Windows Store Delphi Component ReadMe

- The app that the component is going to be used with should have a package manifest file AppxManifest.xml. More information about the package manifest can be found here: https://docs.microsoft.com/en-us/uwp/schemas/appxpackage/appx-package-manifest.
- It's advised that the app package should have already been published in the Windows store (it can be hidden from the public) before working with the component.

In order to work with the Windows.Services.Store namespace, the app should be installed at least once from the Windows store. This is needed so that the app's license got installed on the computer, as well.

After that you can delete the app installed from the store (right click on the tile and choose Uninstall) and install the local debuggable copy using the following PowerShell command (note, the command should be executed from the folder where the app and its manifest files are):

Add-AppxPackage -Register AppxManifest.xml

You will see the app's tile created in the Windows start menu.

Windows store functionality will work only if the app is started from its tile. Only then the store app id will be presented in the license.

To work with the namespace, you'd first need to get an interface instance:

```
IStoreContext:
  var
    StoreContext: IStoreContext;
  ...
  StoreContext := TStoreContextStatics.Statics.GetDefault;
```

or

```
  StoreContext := TStoreContextStatics.Statics.GetGetForUser(User);
```

In case of a multiuser app.

For non-UWP apps initialization is required:

```
  (StoreContext as IInitializeWithWindow).Initialize(MainAppWindow.Handle);
```

After that proceed according to the Windows.Services.Store namespace documentation: https://docs.microsoft.com/en-us/uwp/api/windows.services.store

Couple of additional remarks:

Converting HSTRING into Delphi String:

```
  String := HString.ToString;
```

Converting Delphi String into HSTRING:

```
  HString := TWindowsString.Create(String);
```

A string list for some methods parameters (interface IIterable_1__HSTRING):

```
  ParamList :=
TIterable_1__HSTRING.Create('ExampleString1,ExampleString2,ExampleString3');
```

Converting DateTime into Delphi TDateTime:

```
  DelphiDateTime := DateTimeToDelphiDateTime(DateTime);
```

Event handler

```
  Create own class from TTypedEventHandler_2__IInspectable__IInspectable and
override Execute method
```

Waiting for the operations to complete (IAsyncOperation_* interface) (example):

```
    uses
        Winapi.Foundation;
      ...
      var
        ASOP: IAsyncOperation_1__IInspectable;
        SPR: IStorePurchaseResult;
      begin
        ASOP :=
FStoreContext.RequestPurchaseAsync(TWindowsString.Create(StoreId));
        try
         while (ASOP as IAsyncInfo).Status = AsyncStatus.Started do
            Application.ProcessMessages;
         SPR := (ASOP.GetResults as IStorePurchaseResult);
        finally
         (ASOP as IAsyncInfo).Close;
        end;
      end;
```

Enumeration of IVectorView_* (example):

```
    uses
        Winapi.Foundation.Collections;
      ...
      var
        SKUs: IVectorView_1__IInspectable;
        I: Integer;
        SKU: IStoreSku;
      begin
        SKUs := StoreProduct.Skus;
        for I := 0 to SKUs.Size - 1 do
         SKU := (SKUs.GetAt(I) as IStoreSku);
      end;
```

Note, the component uses some WinAPI libraries from Delphi 10.2, which will be included in the package.