PHILIPS

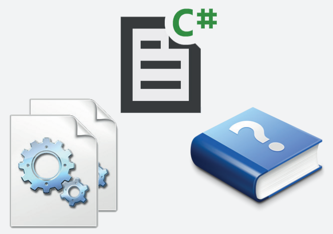
SpeechAir SDK Overview

This SDK is used to integrate Philips SpeechAir devices with 3rd party applications. This section briefly describes the steps from obtaining the SDK to redistributing it.

How to obtain the SDK → SDK Contents → Overall structure of the SDK → SDK Software Architecture → Redistribution

What the SDK includes

- SDK Binaries
- C# Test application including source code
- SDK reference manual



How to redistribute the SDK

The .NET SDK supports side-by-side installation only, no merge modules or setups are provided similar to other side-by-side SDK installations.

The integrator should copy the provided files to the integration´s installation folder besides the executable:

- PSP.SpeechPad.SDK.dll
- PSP.SpeechPad.Device32.dll
- PSP.SpeechPad.Device64.dll
- PSP.SpeechPad.Device32.ini
- PSP.SpeechPad.Device64.ini
- NLog.dll

The integrator must take care of the .NET Framework installation, one of the following .NET Framework versions must be installed:

- NET Framework 4.0 Client Profile
- NET Framework 4.0
- NET Framework 4.5.x
- NET Framework 4.6

The .NET SDK does not use strong names, so it can be replaced in an existing installation.

Visit the site below to apply for SpeechAir SDK. After accepting the SDK agreement, you will receive login information for our partner portal, where you can download the SDK including manuals, and send detailed inquiries as well.

http://www.dictation.philips.com/products-solutions/product/sdk_for_dictation_hardware_lfh7475/

Supported devices

SpeechAir device

Supported Operating Systems

- Windows 7 SP1 x86/x64
- Windows 8.1 x86/x64
- Windows 10 x64
- .NET Framework version 4.0 Client Profile, 4.0, 4.5.x or 4.6

Development Environment

Microsoft Visual Studio 2010/2012/2013/2015

SDK Folder structure

- **Bin**
  Contains the redistributable binary files
- **Doc**
  Contains the SDK reference manual
- **TestApplication**
  Contains the executable test application
  - **Source**
    Contains the source code of the test application

SDK Architecture

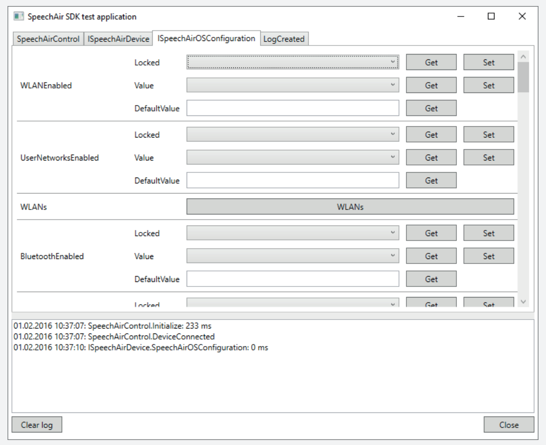
- The SpeechAir SDK is a .NET API which provides a range of interfaces.
- The .NET SDK is built as an AnyCPU assembly. This way a single .NET SDK is provided with a single .NET test application and it can be used in any type of .NET applications (x86, x64 and AnyCPU).
- It can be used to integrate communication directly between a single SpeechAir device and the device application.
- With the SpeechAir SDK, you can change the configuration of a SpeechAir device and the Philips dictation recorder app (PDR) that runs on the device.
- 3rd party applications should create a single instance of the SDK´s main interface (SpeechAirControl) on its main thread

Simple steps of development (.NET application)

1. Create new project (e.g. using Visual Studio) → 2. Copy binaries to your project folder → 3. Create a reference to the main binary (PSP.SpeechAir.SDK.dll) → 4. Port code from the test application

Test Application

The SDK includes a C# test application, which has all SDK functionality embedded. The test application can be used as reference for other applications using the SDK.



```
01.02.2016 10:37:07: SpeechAirControl.Initialize: 233 ms
01.02.2016 10:37:07: SpeechAirControl.DeviceConnected
01.02.2016 10:37:10: ISpeechAirDevice.SpeechAirOSConfiguration: 0 ms
```

API description

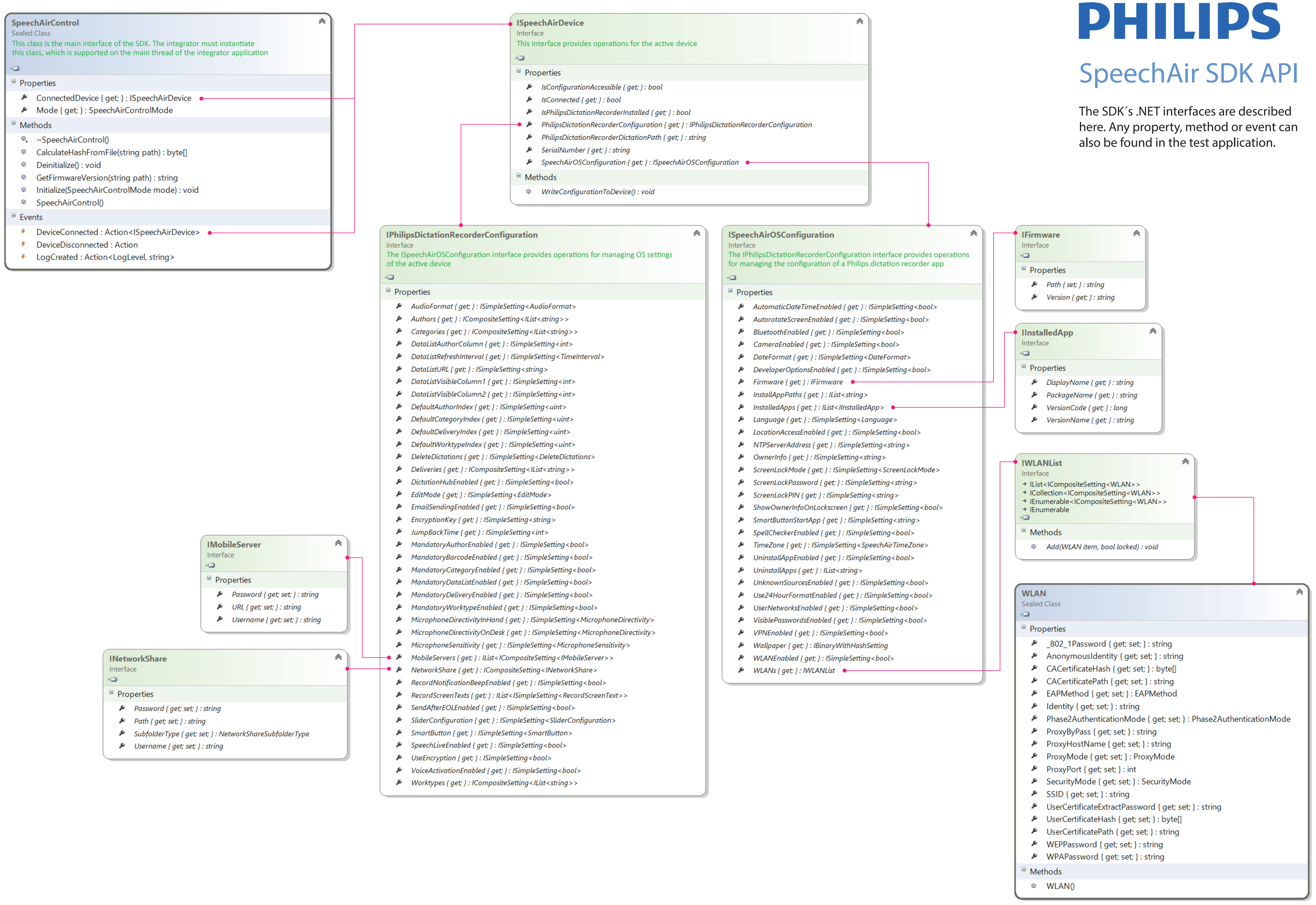
Detailed desprictions of the API´s interfaces, methods and properties can be found in the SDK reference manual located in \Doc\SpeechAirSDK.CHM

Refer to the next page for detailed API descriptions!

PHILIPS

SpeechAir SDK API

The SDK´s .NET interfaces are described here. Any property, method or event can also be found in the test application.



```
SpeechAirControl
Sealed Class
This class is the main interface of the SDK. The integrator must instantiate
this class, which is supported on the main thread of the integrator application

Properties
  ConnectedDevice { get; } : ISpeechAirDevice
  Mode { get; } : SpeechAirControlMode
Methods
  ~SpeechAirControl()
  CalculateHashFromFile(string path) : byte[]
  Deinitialize() : void
  GetFirmwareVersion(string path) : string
  Initialize(SpeechAirControlMode mode) : void
  SpeechAirControl()
Events
  DeviceConnected : Action<ISpeechAirDevice>
  DeviceDisconnected : Action
  LogCreated : Action<LogLevel, string>
```

```
ISpeechAirDevice
Interface
This interface provides operations for the active device

Properties
  IsConfigurationAccessible { get; } : bool
  IsConnected { get; } : bool
  IsPhilipsDictationRecorderInstalled { get; } : bool
  PhilipsDictationRecorderConfiguration { get; } : IPhilipsDictationRecorderConfiguration
  PhilipsDictationRecorderDictationPath { get; } : string
  SerialNumber { get; } : string
  SpeechAirOSConfiguration { get; } : ISpeechAirOSConfiguration
Methods
  WriteConfigurationToDevice() : void
```

```
IPhilipsDictationRecorderConfiguration
Interface
The ISpeechAirOSConfiguration interface provides operations for managing OS settings
of the active device

Properties
  AudioFormat { get; } : ISimpleSetting<AudioFormat>
  Authors { get; } : ICompositeSetting<IList<string>>
  Categories { get; } : ICompositeSetting<IList<string>>
  DataListAuthorColumn { get; } : ISimpleSetting<int>
  DataListRefreshInterval { get; } : ISimpleSetting<TimeInterval>
  DataListURL { get; } : ISimpleSetting<string>
  DataListVisibleColumn1 { get; } : ISimpleSetting<int>
  DataListVisibleColumn2 { get; } : ISimpleSetting<int>
  DefaultAuthorIndex { get; } : ISimpleSetting<uint>
  DefaultCategoryIndex { get; } : ISimpleSetting<uint>
  DefaultDeliveryIndex { get; } : ISimpleSetting<uint>
  DefaultWorktypeIndex { get; } : ISimpleSetting<uint>
  DeleteDictations { get; } : ISimpleSetting<DeleteDictations>
  Deliveries { get; } : ICompositeSetting<IList<string>>
  DictationHubEnabled { get; } : ISimpleSetting<bool>
  EditMode { get; } : ISimpleSetting<EditMode>
  EmailSendingEnabled { get; } : ISimpleSetting<bool>
  EncryptionKey { get; } : ISimpleSetting<string>
  JumpBackTime { get; } : ISimpleSetting<int>
  MandatoryAuthorEnabled { get; } : ISimpleSetting<bool>
  MandatoryBarcodeEnabled { get; } : ISimpleSetting<bool>
  MandatoryCategoryEnabled { get; } : ISimpleSetting<bool>
  MandatoryDataListEnabled { get; } : ISimpleSetting<bool>
  MandatoryDeliveryEnabled { get; } : ISimpleSetting<bool>
  MandatoryWorktypeEnabled { get; } : ISimpleSetting<bool>
  MicrophoneDirectivityInHand { get; } : ISimpleSetting<MicrophoneDirectivity>
  MicrophoneDirectivityOnDesk { get; } : ISimpleSetting<MicrophoneDirectivity>
  MicrophoneSensitivity { get; } : ISimpleSetting<MicrophoneSensitivity>
  MobileServers { get; } : IList<ICompositeSetting<IMobileServer>>
  NetworkShare { get; } : ICompositeSetting<INetworkShare>
  RecordNotificationBeepEnabled { get; } : ISimpleSetting<bool>
  RecordScreenTexts { get; } : IList<ISimpleSetting<RecordScreenText>>
  SendAfterEOLEnabled { get; } : ISimpleSetting<bool>
  SliderConfiguration { get; } : ISimpleSetting<SliderConfiguration>
  SmartButton { get; } : ISimpleSetting<SmartButton>
  SpeechLiveEnabled { get; } : ISimpleSetting<bool>
  UseEncryption { get; } : ISimpleSetting<bool>
  VoiceActivationEnabled { get; } : ISimpleSetting<bool>
  Worktypes { get; } : ICompositeSetting<IList<string>>
```

```
ISpeechAirOSConfiguration
Interface
The IPhilipsDictationRecorderConfiguration interface provides operations
for managing the configuration of a Philips dictation recorder app

Properties
  AutomaticDateTimeEnabled { get; } : ISimpleSetting<bool>
  AutorotateScreenEnabled { get; } : ISimpleSetting<bool>
  BluetoothEnabled { get; } : ISimpleSetting<bool>
  CameraEnabled { get; } : ISimpleSetting<bool>
  DateFormat { get; } : ISimpleSetting<DateFormat>
  DeveloperOptionsEnabled { get; } : ISimpleSetting<bool>
  Firmware { get; } : IFirmware
  InstallAppPaths { get; } : IList<string>
  InstalledApps { get; } : IList<IInstalledApp>
  Language { get; } : ISimpleSetting<Language>
  LocationAccessEnabled { get; } : ISimpleSetting<bool>
  NTPServerAddress { get; } : ISimpleSetting<string>
  OwnerInfo { get; } : ISimpleSetting<string>
  ScreenLockMode { get; } : ISimpleSetting<ScreenLockMode>
  ScreenLockPassword { get; } : ISimpleSetting<string>
  ScreenLockPIN { get; } : ISimpleSetting<string>
  ShowOwnerInfoOnLockscreen { get; } : ISimpleSetting<bool>
  SmartButtonStartApp { get; } : ISimpleSetting<string>
  SpellCheckerEnabled { get; } : ISimpleSetting<bool>
  TimeZone { get; } : ISimpleSetting<SpeechAirTimeZone>
  UninstallAppEnabled { get; } : ISimpleSetting<bool>
  UninstallApps { get; } : IList<string>
  UnknownSourcesEnabled { get; } : ISimpleSetting<bool>
  Use24HourFormatEnabled { get; } : ISimpleSetting<bool>
  UserNetworksEnabled { get; } : ISimpleSetting<bool>
  VisiblePasswordsEnabled { get; } : ISimpleSetting<bool>
  VPNEnabled { get; } : ISimpleSetting<bool>
  Wallpaper { get; } : IBinaryWithHashSetting
  WLANEnabled { get; } : ISimpleSetting<bool>
  WLANs { get; } : IWLANList
```

```
IFirmware
Interface

Properties
  Path { set; } : string
  Version { get; } : string
```

```
IInstalledApp
Interface

Properties
  DisplayName { get; } : string
  PackageName { get; } : string
  VersionCode { get; } : long
  VersionName { get; } : string
```

```
IWLANList
Interface
→ IList<ICompositeSetting<WLAN>>
→ ICollection<ICompositeSetting<WLAN>>
→ IEnumerable<ICompositeSetting<WLAN>>
→ IEnumerable

Methods
  Add(WLAN item, bool locked) : void
```

```
WLAN
Sealed Class

Properties
  _802_1Password { get; set; } : string
  AnonymousIdentity { get; set; } : string
  CACertificateHash { get; set; } : byte[]
  CACertificatePath { get; set; } : string
  EAPMethod { get; set; } : EAPMethod
  Identity { get; set; } : string
  Phase2AuthenticationMode { get; set; } : Phase2AuthenticationMode
  ProxyByPass { get; set; } : string
  ProxyHostName { get; set; } : string
  ProxyMode { get; set; } : ProxyMode
  ProxyPort { get; set; } : int
  SecurityMode { get; set; } : SecurityMode
  SSID { get; set; } : string
  UserCertificateExtractPassword { get; set; } : string
  UserCertificateHash { get; set; } : byte[]
  UserCertificatePath { get; set; } : string
  WEPPassword { get; set; } : string
  WPAPassword { get; set; } : string
Methods
  WLAN()
```

```
IMobileServer
Interface

Properties
  Password { get; set; } : string
  URL { get; set; } : string
  Username { get; set; } : string
```

```
INetworkShare
Interface

Properties
  Password { get; set; } : string
  Path { get; set; } : string
  SubfolderType { get; set; } : NetworkShareSubfolderType
  Username { get; set; } : string
```