CASE STUDY

Pelican **Claims Management** Software

Pro GLOBAL

Client: A host of global (re) insurers and legacy acquirers

– operating in both the live and legacy EL/PL market.

EL/PL claims present specific challenges. Liabilities range from casualty/short tail claims to complex long tail claims, which can be submitted many years after the policies were originally underwritten and premiums received.

The problem

Often claims are further complicated by the existence of legacy technology systems which are not fit for purpose to manage the many challenges associated with these claims e.g.

- Multiple stakeholders creates
  - Duplication of effort
  - Multiple versions of the truth
  - Data protection risks
- Complex apportionments
- Excesses/limits of indemnity
- Erosion of aggregate deductibles
- Requirement for front end controls to minimise leakage
- Need for smart workflow tooling to optimise resource
- Inability to drive insight through data

PRO GLOBAL

PELICAN

Our Approach

Pro has created a purpose built platform called Pelican that provides a consolidated and integrated platform to manage all EL/PL claims.

The software is cloud-based and built in state of the art Azure.net platform. This means it can be accessed by all relevant stakeholders securely and can be modified easily to meet the specific requirements of each client.

Pelican is a secure system that creates a single version of the truth and eliminates the need for double-keying and duplication.

Pelican delivers significant automation of complex calculations and reduces the risk of human error and drives operational efficiencies, whilst maximising outcomes for our clients including more accurate reserving.

```
    mirror_mod.use_y = True
    mirror_mod.use_z = False
elif _operation == "MIRROR_Z":
    mirror_mod.use_x = False
    mirror_mod.use_y = False
    mirror_mod.use_z = True

#selection at the end -add back the deselected mirror modifier ob
rror_ob.select= 1
difier_ob.select=1
py.context.scene.objects.active = modifier_ob
int("Selected" + str(modifier_ob)) # modifier ob is the active ob
```

Thank you for reading

For more information please contact
getintouch@pro-global.com

proglobal.com

Pro
GLOBAL