# Coding with Identity Management & Security

## Part 2 of Identity Management with OpenEdge

Peter Judge
OpenEdge Development
pjudge@progress.com

PROGRESS software

# What is Identity Management?

PROGRESS software

It's about protecting your business data by

- Controlling and verifying who accesses your data
- Controlling what they can do with your data
- Reviewing what they did with your data
- Maintaining information about your users

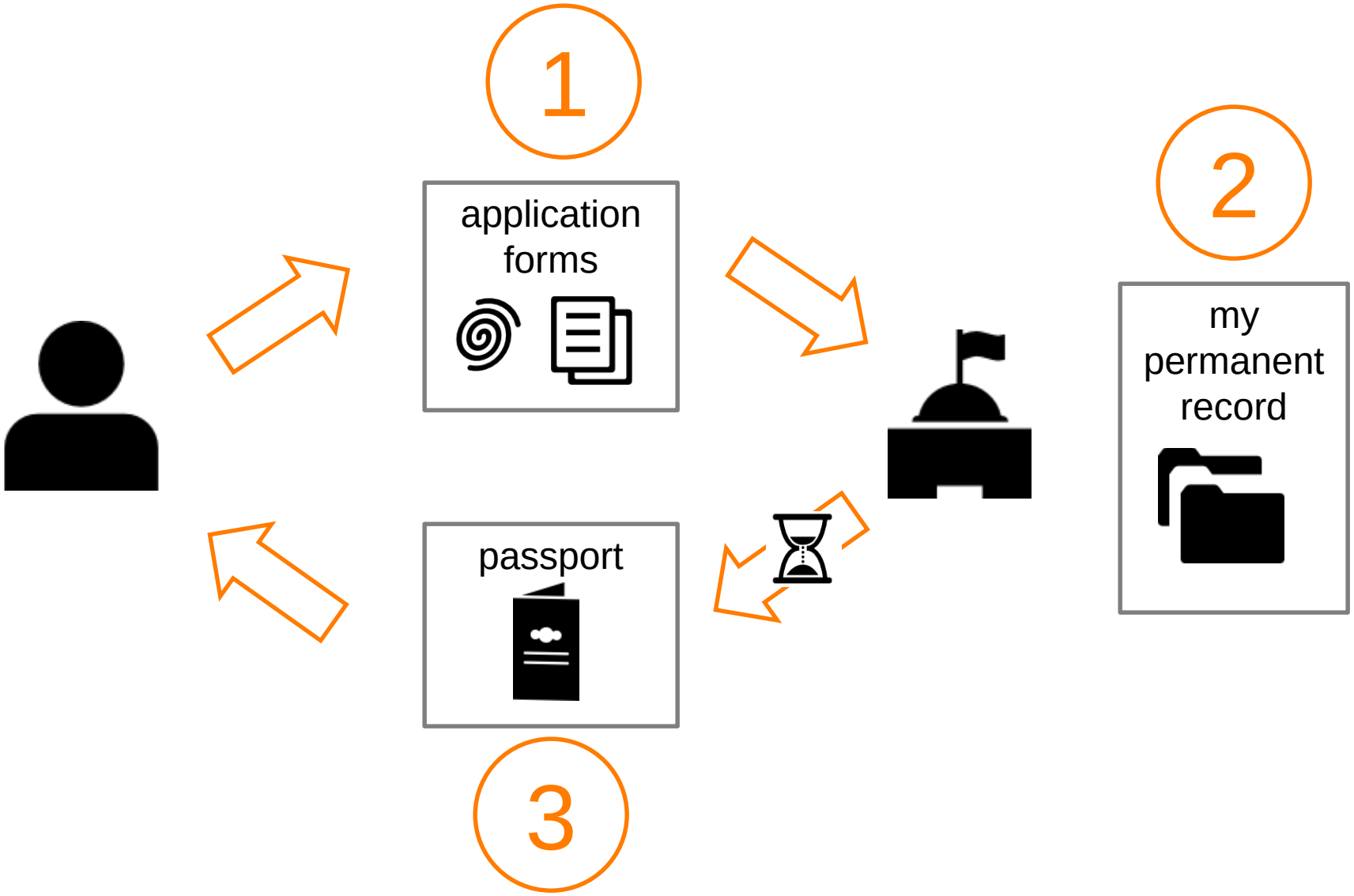
**You** make security decisions on behalf of your customers … understand the maximum loss **they** might suffer

# What is Identity Management?

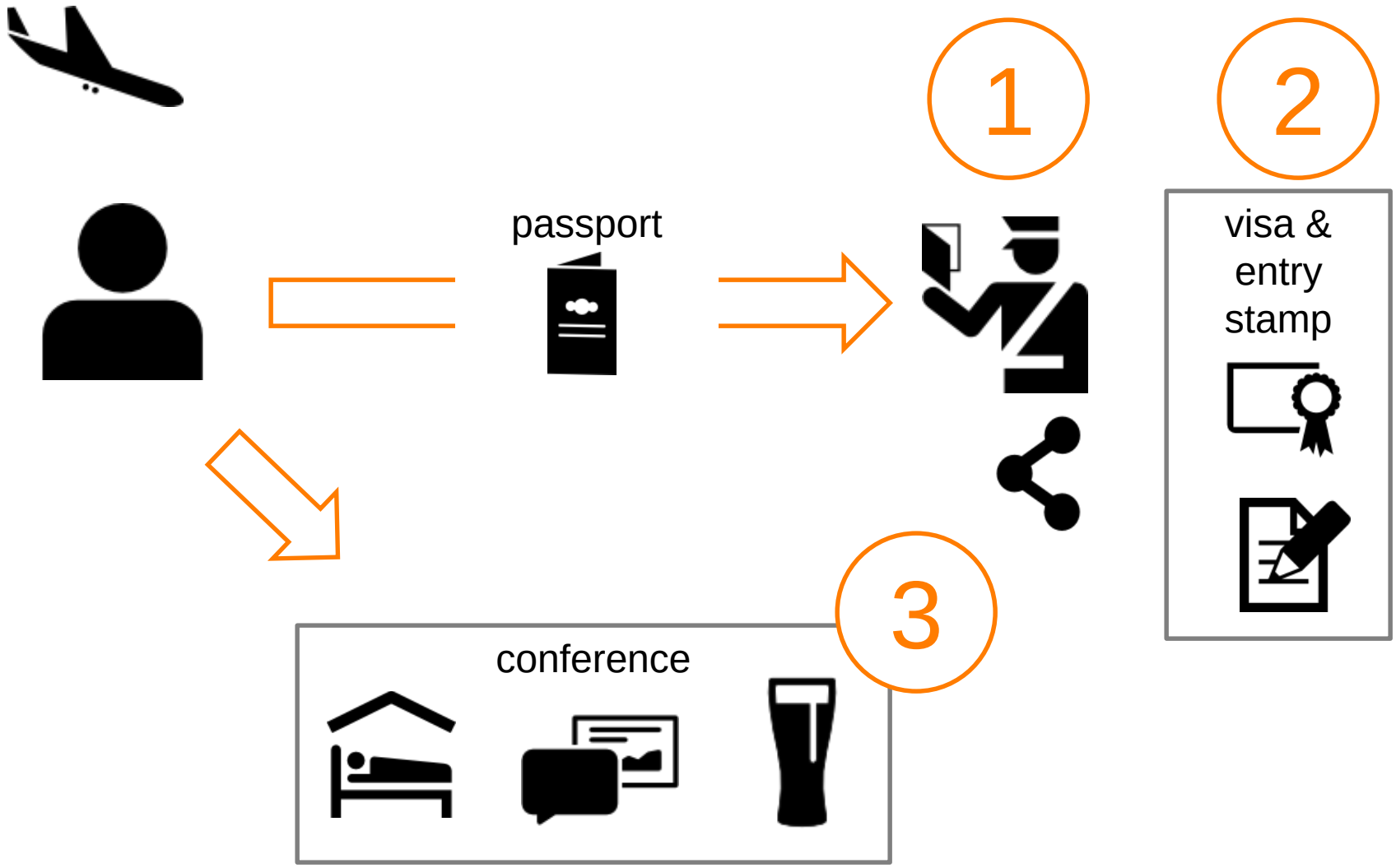
It's about protecting your business data by

- Controlling and verifying who accesses your data — Authentication
- Controlling what they can do with your data — Authorisation
- Reviewing what they did with your data — Auditing
- Maintaining information about your users — Administration

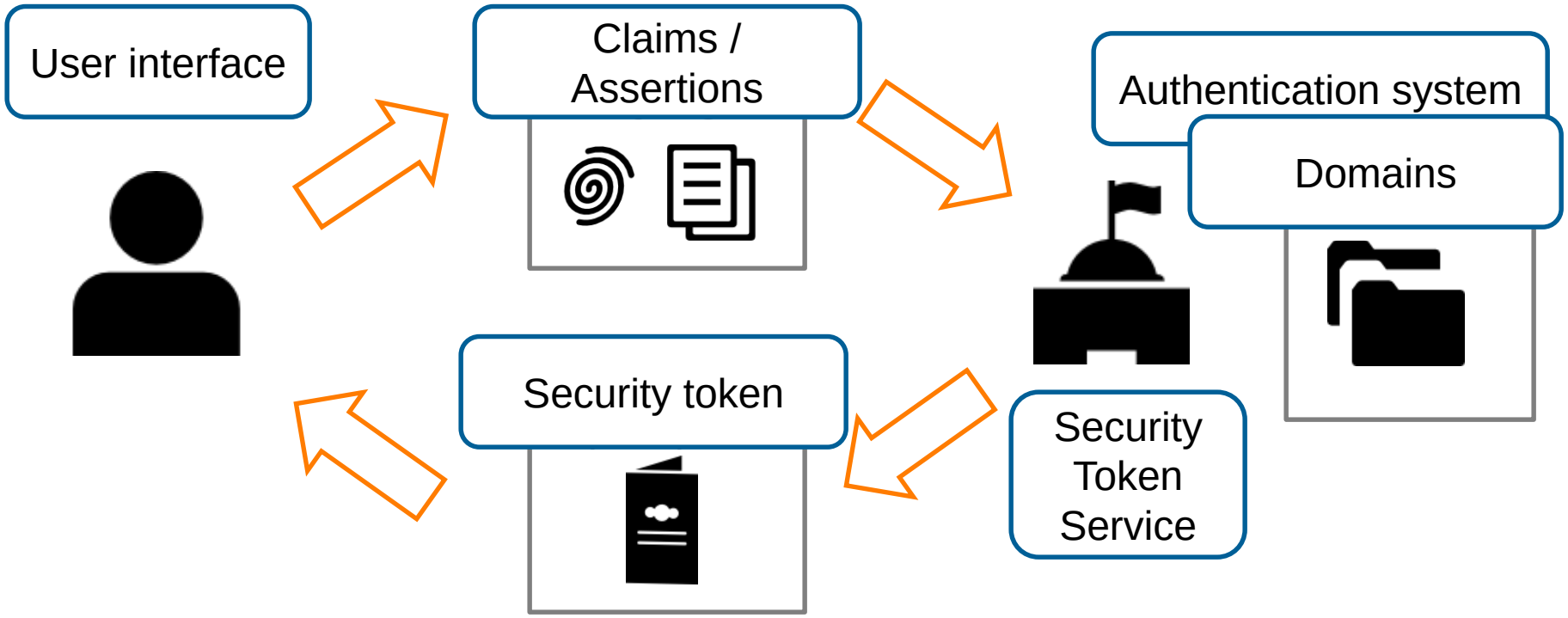
# Getting a Passport



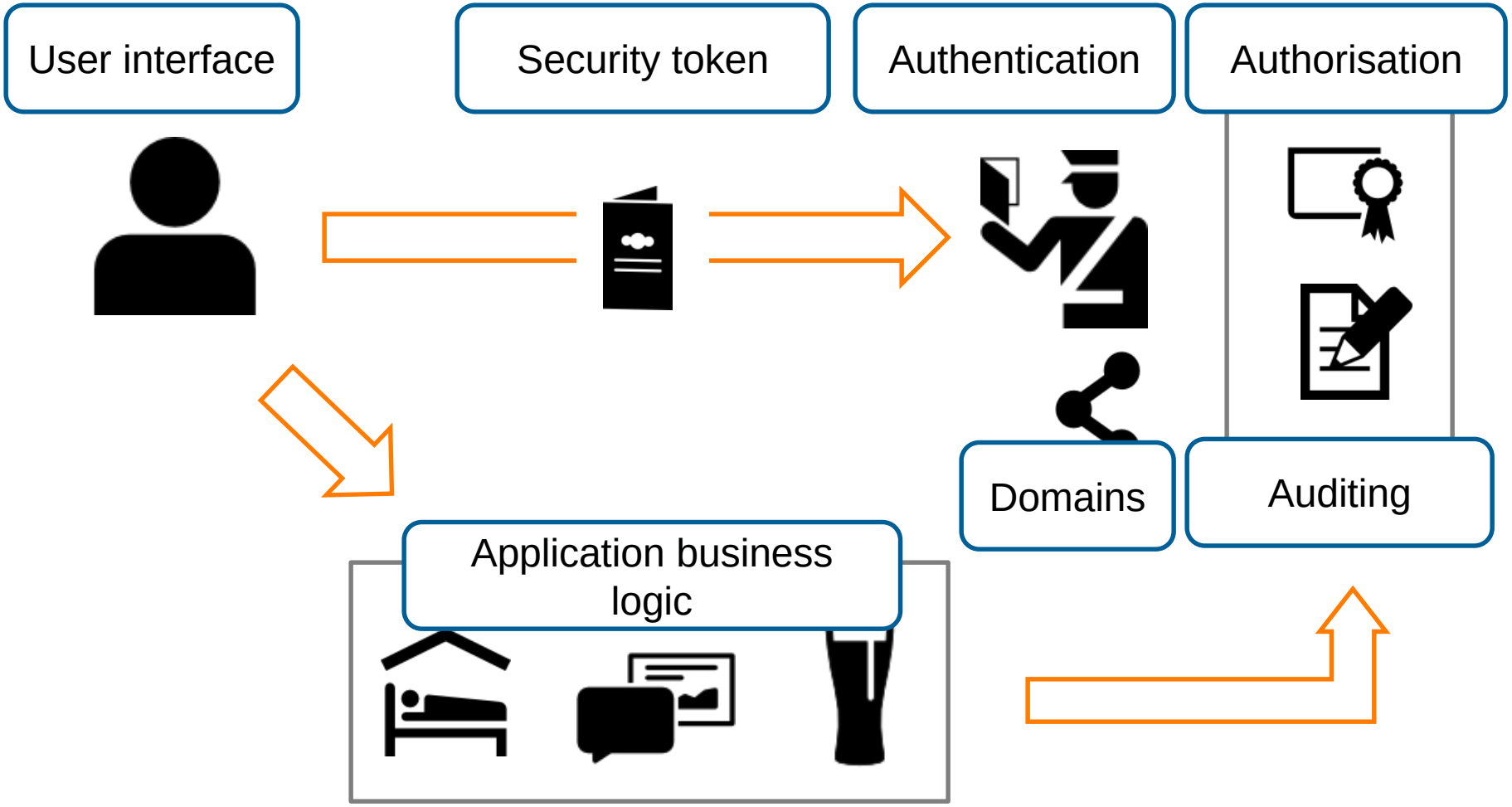
# Using a Passport



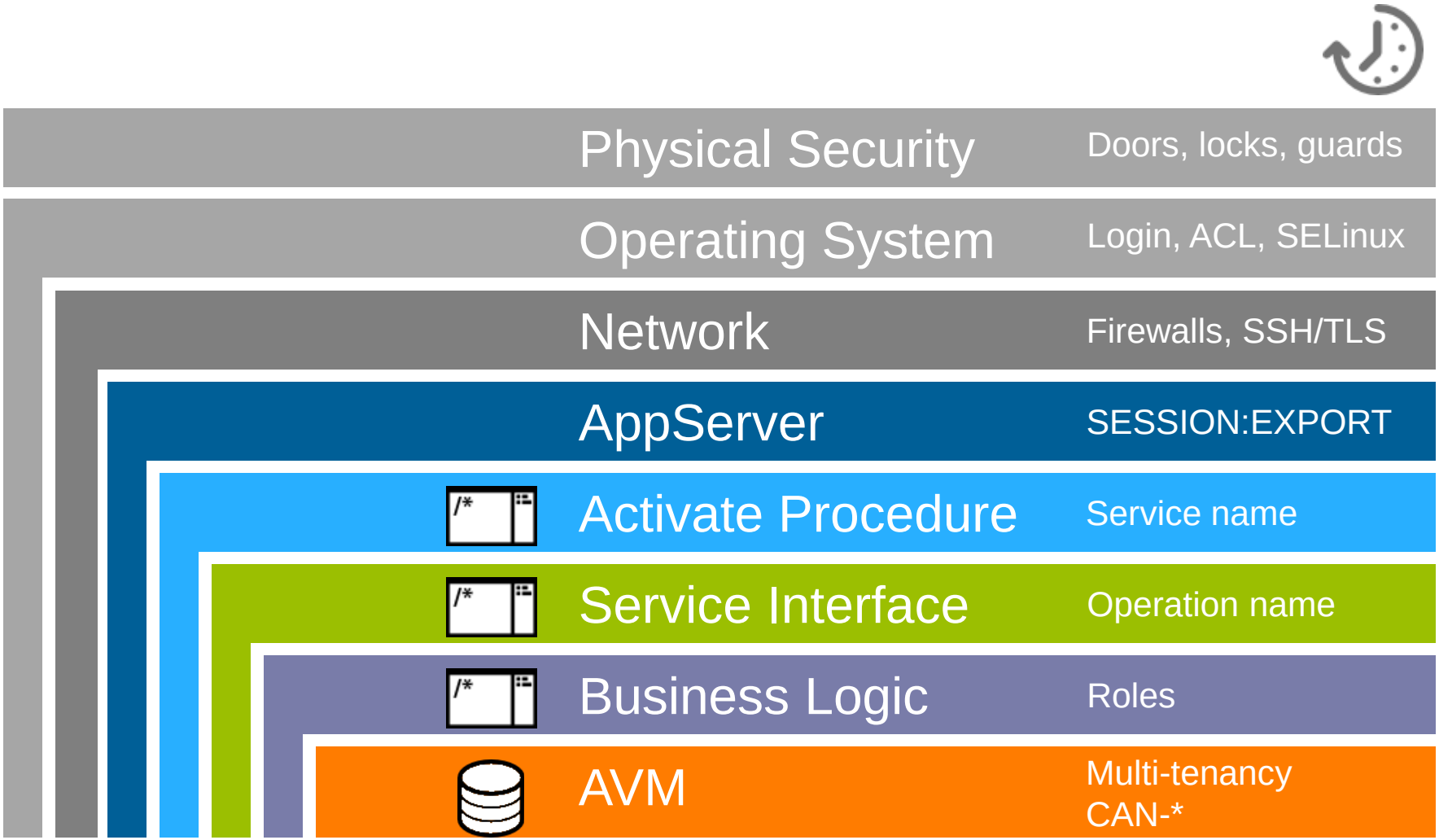
# Application Architecture



# Application Architecture



# Authorisation: Defence in Depth



# ~~If~~ When Authorisation Fails

- Record
- Rewind
  - Deeper you are in the stack, the harder it is to unwind
  - Deeper you are in the stack, the less info you have
- Return
  - Nondescript error messages

```
~~undo, throw new AppError(~~
~~  substitute("User &1 not authorised for service &2",~~
~~              phCP:qualified-user-id, pcServiceName)).~~

undo, throw new AppError(
  "User not authorised for service").
```

# Roles & Responsibilities

| | Anonymous | Customer | Employee | System |
|---|---|---|---|---|
| See catalogue | X | X | X | |
| Modify catalogue | | | X | |
| Update shopping cart | | X | X | |
| Add users | | | X | |
| Dump & load data | | | | X |
| Provision services | | | | X |
| Level of trust | | | | |

# _File Operations

```
find _File where _File-Name eq "Customer"
_Desc       : "Customer master data"
_Can-Create : "*,!*@customer"
_Can-Write  : "*,!*@customer"
_Can-Delete : "*,!*@customer"
_Can-Read   : "*"

find _File where _File-Name eq "Order"
_Desc       : "Order header data"
_Can-Create : "*"
_Can-Write  : "*"
_Can-Delete : "*"
_Can-Read   : "*"

find _File where _File-Name eq "ApplicationUser"
_Desc       : "Application login data"
_Can-Create : "!batch@system,*@system"
_Can-Write  : "!batch@system,*"
_Can-Delete : "!batch@system,*@system"
_Can-Read   : "!batch@system,*"
```

OE 3.0+ | OE 11.0+

# What Are Domains?

- A group of users with a common set of
  - Roles and responsibilities
  - Level of security
  - Data access privileges
- Configured in db meta-schema
  - Authentication systems
  - Tenants

```
_sec-authentication-domain

  _Domain-name
  _Domain-type
  _Domain-description
  _Domain-access-code
  _Domain-runtime-options
  _Tenant-name

  _Domain-enabled
```

OE 10.1A+ | OE 11.0+ | OE 11.1+

# Per-role Domains

Customer

```
create _sec-authentication-domain.
_Domain-name        = 'customer'.
_Domain-type        = 'TABLE-ApplicationUser'.
_Domain-access-code = audit-policy:encrypt-audit-mac-key(
                    's00perSecr1tK3y4CUSTOMER').
_Domain-enabled     = true.
```

Anonymous

```
create _sec-authentication-domain.
_Domain-name        = 'anonymous'.
_Domain-type        = 'FTP'.
_Domain-access-code = audit-policy:encrypt-audit-mac-key(
                    's00perSecr1tK3y4ANONYMOUS').
_Domain-enabled     = true.
```

Employee

```
create _sec-authentication-domain.
_Domain-name        = 'employee'.
_Domain-type        = 'TABLE-ApplicationUser'.
_Domain-access-code = audit-policy:encrypt-audit-mac-key(
                    's00perSecr1tK3y4EMPLOYEE').
_Domain-enabled     = true.
```

# Per-role Domains, Ctd.

System

```
create _sec-authentication-domain.
_Domain-name        = 'system'.
_Domain-type        = '_extsso'.
_Domain-access-code = audit-policy:encrypt-audit-mac-key(
                    's00perSecr1tK3y4SYSTEM').
_Domain-enabled     = true.
```

# Per-role Domains, Ctd.

System

```
create _sec-authentication-domain.
_Domain-name        = 'system'.
_Domain-type        = '_extsso'.
_Domain-access-code = audit-policy:encrypt-audit-mac-key(
                    's00perSecr1tK3y4DBSYSTEM').
_Domain-enabled     = false.
```

DB Admin

```
create _sec-authentication-domain.
_Domain-name        = 'db-admin'.
_Domain-type        = '_oslocal'.
_Domain-access-code = audit-policy:encrypt-audit-mac-key(
                    's00perSecr1tK3y4DBADMIN').
_Domain-enabled     = true.
```

App Admin

```
create _sec-authentication-domain.
_Domain-name        = 'app-admin'.
_Domain-type        = '_extsso'.
_Domain-access-code = audit-policy:encrypt-audit-mac-key(
                    's00perSecr1tK3y4APPADMIN').
_Domain-enabled     = true.
```

# Per-role Domains with Multi-tenancy

PROGRESS software

Customer Tenant 1

```
create _sec-authentication-domain.
_Domain-name        = 'customer.tenant1'.
_Domain-type        = 'TABLE-ApplicationUser'.
_Domain-access-code = audit-policy:encrypt-audit-mac-key(
                    's00perSecr1tK3y4CUSTOMER.T1').
_Domain-enabled     = true.
_Tenant-name        = 'tenant-ONE'.
```

Customer Tenant 2

```
create _sec-authentication-domain.
_Domain-name        = 'customer.tenant2'.
_Domain-type        = 'TABLE-ApplicationUser'.
_Domain-access-code = audit-policy:encrypt-audit-mac-key(
                    's00perSecr1tK3y4CUSTOMER.T2').
_Domain-enabled     = true.
_Tenant-name        = 'tenant-TWO'.
```

Customer Tenant 3

```
create _sec-authentication-domain.
_Domain-name        = 'customer.tenant3'.
_Domain-type        = 'TABLE-ApplicationUser'.
_Domain-access-code = audit-policy:encrypt-audit-mac-key(
                    's00perSecr1tK3y4CUSTOMER.T3').
_Domain-enabled     = true.
_Tenant-name        = 'tenant-THREE'.
```

# Multiple Domains per Tenant

PROGRESS software

Tenant 1
Customer

```
create _sec-authentication-domain.
_Domain-name        = 'customer.tenant1'.
_Domain-type        = 'TABLE-ApplicationUser'.
_Tenant-name        = 'tenant-ONE'.
```

Tenant 1
Employee

```
create _sec-authentication-domain.
_Domain-name        = 'employee.tenant1'.
_Domain-type        = 'LDAP'.
_Tenant-name        = 'tenant-ONE'.
```

Tenant 1
System

```
create _sec-authentication-domain.
_Domain-name        = 'system.tenant1'.
_Domain-type        = '_oslocal'.
_Tenant-name        = 'tenant-ONE'.
```

# Roles & Responsibilities

| | Anonymous | Customer | Employee | System |
|---|---|---|---|---|
| See catalogue | X | X | X | |
| Modify catalogue | | | X | |
| Update shopping cart | | X | X | |
| Add users | | | X | |
| Dump & load data | | | | X |
| Provision services | | | | X |
| Level of trust | ■□□ | ■■□ | ■■□ | ■■■ |

# Roles

- Roles a way of mapping sets of capabilities to classes of users
- May not serve the principle of least privilege

  (which states that one should have the minimal privileges necessary, and no more)
- On the other end of the spectrum, one can define one role for every set of resource capabilities one might want to allow
- Map roles to static sets of capabilities

Role definition from OWASP https://www.owasp.org/index

# Configuration: Roles

```
create _Sec-role.
_Role-name = 'ShoppingCart.Data.Write'.
_Role-creator = 'app-admin@system'.
_Role-description = 'Allows users to mod the shopping cart table'.

create _Sec-granted-role.
_Role-name = 'ShoppingCart.Data.Write'.
_Grantee = 'amy@customer'    /* one record per user */
_Grantor = 'app-admin@system'.

create _Sec-role.
_Role-name = 'Customer.Data.Read'.
_Role-creator = 'app-admin@system'.
_Role-description = 'Allows users to read the customer table'.

create _Sec-granted-role.
_Role-name = 'Customer.Data.Read'.
_Grantee = 'amy@customer'.   /* one record per user */
_Grantor = 'app-admin@system'.
```

OE 10.1A+

# Using Roles

```
/* when the user has been successfully authenticated */
/* hCP:qualified-user-id = amy@customer */

for each _Sec-granted-role where
   _Grantee = hCP:qualified-user-id:

  hCP:roles = hCP:roles + ','
            + _Sec-granted-role._Role-name .
end.

/* Roles: ShoppingCart.Data.Write, Customer.Data.Read ... */
```

```
/* later, when the user attempts access */
pcOperation = 'ShoppingCart.Data.Read'.
if not can-do(hCP:roles, pcOperation) then
    undo, throw new AppError(
              'User not authorised for operation').
```

# Configuration: Operation Access

```
create _Sec-role.
_Role-name = 'Customer.Service.GetCustomers'.
_Role-creator = 'app-admin@system'.
_Role-description =
    'Allows users to access the Get Customer operation'.

create _Sec-role.
_Role-name = 'Customer.Service.GetCustomerOrders'.
_Role-creator = 'app-admin@system'.
_Role-description =
    'Allows users to access the Customer Order operation'.

create _Sec-role.
_Role-name = 'ShoppingCart.Service.UpdateCart'.
_Role-creator = 'app-admin@system'.
_Role-description =
    'Allows users to access the update a shopping cart'.
```

# Using Operation Access

```
/* BusinessLogic/si_ShoppingCartService.p */
routine-level on error undo, throw.

{dsShoppingCart.i}

procedure UpdateShoppingCartService:
  define input-output parameter dataset for dsShoppingCart.

  Security.AuthorisationService:AuthoriseOperation(
       'ShoppingCart.Service.UpdateCart').

  ShoppingCartService:Instance:UpdateShoppingCartService(
      input-output dataset dsShoppingCart by-reference).
end procedure.
```

# Roles & Responsibilities

| | Anonymous | Customer | Employee | System |
|---|---|---|---|---|
| See catalogue | X | X | X | |
| Modify catalogue | | | X | |
| Update shopping cart | | X | X | |
| Add users | | | X | |
| Dump & load data | | | | X |
| Provision services | | | | X |
| Level of trust | | | | |

# Configuration: Service Names

```
create _Sec-role.
_Role-name = 'Customer.Service.Access'.
_Role-description = 'Allows access to the Customer service'.

_Role-name = 'ShoppingCart.Service.Access'.
_Role-description = 'Allows access to the ShoppingCart service'.
```

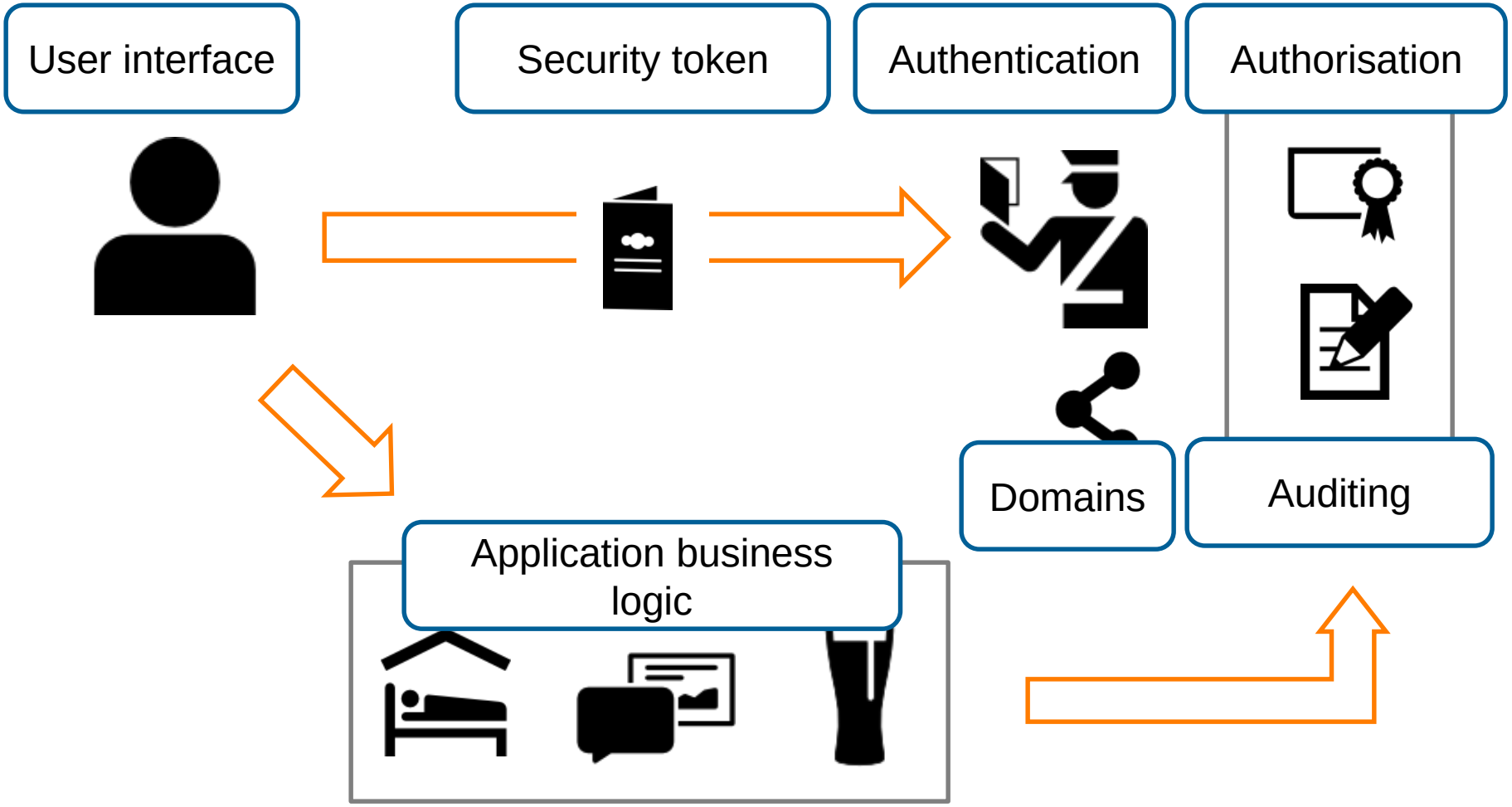
```
define private temp-table ttService no-undo
  field service as character
  field role    as character

create ttService.
 service = "BusinessLogic/si_ShoppingCartService.p".
 role    = "ShoppingCartService.Service.Access".

create ttService.
 service = "BusinessLogic/GetCustomers.p"
 role    = "Customer.Service.Access"

create ttService.
 service = "BusinessLogic/si_GenericFetchData.p".
 role    = "ShoppingCartService.Service.Access".
```

# AppServer Access

```
/* ttService populated from database or config files */
define private temp-table ttService no-undo
  field service as character
  field role    as character

define variable cDelim as character no-undo.
define variable cExportList as character no-undo.

cDelim = ''.

for each ttService break by service:
    if first-of(ttService.service) then
        assign cExportList = cExportList
                           + cDelim
                           + ttService.service
               cDelim = ','.
end.

session:export(cExportList).
```

OE ~9.0+

# Application Architecture: Business Logic



# Application Architecture: Business Logic

PROGRESS software

```
RUN getcustomerlist.p ON SERVER hAppServer
    (OUTPUT DATASET dsCustomer)
```

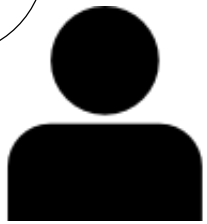
1

Security token

Authentication

Authorisation

2



```
activate.p

STS:ValidateToken
  (INPUT cToken).

security-policy:set-client
  (<<user>>)

AuthoriseService
  ("getcustomer.p")
```

Application business

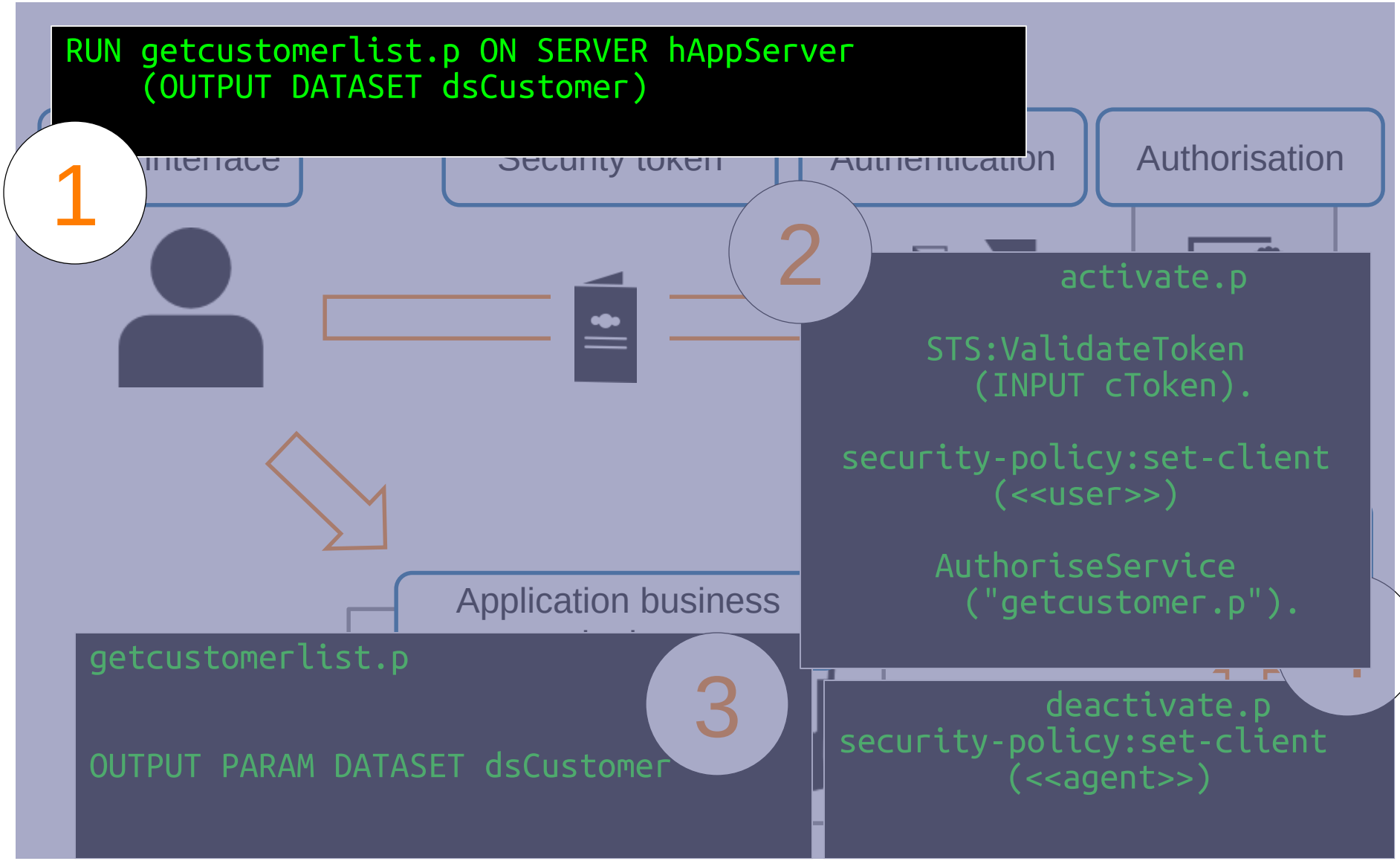
```
getcustomerlist.p


OUTPUT PARAM DATASET dsCustomer
```

3

4

```
deactivate.p
security-policy:set-client
  (<<agent>>)
```

# Application Architecture: Business Logic

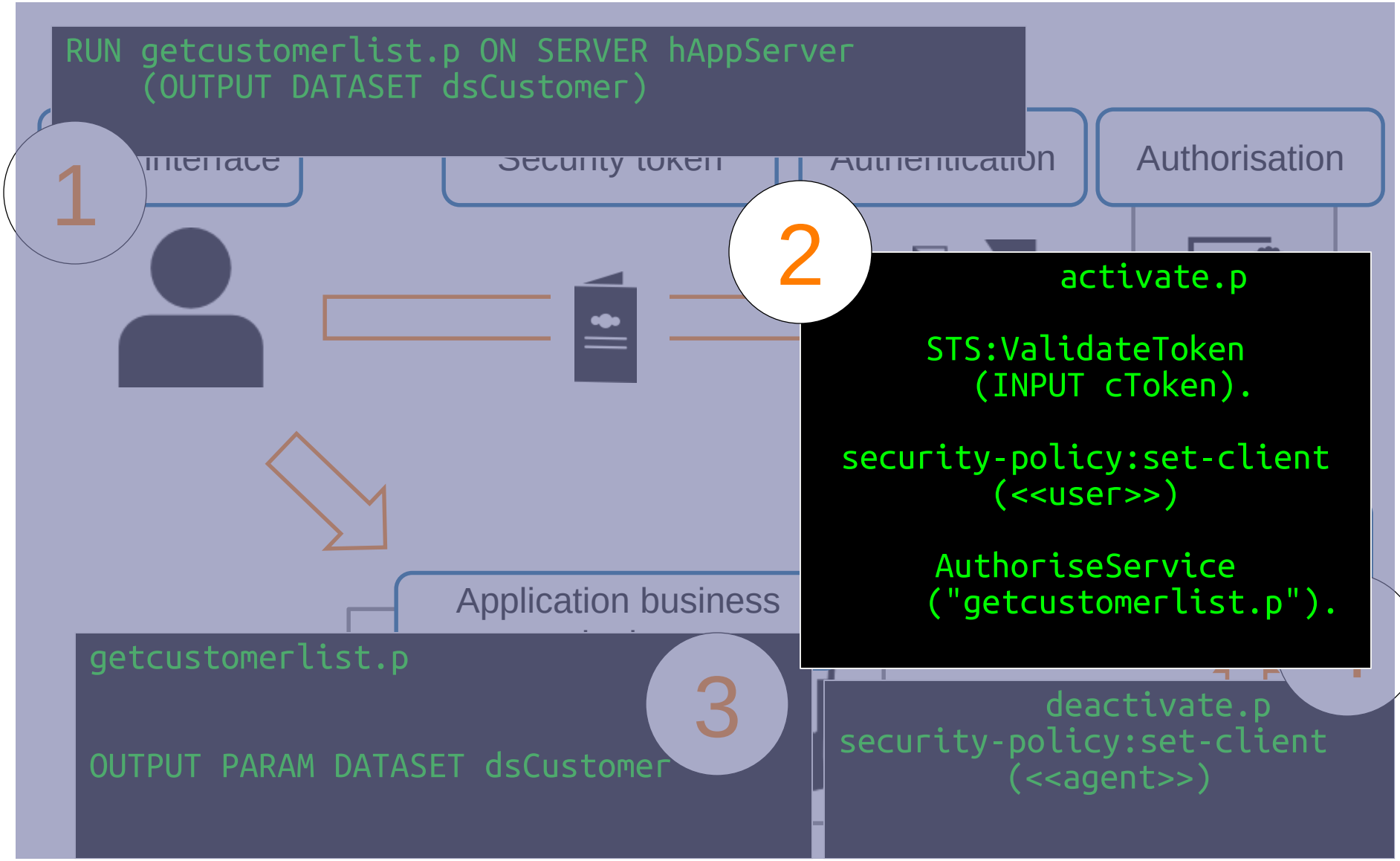


# Desktop.MainForm.cls

```
method protected void RefreshCustomerList():
  define variable hAppServer as handle no-undo.

  run BusinessLogic/GetCustomerList.p on hAppServer
                          (output dataset dsCustomerOrder).

  open query qryCustomer preselect
     each ttCustomer by ttCustomer.CustNum.

  bsCustomer:Handle = query qryCustomer:handle.

  query qryCustomer:reposition-to-row(1).
end method.
```

# Application Architecture: Business Logic

# Application Architecture: Business Logic



# Security/Activate.p

```
hClientPrincipal = Security.SecurityTokenService:Instance:
    GetClientPrincipal(
        session:current-request-info:ClientContextId).

/* authenticate client-principal */
security-policy:set-client(hClientPrincipal).

/* authorise service access */
Security.AuthorisationService:Instance
    :AuthoriseService(
        hClientPrincipal,
        session:current-request-info:ProcedureName).
```

# Security/Activate.p

```
hClientPrincipal = Security.SecurityTokenService:Instance:
    GetClientPrincipal(
        session:current-request-info:ClientContextId).

/* authenticate client-principal */
security-policy:set-client(hClientPrincipal).

/* authorise service access */
Security.AuthorisationService:Instance
    :AuthoriseService(
        hClientPrincipal,
        session:current-request-info:ProcedureName).
```

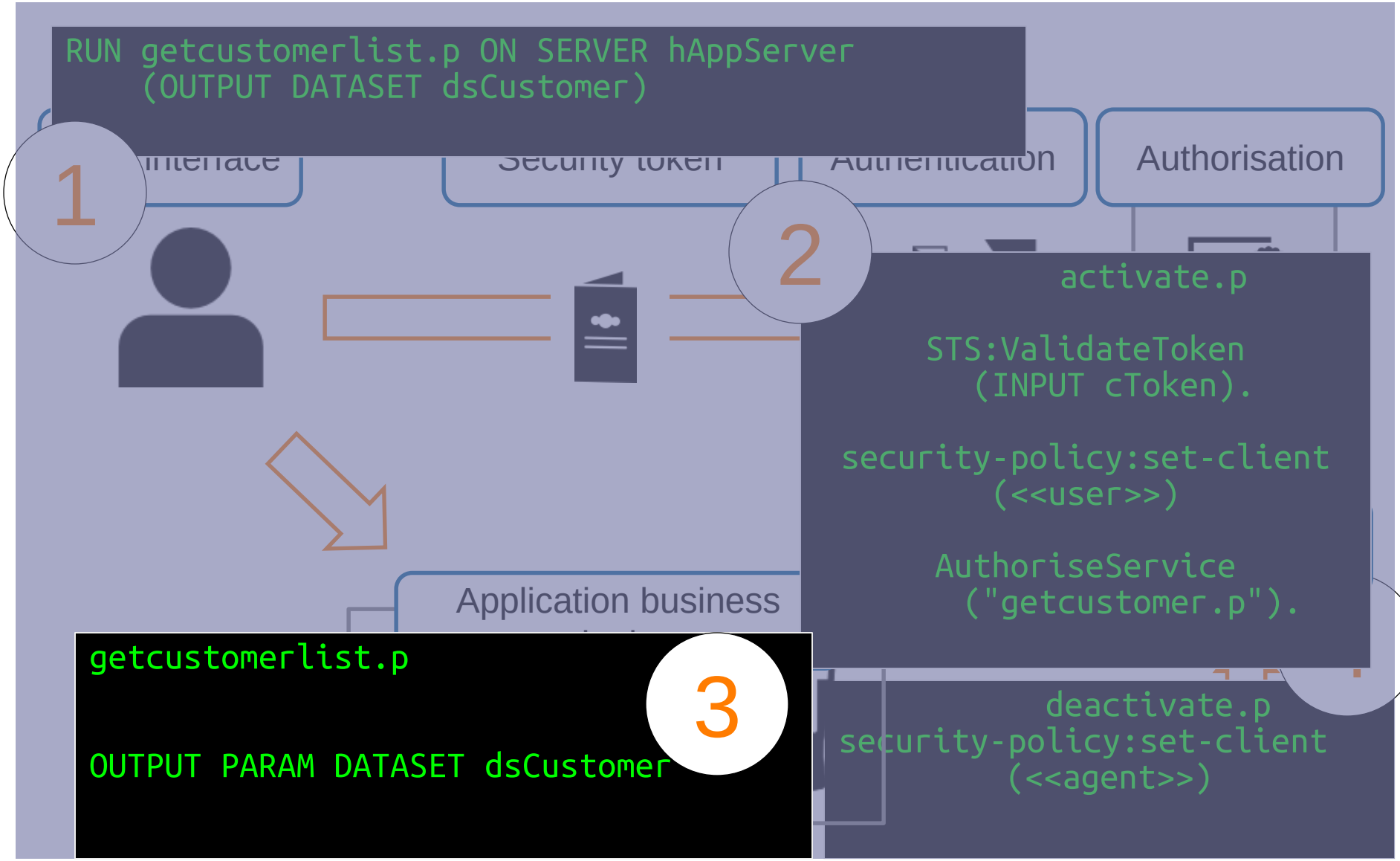
# Security/Activate.p

```
hClientPrincipal = Security.SecurityTokenService:Instance:
    GetClientPrincipal(
        session:current-request-info:ClientContextId).

/* authenticate client-principal */
security-policy:set-client(hClientPrincipal).

/* authorise service access */
Security.AuthorisationService:Instance
    :AuthoriseService(
        hClientPrincipal,
        session:current-request-info:ProcedureName).
```

# Security.AuthorisationService

```
define private temp-table ttService no-undo
  field service as character
  field role    as character
  index idx1 as primary service.

/* pcServiceName: BusinessLogic/GetCustomerList.p */
method public void AuthoriseService(
                      input phCP as handle,
                      input pcServiceName as character):
  define variable lIsAuthorised as logical no-undo.

  lIsAuthorised = can-find(
      first ttService where ttService.service eq pcServiceName).

  for each ttService where ttService.service eq pcServiceName
      while lIsAuthorised:
     lIsAuthorised = not can-do(phCP:roles, ttService.role).
  end.

  if not lIsAuthorised then
    undo, throw new AppError("User not authorised for service").
end method.
```

# Application Architecture: Business Logic



# BusinessLogic/GetCustomerList.p

```
{BusinessLogic/dsCustomerOrder.i}

define output parameter dataset for dsCustomerOrder.

define variable oBusinessEntity as CustomerOrderBE no-undo.

Security.AuthorisationService:Instance
    :AuthoriseOperation("Customer.Service.GetCustomers").

oBusinessEntity  = new CustomerOrderBE().

oBusinessEntity:GetCustomers(
    output dataset dsCustomerOrder).

/* eof */
```

# Security.AuthorisationService

```
method public void AuthoriseOperation(
                        input pcOperation as character):
  define variable hCP as handle no-undo.

  hCP = security-policy:get-client().

  if not can-do(hCP:roles, pcOperation) then
    undo, throw new AppError("User not authorised for service").
end method.
```

# BusinessLogic/GetCustomerList.p

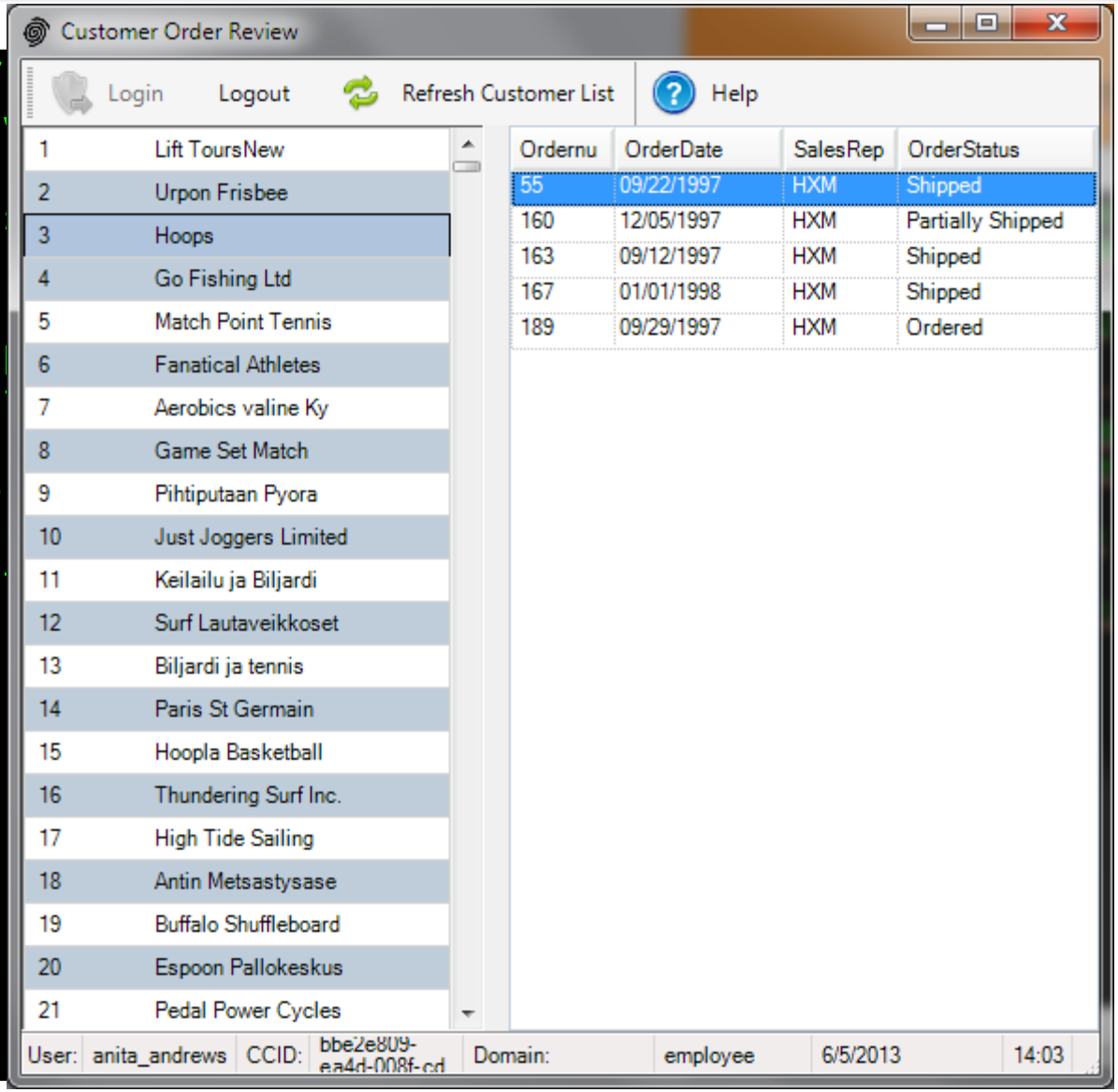
```
{BusinessLogic/dsCustomerOrder.i}

define output parameter dataset for dsCustomerOrder.

define variable oBusinessEntity as CustomerOrderBE no-undo.

Security.AuthorisationService:Instance
    :AuthoriseOperation("GetCustomers").

oBusinessEntity  = new CustomerOrderBE().

oBusinessEntity:GetCustomers(
    output dataset dsCustomerOrder).

/* eof */
```

# CustomerOrderBE.cls

```
{BusinessLogic/dsCustomerOrder.i}

method public void GetCustomers(
             output parameter dataset dsCustomerOrder):
  define data-source srcCustomer for Customer.
  define data-source srcOrder for Order.

  buffer ttCustomer:attach-data-source(
              data-source srcCustomer:handle).
  buffer ttOrder:attach-data-source(
              data-source srcOrder:handle).

  /* multi-tenancy magic happens here.
     CAN-READ too.
     based on the asserted user via SECURITY-POLICY
     GET-CLIENT */
  dataset dsCustomerOrder:fill().

  buffer ttCustomer:detach-data-source().
  buffer ttOrder:detach-data-source().

end method.
/* eof */
```

# Desktop.MainForm.cls

```
method protected void Ref
  define variable hAppSer

  run BusinessLogic/GetCu


  open query qryCustomer
     each ttCustomer by t

  bsCustomer:Handle = que

  query qryCustomer:repos
end method.
```



# Application security principles

Applications must have security designed in. Some proven application security principles

1. **Identify and secure the weakest link**
2. **Practice defense in depth**
3. Be reluctant to trust
4. Remember that hiding secrets is hard
5. **Follow the principle of least privilege**
6. **Fail and recover securely**
7. Compartmentalize
8. Keep it simple, stupid
9. Keep trust to yourself
10. **Assume nothing**

http://news.cnet.com/2008-1082-276319.html

# Summary

PROGRESS software

- Identity management is a process that helps protect your business data
  - Strength in depth
- OpenEdge provides components of identity management
  - CLIENT-PRINCIPAL
  - Multi-tenancy, Domains & Authentication Systems
- Run with least privilege
  - Use Domains and Roles to keep privileges ‘tight’
  - Reset to lower privileges when done
- Configuration > Code
  - Code is the weakest link

# Extra materials

PROGRESS software

- This session
  - Slides to be posted on PUG Challenge site
  - Supporting code at https://github.com/nwahmaet/IdM_Sample
- Other PUG Challenge sessions
  - Identity Management Basics (Part 1)
    Peter Judge, PSC
  - Advanced OpenEdge REST/Mobile Security
    Mike Jacobs, PSC
  - Programming with the Client-Principal Object
    Chris Longo, BravePoint

Image Credits:
Passport designed by Catia G, Time designed by wayne25uk, Database designed by Anton Outkine, Code designed by Nikhil Dev, Ninja designed by John O'Shea, Imposter designed by Luis Prado, User designed by T. Weber, Fingerprint designed by Andrew Forrester, Document designed by Samuel Green, Certificate designed by VuWorks, Network designed by Ben Rex Furneaux, Beer designed by Leigh Scholten; all from The Noun Project

# PROGRESS EXCHANGE 2013

DISCOVER. DEVELOP. DELIVER.

**October 6–9, 2013 • Boston**

**#PRGS13**

**www.progress.com/exchange-pug**

Special **low rate of $495** for PUG Challenge attendees with the code **PUGAM**

And visit the Progress booth to learn more about the Progress App Dev Challenge!

PROGRESS
software