Controller(object)
api/v2/base.py

```
__init__(self, plugin, collection, resource, attri_info, ….)
   self._plugin_handlers = {
      self.LIST: ‘get%s_%s’ % (parent_resource, self.collection),
      self.SHOW: ‘get%s_%s’ % (parent_resource, self.resource)
   }
   for action in [self.CREATE, self.UPDATE, self.DELETE]:
      self._plugin_handlers[action] = ‘%s%s_%s’ % (
         action, parent_resource, self.resource)
```

ResourceExtension

```
__init__(self, collection, controller, ….)
   self.collection = collection
   self.controller = controller # legacy controller
```

NeutronManager(object)

```
Load core plugin and service plugins based on config
__init__(self, options, config_file)
   self.plugin = self._get_plugin_instance()  # core plugin
   self.service_plugins  = {service_type: plugin}
   self._load_service_plugins()
   self.resource_plugin_mappings = {}
   self.resource_controller_mappings = {}

@classmethod
get_plugin()  # return core plugin
   return cls.get_instance().plugin

@classmethod
set_controller_for_resource(cls, resource, controller)
@classmethod
set_plugin_for_resource(cls, resource, plugin)
```

ExtensionDescriptor(object)

```
get_name()
get_alias()
get_description()
get_extended_resources(self, version)
get_plugin_interface()
update_attributes_map(self, extended_attributes,
                      extension_attrs_map)
get_resources() # return ResourceExtension()
                # find service plugin from neutron
                # manager by service_type
get_pecan_resources()
```

ServicePluginBase(
extensions.PluginInterface,
WorkerSupportedServiceMixin)

```
supported_extension_aliases = []
get_plugin_type()
get_plugin_description()
```

ExtensionManager(object)

```
Load extensions from the configured extension path
__init__(self, path)
get_resources()
get_pecan_resources()
extend_resources(self, version, attr_map)
_load_all_extensions()
_load_all_extensions_from_path(self, path)
   ext_name = mod_name[0].upper() + mod_name[1:]
   instantiate
```

PluginAwareExtensionManager(
ExtensionManager)

```
__init__(self, path, plugins)
@classmethod
get_instance()
   NeutronManager.get_service_plugins()
   cls(extension_path, service_plugins)
```

NeutronPecanController(object)

```
__init__(self, collection, resource, plugin, parent_resource, ...)
   self._plugin_handlers = {
      self.LIST: ‘get%s_%s’ % (parent_resource, self.collection),
      self.SHOW: ‘get%s_%s’ % (parent_resource, self.resource)
   }
   for action in [self.CREATE, self.UPDATE, self.DELETE]:
      self._plugin_handlers[action] = ‘%s%s_%s’ % (
         action, parent_resource, self.resource)

@property
plugin_handlers():
   return self._plugin_handlers
```

ItemController(NeutronPecanController)

CollectionsController(NeutronPecanController)