant for tracking call quality issues, as well as an overview of USB devices in your environment.
Is it possible to deploy the software on demand?
Deployment is controlled by your deployment software, e.g. SCCM.
Is this something that can be committed to long-term? (i.e. no product withdrawals/changes planned)
Yes, Jabra guarantees that the software will be maintained in a long-term perspective.
Who writes the software? How reliable/secure is it? How many updates? What happens if there is a bug?
Jabra develops and supports Jabra Integration Service software. Bugs etc. are either reported to the integration partner or directly to Jabra Support. Currently, we expect to release a new client with new product support 2-3 times a year. It is only necessary to update the client if new products need to be managed.
Can it be packaged easily and at a low cost for mass deployment?
Yes, it is a standard MSI package.
Can we change the configuration remotely?
Yes, device firmware update and settings configuration can be changed remotely. Configuration of the client software itself happens during the installation using MSI properties.
How much disk space, CPU load, and RAM are required?
~10MB on disk, ~20MB RAM, and ~0% CPU load when idle. Very small load when a device is connected/disconnected or updated.
Is there any visible pop-up or notification to the users?
Yes, there is a UI pop-up, but only when a firmware update is initiated as it is a critical operation requiring the attention of the end-user. Change of device settings happens instantly with no user interaction.
Can Jabra Integration Service and partner device management systems like PowerSuite and OVOC co-exist with Jabra Direct and Jabra Xpress?
A1: All versions of Jabra Integration Service can co-exist with a Jabra Direct stand-alone installation. (stand-alone means that Jabra Direct was downloaded and installed on the PC from jabra.com i.e. not through Jabra Xpress). *Note:* there is one exception to this rule. It is not possible to use the Jabra PanaCast camera with Jabra Direct when Jabra Integration Service is running.
A2: Jabra Integration Service version 1.5 and older cannot manage devices on the PC if Jabra Direct was deployed using Jabra Xpress.
A3: Jabra Integration Service version 1.6 and newer can manage devices even if Jabra Direct was deployed to the PC using Jabra Xpress, but only as long as Jabra Xpress does not manage devices on the PC. This use case is relevant when a customer wants to mass deploy Jabra Direct but manages device firmware and settings using a partner device management system such as PowerSuite or OVOC.

14 Supported devices

For a list of supported devices please refer to the document *supported_devices.txt*, that is a part of the Jabra Integration Service release package.