Interface definition

```
class PowertoyModuleIface {
public:
  virtual const wchar_t* get_name() = 0;
  virtual const wchar_t** get_events() = 0;
  virtual bool get_config(wchar_t* buffer, int *buffer_size) = 0;
  virtual void set_config(const wchar_t* config) = 0;
  virtual void call_custom_action(const wchar_t* action) {};
  virtual void enable() = 0;
  virtual void disable() = 0;
  virtual bool is_enabled() = 0;
  virtual void destroy() = 0;
};

typedef PowertoyModuleIface* (__cdecl *powertoy_create_func)();
```

Runtime logic

The PowerToys runner will, for each PowerToy DLL: - load the DLL, - call `powertoy_create()` to create the PowerToy.

On the received object, the runner will call: - `get_name()` to get the name of the PowerToy, - `enable()` to initialize the PowerToy.

While running, the runner might call the following methods between create_powertoy() and destroy(): - `disable()`/`enable()`/`is_enabled()` to change or get the PowerToy's enabled state, - `get_config()` to get the available configuration settings, - `set_config()` to set settings after they have been edited in the Settings editor, - `call_custom_action()` when the user selects a custom action in the Settings editor,

When terminating, the runner will: - call `disable()`, - call `destroy()` which should free all the memory and delete the PowerToy object, - unload the DLL.

Method definition

This section contains a more detailed description of each of the interface methods.

powertoy_create_func

```
typedef PowertoyModuleIface* (__cdecl *powertoy_create_func)()
```

Typedef of the factory function that creates the PowerToy object. Must be exported by the DLL as `powertoy_create()`.

Called by the PowerToys runner to initialize each PowerToy. It will be called only once before a call to `destroy()` is made.

The returned PowerToy should be in the disabled state. The runner will call the `enable()` method to start the PowerToy.

In case of errors returns `nullptr`.

get_name

```
virtual const wchar_t* get_name()
```

Returns the name of the PowerToy, it will be cached by the runner.

get_config

```
virtual bool get_config(wchar_t* buffer, int *buffer_size)
```

Fills a buffer with the available configuration settings.

If `buffer` is a null pointer or the buffer size is not large enough sets the required buffer size in 'buffer_size' and return false.

Returns true if successful.

set_config

```
virtual void set_config(const wchar_t* config)
```

After the user has changed the module settings in the Settings editor, the runner calls this method to pass to the module the updated values. It's a good place to save the settings as well.

call_custom_action

```
  virtual void call_custom_action(const wchar_t* action)
```

Calls a custom action in response to the user pressing the custom action button in the Settings editor. This can be used to spawn custom editors defined by the PowerToy.

enable

```
  virtual void enable()
```

Enables the PowerToy.

disable

```
  virtual void disable()
```

Disables the PowerToy, should free as much memory as possible.

is_enabled

```
virtual bool is_enabled() = 0;
```

Returns the PowerToy state.

destroy

```
virtual void destroy()
```

Destroy the PowerToy and free all memory.

Code organization

powertoy_module_interface.h

Contains the PowerToys interface definition.