# Cortana 白皮书

## Cortana概述

Cortana 是一款个人智能助理，它能够了解用户的喜好和习惯，帮助用户进行日程安排、问题解答等。Cortana 通过您的设备、个人 Microsoft 帐户、第三方服务和其他 Microsoft 服务的数据，了解您以及您的活动后，可以更好地为您服务。为了启用 Cortana 提供个性化体验和相关建议，Microsoft 将收集并使用各种类型的数据，如您的设备位置、您日历中的数据、您使用的应用程序、您电子邮件和短信中的数据、您的通话信息、您的联系人以及您通过设备与其交流的人员。Cortana 还会通过收集您使用设备和其他 Microsoft 服务的相关数据来了解您，比如您喜欢的音乐、闹钟设置、锁屏是否启用、您浏览和购买的物品、您的浏览和 Bing 搜索历史记录，等等。您可以在 Cortana 设置和笔记本中管理 Cortana 使用的数据，以及其对您的了解信息。

**定位服务。**Cortana 定期收集和使用您的当前位置和位置历史记录，以及其他位置信号（例如在您上传到 OneDrive 的照片中标记的位置）。该位置数据用来向您提供与您关系最密切的内容和结果，并提供可以为您节省时间的建议，如交通和移动定位提醒。Cortana 仅可在位置服务启动时运行，如果关闭位置服务，Cortana 将被禁用。

**短信和电子邮件。**Cortana 会访问您的消息以执行各种活动，如：允许您将事件添加到您的日历，通知您重要消息，让您及时跟进与您相关的重要事件或其他内容，如包裹或航班跟踪。Cortana 也使用您的消息帮助您进行事件规划，并为您提供其他有用的建议和推荐。

**通信历史记录。**Cortana 知道您的通话、短信以及电子邮件历史记录中对您最重要的人员。这用来跟踪与您关系最密切的人员以及您最喜欢的通信方式，标记对您重要的消息，并改善其他 Cortana 服务，如语音识别。

**语音和输入个性化。**为了帮助 Cortana 更好地理解您说话的方式和您的语音指令，语音数据将被发送至 Microsoft ，为您建立个性化语音模型并提高语音识别能力和对用户意图的理解。在 Windows 设备上，Cortana 仅可在"输入个性化"启动时运行，如果将其关闭，Cortana 将被禁用。

**Microsoft 应用程序和服务。**Cortana 使用其他 Microsoft 服务收集的数据，提供个性化建议。例如，Cortana 使用 MSN Sports 应用程序收集的数据，从而自动显示您所关注运动队的信息。其也通过 Microsoft 的地图应用程序了解您喜爱的地点，从而提供更好的建议。您对 Cortana 笔记本的兴趣可能被运用到其他 Microsoft 服务，如 Bing 或 MSN，以在那些体验中自定义您的兴趣、偏好和首选。

**第三方服务。**Cortana 也允许您连接至第三方服务，然后根据来自第三方服务的数据，实现其他的个性化体验。例如，选择在 Cortana 内登录 Facebook 或 LinkedIn 将允许 Microsoft 访问某些 Facebook 或 LinkedIn 数据，以便 Cortana 和 Bing 为您提供更多个性化推荐。您可以在 Cortana 笔记本中管理 Cortana 与第三方服务的连接。

**浏览历史。**如果您选择将 Microsoft Edge 内的全部浏览历史记录发送至 Microsoft，Cortana 可根据您在 Microsoft Edge 内访问的网站提供建议。Cortana 不会收集您访问的 InPrivate 标签内的站点信息。

**搜索历史记录。**您的 Bing 搜索查询 - 即使是 Cortana 为您执行的搜索 - 也将与其他 Bing 搜索查询一样处理，并按照 [Bing](https://www.microsoft.com/zh-cn/privacystatement/default.aspx?Componentid=pspMainBingModule&View=Description) 部分中所述的方式使用。

## Win 10 中的Cortana

在Windows Phone 8.1发布的时候，Cortana 正式问世，直到Win10 SDK,在语音命令功能上有了大幅度的提升。其中，最大的进步就是从新定义了命令配置文件，支持更为自然的人工语言形式。并且，支持Win10 SDK的新特性，App Service。通过调用App Service就可以在APP 没有前台运行的时候为Cortana提供数据交互。这样一来Cortana语音命令不仅支持与前台APP的交互，还可以同APP的后台服务进行交互。这次升级，Win10 SDK 为开发者提供更加智能的语音命令API，使开发者可以更为轻松的集成语音命令。接下来我们会为您展示Win 10 中的Cortana，您将学习到如下内容：

* 语音交互应用设计指南
* Cortana在应用中的集成
* Cortana中的语音命令
* Cortana应用示例（奇幻旅行）

## 语音交互应用设计指南

### 语音交互指南

Cortana支持将语音识别和文本到语音转换（又称 TTS 或语音合成）直接集成到你的应用中为用户提供更好的体验，语音可以成为用户与应用交互的可靠而愉快的方式，补充甚至替代键盘、鼠标、触摸和手势。

**语音识别:**对于文本听写，将用户说出的话语转换为用于表单输入的文本，从而指定操作或命令并完成任务。既支持用于自由文本听写和 Web 搜索的预定义语法，也支持使用语音识别语法规范 (SRGS) 版本 1.0 创作的自定义语法。

**文本到语音转换 (TTS):**使用语音合成引擎（语言）可将文本字符串转换为语音字词。输入字符串既可以是基本的未经处理的文本，也可以是更复杂的语音合成标记语言 (SSML)。 SSML 提供用于控制语音输出特征（例如发音、音量、音调、音速和重读）的标准方式。

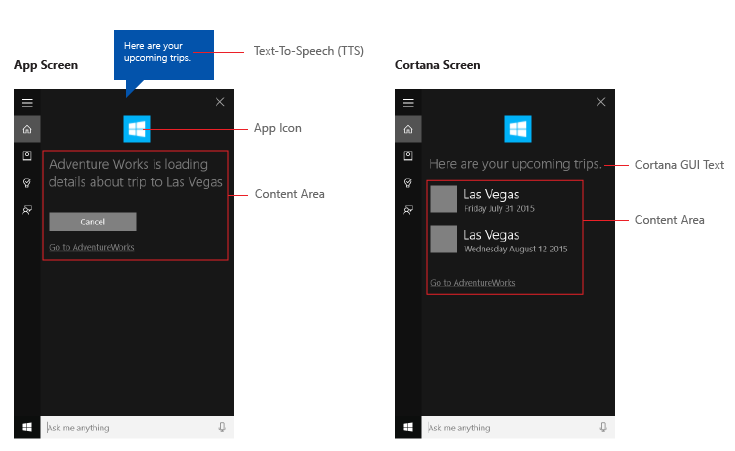
更多内容详见[语音设计指南](https://msdn.microsoft.com/zh-cn/library/windows/apps/xaml/dn596121.aspx)。

### Cortana应用设计指南

使用语音命令，将你的应用提供的功能加入到 **Cortana** 中以对其进行扩展。启动应用并且执行命令，或像在此处讨论的那样，将你的应用的功能直接并入到 **Cortana** UI 中。语音命令是在语音命令定义 (VCD) 文件中定义的具有特定意图的单个发音，通过 **Cortana** 指向一个已安装的应用。 （VCD 文件是定义一个或多个语音命令的 XML 文件，每个文件都带有特定的意图。）VCD 中的每个语音命令定义的复杂程度都不同，并且可以支持任何语音命令，从单个受限发音到一组更灵活的自然语言发音，所有发音都表示相同的特定意图。目标应用可以在前台启动（应用获取焦点）或在后台激活（**Cortana** 保留焦点，但提供来自应用的结果），具体取决于交互的复杂程度。例如，需要额外上下文或用户输入（例如将消息发送给特定联系人）的语音命令最好通过前台应用处理，而基本命令可以通过后台应用使用 **Cortana** 处理。通过集成应用的基本功能，并为用户提供中心入口点以便在无需直接打开应用的情况下完成大多数任务，**Cortana** 成为了应用和用户之间的联络人。在大多数情况下，这可以为用户节省大量时间和精力。

#### Cortana交互设计

这些指南和建议描述了你的应用可以如何充分利用 **Cortana**与用户交互、帮助他们完成任务，以及清楚地表明这一切是如何发生的。**Cortana**允许在后台运行的应用程序提示用户进行确认或消除歧义，转而为用户提供语音命令状态的反馈。该过程轻型、快速，不强制用户退出**Cortana**或将上下文切换到应用程序。虽然用户应该感觉到**Cortana**有助于使该过程尽可能轻而易举，但是你可能希望通过**Cortana**使用您的应用程序来完成任务。我们来看一个示例程序（Adventure Works 是一个关于旅行规划的应用），来展示关于Cortana相关的概念和功能。



在上图中我们可以看到Cortana提供的面板对话信息，这些对话用于Cortana面板与应用程序之间信息的传递。成功的**Cortana**交互要求你在创建文本到语音转换 (TTS) 和 GUI 字符串时，遵循一些基本原则。

* 高效：使用尽可能少的字词，并将最重要的信息放在最前面。如：当然，你在寻找哪部电影？
* 相关：提供仅与任务、内容和上下文相关的信息。如：我已将它添加到你的播放列表。
* 清楚：避免混淆。使用日常生活语言，而不是技术行话。如：我找不到任何到拉斯维加斯的旅行。
* 可信：尽量准确。 对在后台运行的任务了如指掌—如果某个任务尚未完成，则不说它已完成。尊重隐私—不朗读私人信息。如：我无法在我们的目录中找到该电影。

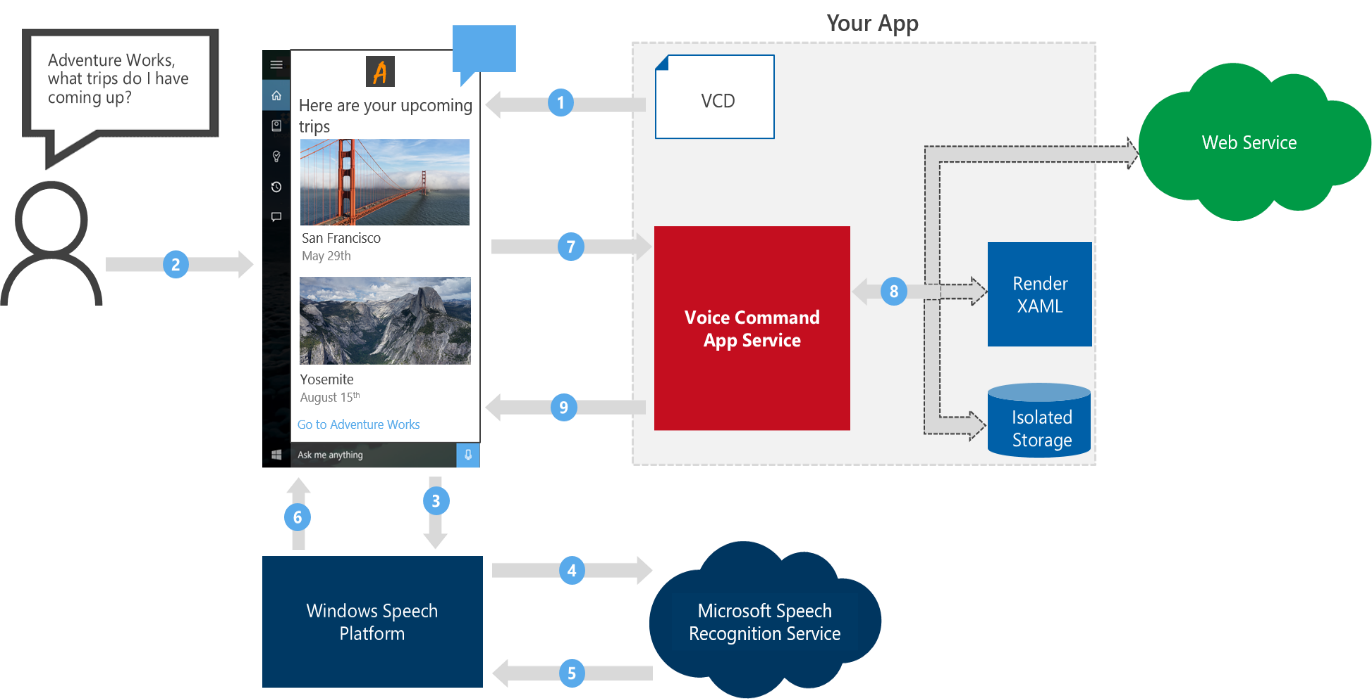
尽量使用更为自然的交互对话，不要为了语法的准确性编写较为生硬的对话，尽可能的使用隐式的第一人称时态。例如，“寻找你的下一个 Adventure Works 之旅”。尽可能的使用一些语法变化来帮助您的应用听起来更为自然流畅，我们可以提供不同版本的TTS和GUI字符串，有效地说出相同的话。例如"你要看哪部电影？"可以替换为"你希望观看哪部电影？"等。人们每次说的话不可能完全相同。只需确保使 TTS 版本和 GUI 版本保持同步。更多内容请参阅[Cortana设计指南](https://msdn.microsoft.com/zh-cn/library/windows/apps/xaml/dn974233.aspx)。

### 多语言支持

若要使用语音命令启动某项操作，你的应用必须使用，用户在其设备上所选的语言(Settings > System > Speech > Speech Language) 注册语音命令。你应该将你的应用所对应的语音命令和所有 TTS 和 GUI 字符串都本地化。你应该避免较长的 GUI 字符串。**Cortana**画布提供三行以供响应，并将截断长于这三行的字符串。[GUI全球化指南](https://msdn.microsoft.com/zh-cn/library/windows/apps/xaml/dn611859.aspx)

## Cortana在应用中的集成

Win10中的Cortana更加智能，现在的Cortana已经成为一个最好的个人助手。最初在Win8平台的Cortana，用户可以通过系统命令来激活应用程序，但是现在Win10中的Cortana，用户不仅可以通过命令来激活任何应用程序，还可以通过后台的App Service来接收用户的语音指令，并且将应用程序的反馈信息直接显示在Cortana的面板中。

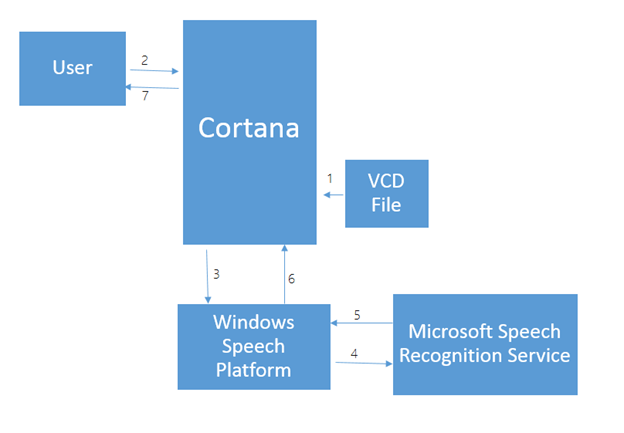


在应用程序启动时，需要在Cortana中安装VCD文件，用户可以在VCD文件中自定义语音命令，Cortana在接到语音命令时，会通过Windows语音平台，调用微软的语音/语法识别服务来完成识别并作出相应的操作。应用程序也可以开启语音命令的后台服务来接收Cortana的语音指令，并将语音指令的执行结果反馈到Cortana面板中去。

### Cortana的前台集成方式

除了在 **Cortana** 内使用语音命令访问系统功能以外，你还可以通过 **Cortana** 使用语音命令启动前台应用并在应用内指定某个要执行的操作或命令。语音命令是在语音命令定义 (VCD) 文件中定义的具有特定意图的单个发音，通过 **Cortana** 指向一个已安装的应用。语音命令定义的复杂程度可能有所不同。它可以支持任何语音命令，从单个受限发音到一组更灵活的自然语言发音，所有发音都表示相同的意图。VCD（Voice Command Definition） 文件定义一个或多个语音命令，每个语音命令都具有一种特殊意图。目标应用可以在前台启动，具体取决于交互的复杂程度。例如，需要额外上下文或用户输入（例如将消息发送给特定联系人）的语音命令最好通过前台应用处理，而基本命令可以通过后台应用使用 **Cortana** 处理。

我们通常在需要快速访问应用程序内部数据的时候，会使用到Cortana前台启动，例如：我想打开奇幻旅行应用，来查看我的所有旅行规划。或是查看去某个地点旅行的信息，我可以通过Cortana说出“奇幻旅行 显示我去伦敦的旅行”。Cortana会帮我们拉起应用并查询伦敦旅行的详细安排。



**Cortana的工作流程如下：**

1. 当APP运行时，需要在Cortana中安装VCD文件。（VCD：语音命令定义。是一个XML文件，其中包含了所有的命令，用来激活应用程序）。
2. 用户可以使用在VCD文件中定义的命令，来启动应用程序。
3. （3、4、5、6）用户通过微软云端语音平台和微软语音识别服务将语音转换成命令文本。
4. Cortana接收通过识别的字符串，来确定启动应用程序，并在应用程序中执行相应的事件。

#### Voice Command Definition （VCD）文件

如下为一个VCD文件的内容：

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <VoiceCommands xmlns="http://schemas.microsoft.com/voicecommands/1.2">  <CommandSet xml:lang="zh-cn" Name="AdventureWorksCommandSet\_zh-cn">  <CommandPrefix> 奇幻旅行 </CommandPrefix>  <Example>显示伦敦之行</Example>  <Command Name="showTripToDestination">  <Example>显示伦敦之行</Example>  <ListenFor>显示[我]去{destination}的旅行</ListenFor>  <ListenFor RequireAppName="ExplicitlySpecified">显示[我]{builtin:AppName}去{destination}的旅行</ListenFor>  <Feedback>显示{destination}之行</Feedback>  <Navigate Target="foo"/>  </Command>  <PhraseList Label="destination">  <Item>伦敦</Item>  <Item>拉斯维加斯</Item>  <Item>墨尔本</Item>  </PhraseList>  </CommandSet>  </VoiceCommands> |

#### 安装VCD语音命令文件

在App.xaml.cs文件中找到OnLaunched方法，在该方法的最后，添加安装VCD文件的代码，同时还要修改OnLaunched方法为async。安装VCD文件完整代码如下：

|  |
| --- |
| try  {  StorageFile vcdStorageFile = await Package.Current.InstalledLocation.GetFileAsync(@"AdventureWorksCommands.xml");  await Windows.ApplicationModel.VoiceCommands.VoiceCommandDefinitionManager.InstallCommandDefinitionsFromStorageFileAsync(vcdStorageFile);  }  catch (Exception ex)  {  System.Diagnostics.Debug.WriteLine("Installing Voice Commands Failed: " + ex.ToString());  } |

#### 响应Cortana的调用

我们找到protected override void OnActivated(IActivatedEventArgs args)方法，添加如下代码：

|  |
| --- |
| protected override void OnActivated(IActivatedEventArgs args)  {  base.OnActivated(args);  Type navigationToPageType;  ViewModel.TripVoiceCommand? navigationCommand = null;  if (args.Kind == ActivationKind.VoiceCommand)  {  var commandArgs = args as VoiceCommandActivatedEventArgs;  Windows.Media.SpeechRecognition.SpeechRecognitionResult speechRecognitionResult = commandArgs.Result;  string voiceCommandName = speechRecognitionResult.RulePath[0];  string textSpoken = speechRecognitionResult.Text;  string commandMode = speechRecognitionResult.SemanticInterpretation.Properties["commandMode"].FirstOrDefault();  if(voiceCommandName == "showTripToDestination")  {  string destination =  speechRecognitionResult.SemanticInterpretation.Properties["destination "].FirstOrDefault();  navigationCommand = new ViewModel.TripVoiceCommand(  voiceCommandName,  commandMode,  textSpoken,  destination);  navigationToPageType = typeof(View.TripDetails);  }  }  else  {  navigationToPageType = typeof(View.TripListView);  }  Frame rootFrame = Window.Current.Content as Frame;  if (rootFrame == null)  {  rootFrame = new Frame();  App.NavigationService = new NavigationService(rootFrame);  Window.Current.Content = rootFrame;  }  rootFrame.Navigate(navigationToPageType, navigationCommand);  Window.Current.Activate();  } |

该方法的主要功能是，通过识别Cortana的命令来跳转页面。

**Cortana前台启动会使用到Windows.ApplicationModel.VoiceCommands namespace，相关方法如下**：

1、InstallCommandDefinitionsFromStorageFileAsync(StorageFile)：安装VCD文件。

2、确定你的应用已通过语音命令激活，替换**Application.OnActivated**事件并检查**IActivatedEventArgs.Kind**是否是**VoiceCommand**类型。

3、确定命令名称和说出的内容，从IActivatedEventArgs中获取对VoiceCommandActivatedEventArgs对象的引用并查询SpeechRecognitionResult对象的Result属性。

4、在应用中执行相应的操作，通常是导航到相关页面。

关于Cortana的前台集成方式，详见[在Cortana中使用语音命令启动前台应用](https://msdn.microsoft.com/library/windows/apps/xaml/mt185609.aspx)

### Cortana的后台集成方式

除了在**Cortana**内使用语音命令访问系统功能，你还可以使用用于指定在应用内执行操作或命令的语音命令，通过后台应用中的特性和功能扩展**Cortana**。 当应用在后台处理语音命令时，它可以在**Cortana**面板上显示反馈并使用**Cortana**语音与用户交流。

Win 10 SDK给应用程序增加了一个叫做APP Service的技术，应用程序可以通过APP Service公开服务来让其他应用程序调用。APP Service是通过后台任务来处理的，所以不需要我们启动应用程序，调用者只需要知道提供服务的应用程序的程序包名称，以及要调用的服务名称即可进行调用。

以下是使用语音或键盘输入添加语音命令功能并通过应用中的后台功能扩展**Cortana**的基本步骤：

1. 创建Cortana在后台调用的应用服务。（Windows.ApplicationModel.AppService）
2. 创建VCD文件。这是一个XML文档，可以定义在激活应用时用户可说出以启动操作或调用命令的所有语音命令。
3. 当启动应用时，在VCD文件中注册命令集。
4. 处理应用服务的后台激活和语音命令的执行。
5. 在Cortana内显示并说出对应的语音命令的相应反馈。

#### 创建App Service

在 Microsoft Visual Studio 中，右键单击“解决方案名称”，选择 Add->New Project，然后选择“Windows 运行时组件”。键入项目的名称（例如，"VoiceCommandService"），然后单击“确定”。选择 "VoiceCommandService" 项目，并为 Visual Studio 生成的 "Class1.cs" 文件重命名为 "VoiceCommandService.cs"。在 "VoiceCommandService.cs" 中，创建一个实现**IBackgroundTask**接口的新类。**Run**方法是所需的入口点，在**Cortana**识别语音命令后调用。代码如下：

|  |
| --- |
| u using Windows.ApplicationModel.Background;  namespace AdventureWorks.VoiceCommands  {  public sealed class AdventureWorksVoiceCommandService : IBackgroundTask  {  public void Run(IBackgroundTaskInstance taskInstance)  {  BackgroundTaskDeferral \_deferral = taskInstance.GetDeferral();  //  // TODO: Insert code  //  //  \_deferral.Complete();  }  }  } |

值得注意的是我们在工程中添加App Service除了需要在主项目添加App service工程的引用外还需要配置Package.appxmanifest文件，我们需要右键单击“Packae.appxmanifest”文件并选择“查看代码”，添加App Service配置描述文件，如下所示：

|  |
| --- |
| <Package>  <Applications>  <Application>  <Extensions>  <uap:Extension Category="windows.appService" EntryPoint="AdventureWorks.VoiceCommands.AdventureWorksVoiceCommandService">  <uap:AppService Name="AdventureWorksVoiceCommandService" />  </uap:Extension>  </Extensions>  <Application>  <Applications>  </Package> |

在Extension元素中，有两个重要的属性，Category和EntryPoint。

**Category 属性**：定义扩展的类型，此处为一个APP Service.

**EntryPoint 属性**：后台应用服务的一个入口命名空间，当Cortana调用后台服务时，会运行该类的Run方法。

**AppService 元素**：定义了App Service的名称。

#### Voice Command Definition （VCD）文件

Cortana后台启动VCD文件定义如下：

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <VoiceCommands xmlns="http://schemas.microsoft.com/voicecommands/1.2">  <CommandSet xml:lang="zh-cn" Name="AdventureWorksCommandSet\_zh-cn">  <CommandPrefix> 奇幻旅行 </CommandPrefix>  <Command Name="whenIsTripToDestination">  <Example>我的拉斯维加斯之旅的时间？</Example>  <ListenFor>[我]前往{destination}的旅行是什么时候</ListenFor>  <ListenFor RequireAppName="ExplicitlySpecified">显示[我]{builtin:AppName}前往{destination}的旅行是什么时候</ListenFor>  <Feedback>查询{destination}之行</Feedback>  <VoiceCommandService Target="AdventureWorksVoiceCommandService"/>  </Command>  <Command Name="cancelTripToDestination">  <Example>取消我的拉斯维加斯之旅</Example>  <ListenFor>取消[我的]{destination}之旅</ListenFor>  <ListenFor RequireAppName="ExplicitlySpecified">取消[我的]{builtin:AppName}{destination}之旅</ListenFor>  <Feedback>取消{destination}之行</Feedback>  <VoiceCommandService Target="AdventureWorksVoiceCommandService"/>  </Command>  <PhraseList Label="destination">  <Item>伦敦</Item>  <Item>拉斯维加斯</Item>  <Item>墨尔本</Item>  </PhraseList>  </CommandSet>  </VoiceCommands> |

#### 安装VCD语音命令文件

在App.xaml.cs文件中找到OnLaunched方法，在该方法的最后，添加安装VCD文件的代码，同时还要修改OnLaunched方法为async。安装VCD文件完整代码如下：

|  |
| --- |
| try  {  StorageFile vcdStorageFile = await Package.Current.InstalledLocation.GetFileAsync(@"AdventureWorksCommands.xml");  await Windows.ApplicationModel.VoiceCommands.VoiceCommandDefinitionManager.InstallCommandDefinitionsFromStorageFileAsync(vcdStorageFile);  }  catch (Exception ex)  {  System.Diagnostics.Debug.WriteLine("Installing Voice Commands Failed: " + ex.ToString());  } |

#### 使用Cortana与后台应用交互

Cortana与后台应用交互，代码如下所示：

|  |
| --- |
| public sealed class VoiceCommandService : IBackgroundTask  {  private BackgroundTaskDeferral serviceDeferral;  VoiceCommandServiceConnection voiceServiceConnection;  public async void Run(IBackgroundTaskInstance taskInstance)  {  this.serviceDeferral = taskInstance.GetDeferral();  var triggerDetails =  taskInstance.TriggerDetails as AppServiceTriggerDetails;  if (triggerDetails != null &&  triggerDetails.Name == "AdventureWorksVoiceServiceEndpoint")  {  try  {  voiceServiceConnection =  VoiceCommandServiceConnection.FromAppServiceTriggerDetails(  triggerDetails);  voiceServiceConnection.VoiceCommandCompleted +=  VoiceCommandCompleted;  VoiceCommand voiceCommand = await  voiceServiceConnection.GetVoiceCommandAsync();  switch (voiceCommand.CommandName)  {  case "whenIsTripToDestination":  {  var destination =  voiceCommand.Properties["destination"][0];  SendCompletionMessageForDestination(destination);  break;  }  default:  LaunchAppInForeground();  break;  }  }  finally  {  if (this.serviceDeferral != null)  {  this.serviceDeferral.Complete();  }  }  }  }  private void VoiceCommandCompleted(VoiceCommandServiceConnection sender,  VoiceCommandCompletedEventArgs args)  {  if (this.serviceDeferral != null)  {  this.serviceDeferral.Complete();  }  }  private async void SendCompletionMessageForDestination(  string destination)  {  var userMessage = new VoiceCommandUserMessage();  userMessage.DisplayMessage = "Here’s your trip.";  userMessage.SpokenMessage = "Your trip to Vegas is on August 3rd.";  var destinationsContentTiles = new List<VoiceCommandContentTile>();  var destinationTile = new VoiceCommandContentTile();  destinationTile.ContentTileType =  VoiceCommandContentTileType.TitleWith68x68IconAndText;  destinationTile.AppLaunchArgument =  string.Format("destination={0}”, “Las Vegas");  destinationTile.Title = "Las Vegas";  destinationTile.TextLine1 = "August 3rd 2015";  destinationsContentTiles.Add(destinationTile);  var response =  VoiceCommandResponse.CreateResponse(  userMessage, destinationsContentTiles);  response.AppLaunchArgument =  string.Format("destination={0}”, “Las Vegas");  await voiceServiceConnection.ReportSuccessAsync(response);  }  private async void LaunchAppInForeground()  {  var userMessage = new VoiceCommandUserMessage();  userMessage.SpokenMessage = "Launching Adventure Works";  var response = VoiceCommandResponse.CreateResponse(userMessage);  response.AppLaunchArgument = "showAllTrips=true";  await voiceServiceConnection.RequestAppLaunchAsync(response);  }  } |

后台任务的功能主要是响应Cortana收到的语音命令，应用程序通过后台App Service来触发，这样我们就能在后台任务中进行交互。Cortana后台启动会使用到相关方法如下：

**VoiceCommandServiceConnection类**：它的实例可以从后台任务实例的触发器数据中获得。它使用“FromAppServiceTriggerDetails”方法来获取实例。该类有几个重要的方法如下：

* **ReportSuccessAsync(VoiceCommandResponse response)**：通知Cortana，处理已经完成，并返回一条消息，传递给Cortana的面板显示。（注意：一旦Cortana启动后台服务，应用服务有0.5秒的时间调用ReportSuccessAsync方法。Cortana使用应用提供的数据显示并说出VCD文件中指定的反馈。如果应用进行该调用的时间超过0.5秒，Cortana将显示切换屏幕，直到应用程序调用ReportSuccessAsync方法为止，最多显示5秒。如果应用服务不调用 **ReportSuccessAsync**或任何提供为 **Cortana** 提供信息的 **VoiceCommandServiceConnection** 方法，则用户将收到一条错误消息并且该应用服务将会取消。）
* **ReportFailureAsync(VoiceCommandResponse response)**：该方法用于向Cortana反馈错误信息。
* **ReportProgressAsync(VoiceCommandResponse response)**：该方法用于向Cortana报告进度，但并不指定具体的进度值，只是在Cortana面板上会显示时长5秒钟的进度条。
* **RequestAppLaunchAsync(VoiceCommandResponse response)**：该方法用于向Cortana请求启动当前应用。
* **RequestConfirmationAsync(VoiceCommandResponse response)**：该方法用于向Cortana面板发送一条需要用户确认的消息，后台服务会等待用户的确认结果，以决定下一步做什么操作。
* **RequestDisambiguationAsync(VoiceCommandResponse response)**：该方法用于向Cortana面板发送一条需要用户确认的消息。与上面方法不同的是，这个方法会在Cortana面板上列出多个选项，让用户说出或点击要选择的选项，Cortana会把用户选择的项返回给应用服务，以决定下一步做什么操作。（注意：作为方法参数的VoiceCommandResponse实例一定要使用CreateResponseForPrompt静态方法来创建）

**VoiceCommandUserMessage 类：**该类主要用于Cortana在与后台服务的交互过程中，向Cortana面板发送消息的封装。它有两个重要的属性：

**DisplayMessage:**要显示在Cortana面板上的文本信息。

**SpokenMessage:**希望Cortana说出来的文本。

## Cortana中的语音命令

### Voice Command Definition（VCD）文件元素、属性定义

VCD文件主要用来定义Cortana识别的命令，如下示例：

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <VoiceCommands xmlns="http://schemas.microsoft.com/voicecommands/1.2">  <CommandSet xml:lang="zh-cn" Name="AdventureWorksCommandSet\_zh-cn">  <CommandPrefix> 奇幻旅行 </CommandPrefix>  <Example>显示伦敦之行</Example>  <Command Name="showTripToDestination">  <Example>显示伦敦之行</Example>  <ListenFor>显示[我]去{destination}的旅行</ListenFor>  <ListenFor RequireAppName="ExplicitlySpecified">显示[我]{builtin:AppName}去{destination}的旅行</ListenFor>  <Feedback>显示{destination}之行</Feedback>  <Navigate Target="foo"/>  </Command>  <Command Name="whenIsTripToDestination">  <Example>我的拉斯维加斯之旅的时间？</Example>  <ListenFor>[我]前往{destination}的旅行是什么时候</ListenFor>  <ListenFor RequireAppName="ExplicitlySpecified">显示[我]{builtin:AppName}前往{destination}的旅行是什么时候</ListenFor>  <Feedback>查询{destination}之行</Feedback>  <VoiceCommandService Target="AdventureWorksVoiceCommandService"/>  </Command>  <PhraseList Label="destination">  <Item>伦敦</Item>  <Item>拉斯维加斯</Item>  <Item>墨尔本</Item>  </PhraseList>  </CommandSet>  </VoiceCommands> |

**注意：**在XML文件中，我们定义了一个中文的语音命令集，在语音命令定义文件中，有一个CommandPrefix节点，它的内容为“奇幻旅行”。这个节点是用户在调用语音命令时使用的前缀。调用语音命令时，用户需要告诉Cortana” 奇幻旅行显示去伦敦的旅行”，即可打开应用程序。

**Voice Command Definition（VCD）文件，必须包含如下几个元素**：

* **VoiceCommands 元素：**VCD文件的根节点，xmlns属性必须为“http://schemas.microsoft.com/voicecommands/1.2”，它可以包含1-15个CommandSet节点，每个CommandSet节点代表一个语言的语音命令。
* **CommandSet 元素：**是针对不同语言定义的一组命令。如：xml:lang="zh-cn"，定义了一组中文语音指令。（注意：lang 属性必须是唯一的）
* **CommandPrefix 元素**：是**CommandSet**的子节点，定义了命令的前缀，通常命令组成是以**CommandPrefix**开头**+Command**来实现命令。
* **Command 元素：**是CommandSet的子节点，定义了单独的一个命令。它包含Example(1个)、ListenFor（1到10个）、Feedback（1个）和Navigate（1个）等几个元素，且必须按顺序列出。
* **Example 元素：**告诉用户该App所支持的语音指令的描述。
* **ListenFor 元素：**监听并识别用户的指令，每个Command可以有（1-10个）该元素。（该元素{}中的内容在**PhraseList**节点中可以定义，[]为可以忽略的短语）
* **Feedback 元素：**Cortana识别命令后，给用户的一个反馈。用于Cortana显示。
* **Navigate 元素：**声明语音命令将在前台启动应用。与VoiceCommandService元素互斥，两者只能使用其中一个。
* **VoiceCommandSerice 元素：**声明语音命令将启动后台应用。与Navigate 元素互斥，两者只能使用其中一个。
* **PhraseList 元素：**用来定义一组语音字符，指定相应规定的字符，用来消除歧义, 使用 PhraseList 限制识别适用于一组相对较小的单词。当单词组过大（例如数百个单词）或者根本不应被限制时，要使用 PhraseTopic 元素和 Subject 元素来优化语音识别结果的相关性，从而增强可扩展性。
* **PhraseTopic 元素：**可以提高识别率，内部属性Subject可指定该关键字类型，比如 城市名 姓名 地址 等类型。

### 语音命令的动态修改

如果语音命令特定于涉及一些用户定义的收藏夹或临时应用数据的任务，则在运行时动态修改短语列表可能会非常有用。

例如，假设你有一个旅行应用（用户可以在其中输入目的地），并且你希望用户能够通过在说出"显示旅行 <目的地>" 后紧接着说出该应用名称来启动它。无需为每个可能的目的地创建单独的**ListenFor**元素。相反，你可以使用用户所创建的目的地在运行时动态填充**PhraseList**。在**ListenFor**元素本身中，你可以指定如下内容：<ListenFor> Show trip to {destination} </ListenFor>，其中 "destination" 是**PhraseList**的**Label**属性值。

动态修改语音命令需要如下两个步骤：

1. 若要更新 VCD 文件中的 **PhraseList** 元素，请获取包含短语列表的**CommandSet**元素。将**CommandSet**元素的**Name**属性（**Name**在 VCD 文件中必须独一无二）用作密钥以访问**VoiceCommandManager.InstalledCommandSets**属性并获取**VoiceCommandSet**参考。
2. 标识命令集之后，参考要修改的短语列表并调用 **SetPhraseListAsync** 方法；使用**PhraseList**元素的**Label**属性和一组字符串作为短语列表的新内容。

示例代码如下：

|  |
| --- |
| Windows.ApplicationModel.VoiceCommands.VoiceCommnadDefinition.VoiceCommandSet commandSetEnUs;  if (Windows.ApplicationModel.VoiceCommands.VoiceCommandDefinitionManager.  InstalledCommandSets.TryGetValue(  "AdventureWorksCommandSet\_en-us", out commandSetEnUs))  {  await commandSetEnUs.SetPhraseListAsync(  "destination", new string[] {“London”, “Dallas”, “New York”, “Phoenix”});  } |

更多内容，请参考[动态修改VCD短语列表](https://msdn.microsoft.com/zh-cn/library/windows/apps/xaml/mt185601.aspx)

### 自然语言命令

通过语音命令扩展 **Cortana** 要求用户指示应用和要执行的特定命令或功能。该操作通常通过在语音命令开始时读出应用名称来完成。例如，"Adventure Works，新增一个拉斯维加斯之旅。"而指定应用程序名称可防止命令出现发音生硬、不自然甚至是表达不清楚等情况。在许多情况下，如果你能够在其他位置使用命令说出应用名称，则可以使命令显得更为贴切和自然，并且有助于让与用户的交互变得更为直观和趣味盎然。之前的示例："Adventure Works，新增一个拉斯维加斯之旅。"，可以重新表述为"新增一个拉斯维加斯 Adventure Works 之旅。"，也可以表述为"使用 Adventure Works 新增一个拉斯维加斯之旅。"你可以将语音命令设置为支持应用名称作为下列形式：

* 前缀 - 位于命令短语之前
* 中缀 - 位于命令短语内
* 后缀 - 位于命令短语之后

**指定可在语音命令中说出应用名称的位置**

**ListenFor**元素具有一个**RequireAppName**属性，该属性用于指定可在语音命令中显示应用名称的位置。该属性支持四个值。

1. BeforePhrase：默认值。指示用户必须在命令短语之前说出应用名称。例如：

|  |
| --- |
| <ListenFor RequireAppName="BeforePhrase"> when is [my] trip to {destination} |

1. AfterPhrase: 指示用户必须在命令短语之后说出应用名称。例如“

|  |
| --- |
| <ListenFor RequireAppName="AfterPhrase">show [my] next trip to {destination} </ListenFor> |

1. BeforeOrAfterPhrase: 指示用户必须在命令短语之前或之后说出应用名称。例如：

|  |
| --- |
| <ListenFor RequireAppName="BeforeOrAfterPhrase">show [my] next trip to {destination} </ListenFor> |

1. ExplicitlySpecified: 指示用户必须准确说出应用名称在命令短语中所指定的位置。不要求用户说出应用名称是在短语之前还是之后。例如：

|  |
| --- |
| <ListenFor RequireAppName="ExplicitlySpecified">show [my] next {builtin:AppName} trip to {destination} </ListenFor> |

如果支持用户在 **Cortana** 中说出语音命令时采用更多变化形式，那么通常也会增加你的应用的使用率。更多内容，请参考[在Cortana中支持自然语言语音命令](https://msdn.microsoft.com/zh-cn/library/windows/apps/xaml/mt185616.aspx)。

## 示例代码

这个示例项目是一个名叫AdventureWorks的应用程序，它实现了一个关于旅行计划的管理。通过APP用户可以记录旅行的相关信息，如：什么时间段内，在哪个城市，做了什么等。用户通过Cortana的语音命令可以启动前台应用程序，还可以通过语音命令与应用程序的后台服务进行交互。在应用程序中，可以根据语音命令来显示游记的详细信息。为了提高程序的智能程度，应用程序通过当前设备设置的语言来更新命令短语列表。

**通过示例项目您将学会：**

* Voice Command Definition（VCD）文件元素、属性定义。
* 在Cortana中使用语音命令启动前台应用。
* 在Cortana中使用语音命令启动后台应用。
* 如何在Cortana中与后台应用交互。
* 动态修改VCD文件列表。
* 自然语言形式的语音命令。
* UI的中英文切换。
* 应用程序内部实现命令的监听，使用语音的方式与应用前台界面进行交互，并使用语音的方式输入。

[示例项目下载](NULL)

**相关资料：**

动手实验手册：[Cortana 扩展—语音命令的使用](NULL)

动手实验代码框架：[AdventureWorks\_Framework](NULL)