User Authentication using the Client Principle Object

Presented By: Chris Longo

BravePoint
Agenda

- What is the Client Principal Object?
- Why is it useful?
- How do I implement the CP Object?
Application Context

● Unique set information associated with a specific user's application session.

● UserID, PlantID, Session

● Effects:
  ● Authentication
  ● Authorization
  ● Query Filtering
  ● Conditional Processing
Stateful App Environment

- User application sessions are uniquely bound to a single OpenEdge Client.
- Context persists on the OpenEdge Client
  - Shared Vars
  - Persistent procedures
  - UserID()
Stateless App Environment

- User application sessions share OpenEdge Clients.
- User Context must be re-establish with each OpenEdge Client Interaction.
What is a Client Principal Object?

- Dynamic ABL Object
  - Attribute / Methods
- Maintains a User's Identity
  - UserID / Roles
  - SessionID / Session Expiration
- Sets effective UserID() for a database.
- Does not authenticate UserID and Password
Importance of a CP Object

- Establish User Context
- Maintaining a user's identity in a stateless environment.
- Used to maintain an identity authenticated using an external registry other then _User.
  - Application specific user registry
  - LDAP
- Auditing
Establishing a User's Identity

- OpenEdge Client connect to a database:
  - Authenticate using _User table
  - Login.p
  - Provide -U <userid> -P <passwd>
  - Setuserid() UserID() functions provides identity context for the connected databases.
Establishing a User's Identity

- Application Tables / External Registry
  - Application specific code to Authenticate UserID and Password.
  - May not have an effect on UserID value set for the connected database.
  - Use the CP Object to apply an application user's identity.
AppServer/Webspeed Agents

- Client Session Identity is established as an agent connects to a database.
  - Most likely at startup
- Agent is shared by many users but the Identity remains set to the UserID of the process that started the Agents.
Session Context

• A User's Identity is part of application session context.
  • Established between and client and an agent with each interaction.

• Need UserID function to recognize session context.
  • Specifically a user's identity.
CP Object

- The CP Object becomes part of a user's session context.

- It can be used to set the UserIDs of all connected databases at run-time.
Steps to Using CP Object

- Establish an Authentication Domain
- Create CP Object
- Assign three key attribute
  - UserID
  - Domain Name
  - SessionID
- Seal CP Object
  - Domain AccessKey
- Use It
  - Set UserIDs for connected database
Authentication Domains relationship with a CP Object

- Defined internally using the Data Admin Tool
- Alternately defined externally
- Provides encrypted key (access-key).
- Access-key used to seal and validate CP Objects.
Authentication Domain Setup

User Authentication using the Client Principle Object
Security Policy

- An authentication domain must be loaded for a session.
- Security Policy system handle loads domains into the Trusted Domain Registry.
  - security-policy:load-domain('dbName')
    - Domain Registry Locked Automatically
  - security-policy:register-domain('DomainName, AccessKey')
    - security-policy:lock-registration()
CREATE CLIENT-PRINCIPAL hClientPrincipal.

/* Set CP Object Values */

hClientPrincipal:SESSION-ID = BASE64-ENCODERED(GENERATE-UUID).

hClientPrincipal:USER-ID = pcUserID.

hClientPrincipal:DOMAIN-NAME = 'bravepoint.com'.

hClientPrincipal:DOMAIN-TYPE = 'Internal'.

hClientPrincipal:LOGIN-EXPIRATION-TIMESTAMP =

        ADD-INTERVAL(NOW, 60, 'seconds').

hClientPrincipal:ROLES = pcRoles.

hClientPrincipal:SET-PROPERTY('UserPlant', 'Norcross').
Authenticate User Identity

IF Identity.IdentityKey <> ENCODE(pcPasswd) THEN DO:
/* This will set the state-detail attribute */
  hClientPrincipal:AUTHENTICATION-FAILED
    ('UserName Password authenitication failed').

  pcMessage = 'UserName Password authenitication failed'.
END.
CP Object State

• LOGIN-STATE Attribute
  • LOGIN
  • LOGOUT
  • EXPIRED
  • FAILED

• AUTHENTICATION-FAILED()
  • Used on an unsealed CP Object
  • LOGIN-STATE is set to failed.
  • STATE-DETAIL Attribute is set to the supplied reason.
Seal CP Object

- The Domain Access Key was previously defined using the Data Admin tool or setup manually using register-domain().

\[ h\text{ClientPrincipal}:\text{SEAL}(c\text{DomainAccessKey}) \]
Set DB Identity

- SET-DB-CLIENT will set the effective UserID for all connected databases or those explicitly specified.

```
SET-DB-CLIENT(hClientPrincipal)
```
CP Object Portability

- CP Object provides methods to import and export its values.
  - CP Object exports and imports from a raw data type.

```
DEFINE VAR rCP AS RAW NO-UNDO.

rCP = hClientPrincipal:EXPORT-PRINCIPAL().
```
CP Object and Session Context

• Alternative #1:
  • Pass the raw CP Object as a parameter back to the client.
    - Client gets full access to all the CP Objects Attributes.
    - Raw data type might present issue with non ABL clients.
    - Security threat?
CP Object and Session Context

- Alternative #2
  - Store the CP Object in a session context DB Table.
    - `CPObject.SessionID AS CHARACTER`
    - `CPObject.ContextObject AS RAW`
  - Pass an encrypted token containing the associated sessionID back to the client.
    - `SecureToken` is used to reconstitute the CP Object each time a user interacts with an agent.
    - `SecureToken` is a character string.
CP Object and Session Context

/* Store the CP Object as part of a user's session context. */
rCP = hClientPrincipal:EXPORT-PRINCIPAL().

DO TRANSACTION:
  CREATE bCPObjec.
  ASSIGN
    bCPObjec.SessionID = hClientPrincipal:SESSION-ID
    bCPObjec.ContextObject = rCP.

END.
cSessionID = STRING (DECrypt (BASE64-DECODE (pcSecToken), rEncryptKey))
NO-ERROR.

/* Create an empty CP Object. */
CREATE CLIENT-PRINCIPAL hClientPrincipal.

/* Find the session context row containing the previously saved CP Object Data. */
FIND bCPObjec\tWHERE bCPObjec\t.SessionID = cSessionID NO-LOCK NO-ERROR.
IF NOT AVAIL bCPObjec\tTHEN

/* Load the CP Object. So you left with a CP Object as it existed after you sealed it during createCPObject. */

hClientPrincipal::IMPORT-PRINCIPAL (bCPObjec\t.ContextObject).
Validate CP Object

- **VALIDATE-SEAL**(domainKey)
  - Validates the CPObject's MAC, which was previously generated via the SEAL().
  - Must supply the DomainKey used to seal the CPObject.

- **LOGOUT()**
  - In validates a sealed CP Object
  - No longer use it to set DB UserIDs
  - Sets the LOGIN-STATE to LOGOUT
hClientPrincipal: IMPORT-PRINCIPAL(bCPObject.ContextObject).

IF NOT hClientPrincipal: VALIDATE-SEAL(cDomainAccessKey) THEN DO:
  hClientPrincipal: LOGOUT().
  UNDO, THROW NEW Progress.Lang.AppError(
    SUBSTITUTE('CP Object Validation Failed. Login-State = &1',
    hClientPrincipal: LOGIN-STATE), 105).
END.
Session Expiration

- **SEAL-TIMESTAMP**
  - Automatically set.
  - Date and time of when the CP Object was sealed.
- **LOGIN-EXPIRATION-TIMESTAMP**
  - Programmatically set to some point in the future.
  - LOGIN-STATE set to 'Expired' if not sealed prior to the value set in this attribute.
Session Expiration

/* Check expiration */

IF hClientPrincipal:LOGIN-EXPIRATION-TIMESTAMP < NOW THEN DO:

/* This will set the state-detail attribute */

hClientPrincipal:AUTHENTICATION-FAILED
('User Session Expired.').

hClientPrincipal:LOGOUT().

DO TRANSACTION:

   FIND CURRENT bCPObject EXCLUSIVE-LOCK.
   prCP = hClientPrincipal:EXPORT-PRINCIPAL().
   bCPObject.ContextObject = prCP.

END.

END.
Demo App

User Authentication using the Client Principle Object
User Authentication using the Client Principle Object
User Authentication using the Client Principle Object
Demo App

User Authentication using the Client Principle Object
User Authentication using the Client Principle Object
Questions?