```
class MultipleInheritanceMeta(type):
    def __new__(cls, name, bases, dct):
        # Ensure a specific base class is always included
        required_base = SomeRequiredBaseClass
        if required_base not in bases:
            raise TypeError(f"{name} must inherit from {required_base.__name__}.")

        # Check if a forbidden base class is present
        forbidden_base = SomeForbiddenBaseClass
        if forbidden_base in bases:
            raise TypeError(f"{name} cannot inherit from {forbidden_base.__name__}.")

        # Additional rules or restrictions can be added here

        # Create the class using the default behavior
        return super().__new__(cls, name, bases, dct)

# Example usage:
class SomeRequiredBaseClass:
    pass

class SomeForbiddenBaseClass:
    pass

class MyClass1(SomeRequiredBaseClass, metaclass=MultipleInheritanceMeta):
    pass

# This will raise a TypeError since it does not inherit from SomeRequiredBaseClass
try:
    class MyClass2(SomeForbiddenBaseClass, metaclass=MultipleInheritanceMeta):
        pass
except TypeError as te:
    print(f"Error: {te}")
```

```
 Error: MyClass2 must inherit from SomeRequiredBaseClass.
```

Start coding or generate with AI.