Design Document for User Story 6971

As a user I want the patient portal to use a Security Challenge Questions mechanism to be used when the patient tries to reset their password.

# ASP .NET Pages

### reset\_password.aspx.cs

**Code Type:** C# Codebehind Class for ASP .NET Page

**Uses Classes:** CSec, CSecQuestions

**Uses PL/SQL Procedures:** N/A

**Uses MDWS Calls:** N/A

**Lines of Code:** 143

protected void Page\_Load(object sender, EventArgs e)

* loads the change password page

public void Submit\_Click()

* challenges the user to answer security question in order to proceed to the password reset, if it is a first-time login, user is then asked to select their security challenge questions and the corresponding answer for each

# ASP .NET User Controls

### ucLogin.ascx.cs

**Code Type:** C# Codebehind Class for ASP .NET User Control

**Uses Classes:** CSec, CDataUtils, CPatient

**Uses PL/SQL Procedures:** N/A

**Uses MDWS Calls:** N/A

**Lines of Code:** 45

protected void ResetPasswd\_DirectClick(object sender, ext.DirectEventArgs e)

* redirects the patient to the reset password page

# ASP .NET Classes

### CSecQuestions.cs

**Code Type:** C# Class

**Uses Classes:** CSec, CDataSet

**Uses PL/SQL Procedures:** PCK\_FX\_SEC.GetSecurityQuestions, PCK\_FX\_SEC.GetUserQuestions, PCK\_FX\_SEC.UpdateSecQuestions, PCK\_FX\_SEC.CheckSecurityQuestions, PCK\_FX\_SEC.ResetPassword

**Uses MDWS Calls:** N/A

**Lines of Code:** 125

public DataSet GetSecQuestionsDS(long lQuestionGroup)

* get a list of security questions from the database for the user to pick

public DataSet GetUserQuestions(string strUsername)

* gets user's pre-selected challenge questions

public bool UpdateSecQuestions(long lQ1ID,

string strAnswer1,

long lQ2ID,

string strAnswer2,

long lQ3ID,

string strAnswer3)

* updates user's selection of security questions

public bool ValidateAnswers(long lFXUserID,

string strAnswer1,

string strAnswer2,

string strAnswer3)

* validates the submited answers for the challenge security questions

public bool ResetPassword(long lFXUserID,

string strUserName,

string strPassword)

* proceed to reset account password after responses to the challenge questions are validated

# C++ Classes

N/A

# Oracle PL/SQL

### pck\_fx\_sec.bdy

**Lines of Code:** 1974

procedure CreateSession(pi\_vSessionID in varchar2,

pi\_vSessionClientIP in varchar2,

pi\_nUserID in number,

pi\_vWebSessionID in varchar2,

po\_vDBSessionID out varchar2,

po\_nStatusCode out number,

po\_vStatusComment out varchar2) is

v\_vSql varchar2(4000);

v\_vDBSessionID varchar2(4000) := '';

begin

get a new db session id:

v\_vDBSessionID := '';

v\_vSql := 'select t.SID || :P0 || t.SERIAL# || t.USER# || :P1 || :P2 ' ||

'from sys.v\_$session t ' || 'where t.audsid = :P3 ';

execute immediate v\_vSql

into v\_vDBSessionID

using DBMS\_CRYPTO.RandomBytes(3), DBMS\_CRYPTO.RandomBytes(3), userenv('sessionid'), userenv('sessionid');

clean up any old sessions, users can only login once.

v\_vSql := 'delete from tbicds.fx\_session where fx\_user\_id = :P0';

execute immediate v\_vSql

using pi\_nUserID;

commit;

v\_vSql := 'delete from tbicds.fx\_session where db\_session\_id = :P0';

execute immediate v\_vSql

using v\_vDBSessionID;

commit;

v\_vSql := 'delete from tbicds.fx\_session\_value where db\_session\_id = :P0';

execute immediate v\_vSql

using v\_vDBSessionID;

commit;

insert a record into fx\_session

v\_vSql := 'insert into tbicds.fx\_session (' ||

'fx\_user\_id, web\_session\_id, db\_session\_id, date\_created, date\_last\_action, expired, client\_ip) ' ||

'values (' || ':P0, :P1, :P2, :P3, :P4, :P5, :P6) ';

execute immediate v\_vSql

using pi\_nUserID, pi\_vWebSessionID, v\_vDBSessionID, sysdate, sysdate, 0, pi\_vSessionClientIP;

add an audit record for the success

v\_vSql := 'insert into tbicds.fx\_audit (' ||

'db\_session\_id, client\_ip, fx\_user\_id, audit\_date, audit\_name, audit\_data) ' ||

'values (' || ':P0, :P1, :P2, :P3, :P4, :P5) ';

execute immediate v\_vSql

using v\_vDBSessionID, pi\_vSessionClientIP, pi\_nUserID, sysdate, 'LOGIN', 'SUCCESS';

commit;

keep the session id

po\_vDBSessionID := v\_vDBSessionID;

exception

when others then

rollback;

po\_vDBSessionID := '';

po\_nStatusCode := pck\_common.c\_nStatus\_Error;

po\_vStatusComment := '';

end;

* creates a database session for a user. used after a successful login to an external system such as MDWS to establish a session for the user in the database

procedure ValidatePassword(pi\_vKey in varchar2,

pi\_nUserID in number,

pi\_vUserName in varchar2,

pi\_vOldPassword in varchar2,

pi\_vPassword in varchar2,

pi\_vCOldPassword in varchar2,

pi\_vCPassword in varchar2,

pi\_vCUserName in varchar2,

pi\_nResetPassword in number,

po\_nStatusCode out number,

po\_vStatusComment out varchar2) is

v\_nFXUserID number := 0;

v\_nUserCount number := 0;

v\_dDatePasswordChanged date;

v\_vPREV\_PWD1 varchar2(2000);

v\_vPREV\_PWD2 varchar2(2000);

v\_vPREV\_PWD3 varchar2(2000);

v\_vPREV\_PWD4 varchar2(2000);

v\_vPREV\_PWD5 varchar2(2000);

v\_vPREV\_PWD6 varchar2(2000);

v\_vPREV\_PWD7 varchar2(2000);

v\_vPREV\_PWD8 varchar2(2000);

v\_vPREV\_PWD9 varchar2(2000);

v\_vPREV\_PWD10 varchar2(2000);

suat user personal information

v\_vUsrPDataName varchar2(2000);

v\_vUsrPDataName1 varchar2(2000);

v\_vUsrPDataName2 varchar2(2000);

v\_vUsrPDataTitle varchar2(2000);

v\_vUsrPDataUnit varchar2(2000);

v\_vUsrPDataSquadron varchar2(2000);

v\_vUsrPDataPhone varchar2(2000);

patient personal information

v\_vPatPDataFirstName varchar2(2000);

v\_vPatPDataLastName varchar2(2000);

v\_vPatPDataSSN varchar2(2000);

v\_vPatPDataCity varchar2(2000);

v\_vPatPDataPostalCode varchar2(2000);

v\_vPatPDataHomePhone varchar2(2000);

v\_vPatPDataWorkPhone varchar2(2000);

v\_dtPatPDataDOB date;

v\_vPatPDataEDIPN varchar2(2000);

v\_vPWDCheck varchar2(4000);

v\_vCUserName1 varchar2(2000);

v\_vCUsername2 varchar2(2000);

v\_vUNCheck varchar2(2000);

v\_nResetPWD number;

v\_nPWDDiffCharacterCount number := 0;

v\_nPWDGreatest number := 0;

v\_nPWDLeast number := 0;

begin

po\_nStatusCode := 0; 0 = success

po\_vStatusComment := '';

get data needed for the check

select date\_password\_changed,

PREV\_PWD1,

PREV\_PWD2,

PREV\_PWD3,

PREV\_PWD4,

PREV\_PWD5,

PREV\_PWD6,

PREV\_PWD7,

PREV\_PWD8,

PREV\_PWD9,

PREV\_PWD10,

nvl(t.reset\_password, 0)

into v\_dDatePasswordChanged,

v\_vPREV\_PWD1,

v\_vPREV\_PWD2,

v\_vPREV\_PWD3,

v\_vPREV\_PWD4,

v\_vPREV\_PWD5,

v\_vPREV\_PWD6,

v\_vPREV\_PWD7,

v\_vPREV\_PWD8,

v\_vPREV\_PWD9,

v\_vPREV\_PWD10,

v\_nResetPWD

from tbicds.fx\_user t

where t.fx\_user\_id = pi\_nUserID;

limit user changes to their account passwords

once every 24 hours with the exception of privileged

or administrative users. or in the case where the

pwd was reset this keeps the user from spinning through pwds to get to

an old pwd for convenience

if nvl(pi\_nResetPassword, 0) < 1 then

if (v\_dDatePasswordChanged is not null) and

(sysdate 1 < v\_dDatePasswordChanged) and

not HASUSERRIGHT(v\_nFXUserID, PCK\_COMMON.c\_nAdministratorUR) then

po\_nStatusCode := 1;

po\_vStatusComment := 'Password cannot be changed in less than 24 hours since last change!';

return;

end if;

end if;

make sure the password is not the same as the 10 past passwords

if pi\_nUserID > 0 then

if pi\_vPassword = v\_vPREV\_PWD1 or pi\_vPassword = v\_vPREV\_PWD2 or

pi\_vPassword = v\_vPREV\_PWD3 or pi\_vPassword = v\_vPREV\_PWD4 or

pi\_vPassword = v\_vPREV\_PWD5 or pi\_vPassword = v\_vPREV\_PWD6 or

pi\_vPassword = v\_vPREV\_PWD7 or pi\_vPassword = v\_vPREV\_PWD8 or

pi\_vPassword = v\_vPREV\_PWD9 or pi\_vPassword = v\_vPREV\_PWD10 or

pi\_vPassword = pi\_vOldPassword then

po\_nStatusCode := 1;

po\_vStatusComment := 'New password cannot be the same as recent past password. Please choose a different password!';

return;

end if;

end if;

ensure passwords do not contain personal information such as names, telephone numbers, account names, birthdates, or dictionary words.

for rec in (select t.restricted\_name from fx\_restricted\_name t) loop

if (instr(upper(pi\_vCPassword), upper(rec.restricted\_name)) > 0) then

po\_nStatusCode := 1;

po\_vStatusComment := 'Password cannot contain personal information such as names, telephone numbers, account names, birthdates, or dictionary words!';

return;

end if;

end loop;

new account passwords differ from the previous password by at least four characters when a password is changed.

select greatest(length(pi\_vCOldPassword), length(pi\_vCPassword)),

least(length(pi\_vCOldPassword), length(pi\_vCPassword))

into v\_nPWDGreatest, v\_nPWDLeast

from dual;

v\_nPWDDiffCharacterCount := v\_nPWDGreatest v\_nPWDLeast;

if v\_nPWDDiffCharacterCount < 4 then

for i in 1 .. v\_nPWDLeast loop

if (substr(pi\_vCOldPassword, i, 1) <> substr(pi\_vCPassword, i, 1)) then

v\_nPWDDiffCharacterCount := v\_nPWDDiffCharacterCount + 1;

end if;

exit stop\_loop when v\_nPWDDiffCharacterCount > 3;

end loop; stop\_loop;

if v\_nPWDDiffCharacterCount < 4 then

po\_nStatusCode := 1;

po\_vStatusComment := 'Passwords must be different from the previous password by at least four characters!';

return;

end if;

end if;

for i in 1..(length(pi\_vCPassword) 4)

loop

if (instr(pi\_vCOldPassword, substr(pi\_vCPassword, i, 4)) > 0)

then

po\_nStatusCode := 1;

po\_vStatusComment := 'Passwords must be different from the previous password by at least four characters!';

return;

end if;

end loop;

v\_vPWDCheck := upper(pi\_vCPassword);

v\_vUNCheck := upper(pi\_vCUserName);

v\_vCUserName1 := pck\_common.GetPiece(v\_vUNCheck, '.', 0);

v\_vCUserName2 := pck\_common.GetPiece(v\_vUNCheck, '.', 1);

if instr(v\_vPWDCheck, pi\_vCUserName) > 0 or

instr(v\_vPWDCheck, v\_vCUserName1) > 0 or

instr(v\_vPWDCheck, v\_vCUserName2) > 0 then

po\_nStatusCode := 1;

po\_vStatusComment := 'Password cannot contain elements of your user name!';

return;

end if;

make sure the password does not contain personal information

suat user personal information

begin

select name, Title, Unit, Squadron, Phone

into v\_vUsrPDataName,

v\_vUsrPDataTitle,

v\_vUsrPDataUnit,

v\_vUsrPDataSquadron,

v\_vUsrPDataPhone

from tbicds.app\_user

where fx\_user\_id = v\_nFXUserID;

v\_vUsrPDataName := upper(v\_vUsrPDataName);

v\_vUsrPDataTitle := upper(v\_vUsrPDataTitle);

v\_vUsrPDataUnit := upper(v\_vUsrPDataUnit);

v\_vUsrPDataSquadron := upper(v\_vUsrPDataSquadron);

v\_vUsrPDataPhone := upper(v\_vUsrPDataPhone);

v\_vUsrPDataName1 := pck\_common.GetPiece(v\_vUsrPDataName, ' ', 0);

v\_vUsrPDataName2 := pck\_common.GetPiece(v\_vUsrPDataName, ' ', 1);

if instr(v\_vPWDCheck, v\_vUsrPDataName) > 0 or

instr(v\_vPWDCheck, v\_vUsrPDataName1) > 0 or

instr(v\_vPWDCheck, v\_vUsrPDataName2) > 0 or

instr(v\_vPWDCheck, v\_vUsrPDataTitle) > 0 or

instr(v\_vPWDCheck, v\_vUsrPDataUnit) > 0 or

instr(v\_vPWDCheck, v\_vUsrPDataSquadron) > 0 or

instr(v\_vPWDCheck, v\_vUsrPDataPhone) > 0 or

instr(v\_vPWDCheck, replace(v\_vUsrPDataPhone, '', '')) > 0 then

po\_nStatusCode := 1;

po\_vStatusComment := 'Password cannot contain personal information such as names, telephone numbers, account names, birthdates, or dictionary words!';

return;

end if;

exception

when others then

null;

end;

patient personal information

begin

select tbicds.fnc\_utl\_decstr(t.FIRST\_NAME, pi\_vKey, t.PATIENT\_ID) as first\_name,

tbicds.fnc\_utl\_decstr(t.LAST\_NAME, pi\_vKey, t.PATIENT\_ID) as last\_name,

tbicds.fnc\_utl\_decstr(t.SSN, pi\_vKey, t.PATIENT\_ID) as ssn,

t.City,

t.Postal\_Code,

t.HomePhone,

t.WorkPhone,

to\_date(tbicds.fnc\_utl\_decstr(t.dob, pi\_vKey, t.PATIENT\_ID),

'MM/DD/YYYY') as dob,

t.EDIPN

into v\_vPatPDataFirstName,

v\_vPatPDataLastName,

v\_vPatPDataSSN,

v\_vPatPDataCity,

v\_vPatPDataPostalCode,

v\_vPatPDataHomePhone,

v\_vPatPDataWorkPhone,

v\_dtPatPDataDOB,

v\_vPatPDataEDIPN

from tbicds.patient\_demographics t

where t.fx\_user\_id = v\_nFXUserID;

if instr(v\_vPWDCheck, v\_vPatPDataFirstName) > 0 or

instr(v\_vPWDCheck, v\_vPatPDataLastName) > 0 or

instr(v\_vPWDCheck, v\_vPatPDataSSN) > 0 or

instr(v\_vPWDCheck, replace(v\_vPatPDataSSN, '', '')) > 0 or

instr(v\_vPWDCheck, v\_vPatPDataCity) > 0 or

instr(v\_vPWDCheck, v\_vPatPDataPostalCode) > 0 or

instr(v\_vPWDCheck, v\_vPatPDataHomePhone) > 0 or

instr(v\_vPWDCheck, replace(v\_vPatPDataHomePhone, '', '')) > 0 or

instr(v\_vPWDCheck, v\_vPatPDataWorkPhone) > 0 or

instr(pi\_vCPassword, to\_char(v\_dtPatPDataDOB, 'mmddyyyy')) > 0 or

instr(pi\_vCPassword, to\_char(v\_dtPatPDataDOB, 'ddmmyyyy')) > 0 or

instr(pi\_vCPassword, to\_char(v\_dtPatPDataDOB, 'mmddyy')) > 0 or

instr(pi\_vCPassword, to\_char(v\_dtPatPDataDOB, 'ddmmyy')) > 0 or

instr(v\_vPWDCheck, v\_vPatPDataEDIPN) > 0 then

po\_nStatusCode := 1;

po\_vStatusComment := 'Password cannot contain personal information such as names, telephone numbers, account names, birthdates, or dictionary words!';

return;

end if;

exception

when others then

null;

end;

exception

when others then

po\_nStatusCode := 1;

po\_vStatusComment := '5. Invalid User Name/Password';

return;

end;

* validate the password against rules

procedure DeleteAllSessionValues(pi\_vSessionID in varchar2,

pi\_vSessionClientIP in varchar2,

pi\_nUserID in number,

po\_nStatusCode out number,

po\_vStatusComment out varchar2) is

v\_nCount number;

begin

default status to good

po\_nStatusCode := 0; 0 = success

po\_vStatusComment := '';

v\_nCount := 0;

delete from tbicds.fx\_session where db\_session\_id = pi\_vSessionID;

commit;

delete from tbicds.fx\_session\_value

where db\_session\_id = pi\_vSessionID;

commit;

delete from tbicds.patient\_lock where session\_id = pi\_vSessionID;

commit;

exception

when others then

po\_nStatusCode := 1;

po\_vStatusComment := '158 An error occurred while removing session values, Please contact your system administrator.';

end;

* delete all session values

procedure DeleteSessionValue(pi\_vSessionID in varchar2,

pi\_vSessionClientIP in varchar2,

pi\_nUserID in number,

pi\_vKey in varchar2,

po\_nStatusCode out number,

po\_vStatusComment out varchar2) is

v\_nCount number;

begin

default status to good

po\_nStatusCode := 0; 0 = success

po\_vStatusComment := '';

v\_nCount := 0;

check that there is a session matching this session id and client ip

select count()

into v\_nCount

from tbicds.fx\_session

where db\_session\_id = pi\_vSessionID

and client\_ip = pi\_vSessionClientIP;

scrap it all, someone has a mismatched ip/sessid combo

this will affewctivly log the user out

if v\_nCount < 1 then

delete from tbicds.fx\_session where db\_session\_id = pi\_vSessionID;

commit;

delete from tbicds.fx\_session\_value

where db\_session\_id = pi\_vSessionID;

commit;

po\_nStatusCode := 1;

po\_vStatusComment := 'GetSessionValue: failed!';

return;

end if;

delete from tbicds.fx\_session\_value

where db\_session\_id = pi\_vSessionID

and client\_ip = pi\_vSessionClientIP

and upper(db\_session\_key) = upper(pi\_vKey);

commit;

exception

when others then

po\_nStatusCode := 1;

po\_vStatusComment := '159 An error occurred while removing a session value, Please contact your system administrator.';

end;

* delete 1 session value

procedure GetSessionValue(pi\_vDBSessionID in varchar2,

pi\_vWebSessionID in varchar2,

pi\_vSessionClientIP in varchar2,

pi\_nUserID in number,

pi\_vKey in varchar2,

po\_vKeyValue out varchar2,

po\_nStatusCode out number,

po\_vStatusComment out varchar2) is

v\_nCount number;

begin

default status to good

po\_nStatusCode := 0; 0 = success

po\_vStatusComment := '';

po\_vKeyValue := '';

v\_nCount := 0;

check that there is a session matching this session id and client ip

select count()

into v\_nCount

from tbicds.fx\_session

where db\_session\_id = pi\_vDBSessionID

and web\_session\_id = pi\_vWebSessionID

and client\_ip = pi\_vSessionClientIP;

scrap it all, someone has a mismatched ip/sessid combo

this will happen if the user times out for example

this will affewctivly log the user out

if v\_nCount < 1 then

delete from tbicds.fx\_session where db\_session\_id = pi\_vDBSessionID;

commit;

delete from tbicds.fx\_session\_value

where db\_session\_id = pi\_vDBSessionID;

commit;

po\_nStatusCode := 1;

po\_vStatusComment := 'GetSessionValue failed, user may have timed out!';

return;

end if;

select nvl(db\_session\_value, '')

into po\_vKeyValue

from tbicds.fx\_session\_value

where db\_session\_id = pi\_vDBSessionID

and client\_ip = pi\_vSessionClientIP

and upper(db\_session\_key) = upper(pi\_vKey);

exception

when others then

po\_nStatusCode := 1;

po\_vStatusComment := '160 An error occurred while retrieving a session value, Please contact your system administrator.';

end;

* get a session value

procedure SetSessionValue(pi\_vSessionID in varchar2,

pi\_vSessionClientIP in varchar2,

pi\_nUserID in number,

pi\_vKey in varchar2,

pi\_vKeyValue in varchar2,

po\_nStatusCode out number,

po\_vStatusComment out varchar2) is

v\_nCount number;

begin

default status to good

po\_nStatusCode := 0; 0 = success

po\_vStatusComment := '';

v\_nCount := 0;

clear any old values

delete from tbicds.fx\_session\_value

where db\_session\_id = pi\_vSessionID

and db\_session\_key = pi\_vKey;

commit;

check that there is a session matching this session iod and client ip

select count()

into v\_nCount

from tbicds.fx\_session

where db\_session\_id = pi\_vSessionID

and client\_ip = pi\_vSessionClientIP;

scrap it all, someone has a mismatched ip/sessid combo

this will affewctivly log the user out

if v\_nCount < 1 then

delete from tbicds.fx\_session where db\_session\_id = pi\_vSessionID;

commit;

delete from tbicds.fx\_session\_value

where db\_session\_id = pi\_vSessionID;

commit;

po\_nStatusCode := 1;

po\_vStatusComment := 'SetSessionValue: failed!';

return;

end if;

insert a record into fx\_session\_value

insert into tbicds.fx\_session\_value

(db\_session\_id, db\_session\_key, db\_session\_value, client\_ip)

values

(pi\_vSessionID, pi\_vKey, pi\_vKeyValue, pi\_vSessionClientIP);

commit;

exception

when others then

po\_nStatusCode := 1;

po\_vStatusComment := '161 An error occurred while setting a session value , Please contact your system administrator.';

end;

* set a session value

procedure CertLogin(pi\_vSessionID in varchar2,

pi\_vSessionClientIP in varchar2,

pi\_nUserID in number, not used but passed in to maintain consistant default param list

pi\_vCert in varchar2,

po\_nUserID out number,

po\_vDBSessionID out varchar2,

po\_nTimeout out number,

po\_nStatusCode out number,

po\_vStatusComment out varchar2) is

v\_nFXUserID number;

v\_vDBSessionID varchar2(4000);

v\_vEU varchar2(4000);

v\_vEP varchar2(4000);

begin

default status to good

po\_nStatusCode := 0; 0 = success

po\_vStatusComment := '';

po\_nUserID := 0;

v\_vDBSessionID := '';

v\_nFXUserID := 0;

v\_vDBSessionID := '';

login

begin

select nvl(t.fx\_user\_id, 0), t.user\_name, t.password

into v\_nFXUserID, v\_vEU, v\_vEP

from tbicds.fx\_user t

where certificate = pi\_vCert;

exception

when others then

v\_nFXUserID := 0;

end;

if v\_nFXUserID > 0 then

now login with this cac account

Login(pi\_vSessionID,

pi\_vSessionClientIP,

pi\_nUserID,

v\_vEU,

v\_vEP,

pi\_vCert,

po\_nUserID,

po\_vDBSessionID,

po\_nTimeout,

po\_nStatusCode,

po\_vStatusComment);

return;

else

po\_nUserID := 0;

po\_nStatusCode := 1;

po\_vStatusComment := 'Your CAC is not associated with an account!';

add an audit record

insert into tbicds.fx\_audit

(db\_session\_id,

client\_ip,

fx\_user\_id,

audit\_date,

audit\_name,

audit\_data)

values

(v\_vDBSessionID,

pi\_vSessionClientIP,

1,

sysdate,

'CACLOGIN',

'FAIL=' || pi\_vCert || po\_vStatusComment);

end if;

exception

when others then

po\_nStatusCode := 1;

po\_vStatusComment := '162 An error occurred while logging in with CAC, Please contact your system administrator.';

add an audit record

insert into tbicds.fx\_audit

(db\_session\_id,

client\_ip,

fx\_user\_id,

audit\_date,

audit\_name,

audit\_data)

values

(v\_vDBSessionID,

pi\_vSessionClientIP,

1,

sysdate,

'LOGIN',

'FAIL=' || pi\_vCert);

end;

* cac cert login

procedure ChangePassword(pi\_vSessionID in varchar2,

pi\_vSessionClientIP in varchar2,

pi\_nUserID in number, not used but passed in to maintain consistant default param list

pi\_vKey in varchar2,

pi\_vUserName in varchar2,

pi\_vOldPassword in varchar2,

pi\_vPassword in varchar2,

pi\_vCert in varchar2,

pi\_vCOldPassword in varchar2,

pi\_vCPassword in varchar2,

pi\_vCUserName in varchar2,

po\_nUserID out number,

po\_vDBSessionID out varchar2,

po\_nTimeout out number,

po\_nStatusCode out number,

po\_vStatusComment out varchar2)

* change password, this will also logg the user in

procedure Sign(pi\_vSessionID in varchar2,

pi\_vSessionClientIP in varchar2,

pi\_nUserID in number,

pi\_vUserName in varchar2,

pi\_vPassword in varchar2,

po\_vProviderID out varchar2,

po\_nUserType out number,

po\_nStatusCode out number,

po\_vStatusComment out varchar2)

* used for signing notes etc.

procedure Login(pi\_vSessionID in varchar2,

pi\_vSessionClientIP in varchar2,

pi\_nUserID in number, not used but passed in to maintain consistant default param list

pi\_vUserName in varchar2,

pi\_vPassword in varchar2,

pi\_vCert in varchar2,

po\_nUserID out number,

po\_vDBSessionID out varchar2,

po\_nTimeout out number,

po\_nStatusCode out number,

po\_vStatusComment out varchar2) is

v\_nFXUserID number;

v\_nFXUserChangePWD number;

v\_nFXUserInactive number;

v\_nFXUserLocked number;

v\_vDBSessionID varchar2(4000);

v\_nCurrAttempts number;

v\_nCount number;

v\_nCurrIPAttempts number;

v\_nIPIsLocked number;

v\_dtIPlocked date;

v\_nIPLockPeriod number;

v\_dtIPCheck date;

v\_dtExpires date;

v\_dtLastLogin date;

v\_nMAX\_ACCOUT\_LOGIN\_ATTEMPTS number;

v\_nMAX\_IP\_LOGIN\_ATTEMPTS number;

v\_nMAX\_IP\_LOGIN\_TIMEOUT number;

v\_nStatusCode number;

v\_vStatus varchar2(4000);

v\_vContent varchar2(4000);

v\_vAccountCert varchar2(4000);

begin

default status to good

po\_nStatusCode := 0; 0 = success

po\_vStatusComment := '';

po\_nUserID := 0;

v\_vDBSessionID := '';

v\_nFXUserID := 0;

v\_nFXUserChangePWD := 0;

v\_nFXUserInactive := 0;

v\_nFXUserLocked := 0;

v\_vDBSessionID := '';

v\_nCurrAttempts := 0;

v\_nCount := 0;

v\_nCurrIPAttempts := 0;

v\_nIPIsLocked := 0;

v\_nIPLockPeriod := 0;

v\_dtIPCheck := sysdate;

v\_vAccountCert := '';

v\_nMAX\_ACCOUT\_LOGIN\_ATTEMPTS := 3;

v\_nMAX\_IP\_LOGIN\_ATTEMPTS := 10;

v\_nMAX\_IP\_LOGIN\_TIMEOUT := 20;

check to see if we are IP locked

begin

is the ip locked

select nvl(t.is\_locked, 0)

into v\_nIPIsLocked

from tbicds.fx\_iplogin\_attempts t

where t.client\_ip = pi\_vSessionClientIP;

time period in mins to lock

select nvl(t.lock\_period, v\_nMAX\_IP\_LOGIN\_TIMEOUT)

into v\_nIPLockPeriod

from tbicds.fx\_iplogin\_attempts t

where t.client\_ip = pi\_vSessionClientIP;

date locked

select nvl(t.date\_last\_attempt, sysdate)

into v\_dtIPlocked

from tbicds.fx\_iplogin\_attempts t

where t.client\_ip = pi\_vSessionClientIP;

if v\_nIPIsLocked > 0 then

if the current date/time > (lock time + lock period)

v\_dtIPCheck := v\_dtIPlocked + v\_nIPLockPeriod / 1440;

if sysdate > v\_dtIPCheck then

if we are no longer ip locked clear the

fx\_iplogin\_attempts entry

delete from tbicds.fx\_iplogin\_attempts

where client\_ip = pi\_vSessionClientIP;

commit;

else

po\_nUserID := '0';

po\_nStatusCode := 6;

po\_vStatusComment := 'IP Address Locked!';

return;

end if;

end if;

exception

when others then

v\_nIPIsLocked := 0;

end;

login

begin

get the user id and status info about the account

select nvl(t.fx\_user\_id, 0),

nvl(t.reset\_password, 0),

nvl(t.is\_inactive, 0),

nvl(t.is\_locked, 0),

t.date\_password\_changed,

t.date\_last\_login,

t.certificate

into v\_nFXUserID,

v\_nFXUserChangePWD,

v\_nFXUserInactive,

v\_nFXUserLocked,

v\_dtExpires,

v\_dtLastLogin,

v\_vAccountCert

from tbicds.fx\_user t

where upper(t.user\_name) = upper(pi\_vUserName)

and t.password = pi\_vPassword;

exception

when others then

v\_nFXUserID := 0;

end;

if v\_nFXUserID > 0 then

check to see if this user is locked...

if v\_nFXUserLocked > 0 then

po\_nUserID := '0';

po\_nStatusCode := 2;

po\_vStatusComment := 'Your account is locked!';

return;

end if;

check to see if the user is inactive

if v\_nFXUserInactive > 0 then

po\_nUserID := '0';

po\_nStatusCode := 3;

po\_vStatusComment := 'Your account is inactive!';

return;

end if;

check the date\_password\_changed value and compare it

to the rules for pwd expiration...

if v\_dtExpires < trunc(sysdate) 60 then

po\_nUserID := '0';

po\_nStatusCode := 9;

po\_vStatusComment := 'Your account has expired! Please contact your System Administrator to reset your password. ';

return;

end if;

make sure the user has logged in in the last 35 days otherwise they are expired

if v\_dtLastLogin < trunc(sysdate) 35 and

v\_dtExpires < trunc(sysdate) 35 then

po\_nUserID := '0';

po\_nStatusCode := 9;

po\_vStatusComment := 'Your account has expired due to inactivity! Please contact your System Administrator to reset your password. ';

return;

end if;

make sure there is not already a cert attached to this account

if v\_vAccountCert is not null then

if v\_vAccountCert != pi\_vCert then

po\_nUserID := '0';

po\_nStatusCode := 9;

po\_vStatusComment := 'There is already a certificate associated with this account! Please contact your System Administrator to reset your password. ';

return;

end if;

end if;

update the cert so that we autologin next time

update tbicds.fx\_user

set certificate = '' pi\_vCert this app does not support cert!

where fx\_user\_id = v\_nFXUserID;

commit;

reset the fx\_user status info

update tbicds.fx\_user

set is\_locked = 0,

is\_inactive = 0,

date\_last\_login = sysdate,

flogin\_attempts = login\_attempts,

login\_attempts = 0,

LAST\_LOGIN\_IP = pi\_vSessionClientIP

where fx\_user\_id = v\_nFXUserID;

commit;

reset the ip login attempts

delete from tbicds.fx\_iplogin\_attempts

where client\_ip = pi\_vSessionClientIP;

commit;

set the timeout

po\_nTimeOut := 15;

begin

select nvl(t.session\_timeout, 15)

into po\_nTimeout

from tbicds.fx\_user t

where fx\_user\_id = v\_nFXUserID;

exception

when others then

po\_nTimeOut := 0;

end;

check to see if the user needs to change pwd

if v\_nFXUserChangePWD > 0 then

po\_nStatusCode := 4;

po\_vStatusComment := 'Please Change Password!';

return;

end if;

get a new db session id:

v\_vDBSessionID := '';

select t.SID || DBMS\_CRYPTO.RandomBytes(3) || t.SERIAL# || t.USER# ||

DBMS\_CRYPTO.RandomBytes(3) || userenv('sessionid')

into v\_vDBSessionID

from sys.v\_$session t

where t.audsid = userenv('sessionid');

clean up any old sessions, users can only login once!!!

delete from tbicds.fx\_session where fx\_user\_id = v\_nFXUserID;

commit;

delete from tbicds.fx\_session where db\_session\_id = v\_vDBSessionID;

commit;

delete from tbicds.fx\_session\_value

where db\_session\_id = v\_vDBSessionID;

commit;

insert a record into fx\_session

insert into tbicds.fx\_session

(fx\_user\_id,

web\_session\_id,

db\_session\_id,

date\_created,

date\_last\_action,

expired,

client\_ip)

values

(v\_nFXUserID,

pi\_vSessionID,

v\_vDBSessionID,

sysdate,

sysdate,

0,

pi\_vSessionClientIP);

commit;

add an audit record for the success

insert into tbicds.fx\_audit

(db\_session\_id,

client\_ip,

fx\_user\_id,

audit\_date,

audit\_name,

audit\_data)

values

(v\_vDBSessionID,

pi\_vSessionClientIP,

v\_nFXUserID,

sysdate,

'LOGIN',

'SUCCESS');

commit;

return the user id

po\_nUserID := v\_nFXUserID;

po\_vDBSessionID := v\_vDBSessionID;

else

po\_nUserID := '0';

po\_nStatusCode := 1;

po\_vStatusComment := 'Invalid user name/password';

did we fail the login but have a valid user name?

select count()

into v\_nCount

from tbicds.fx\_user t

where upper(t.user\_name) = upper(pi\_vUserName);

if v\_nCount > 0 then

get the number of current attempts

begin

select nvl(t.login\_attempts, 0)

into v\_nCurrAttempts

from tbicds.fx\_user t

where upper(t.user\_name) = upper(pi\_vUserName);

exception

when others then

v\_nCurrAttempts := 0;

end;

increment number of login attempts

v\_nCurrAttempts := v\_nCurrAttempts + 1;

update the fx\_user record

update tbicds.fx\_user

set login\_attempts = v\_nCurrAttempts,

LAST\_FLOGIN\_IP = pi\_vSessionClientIP,

LAST\_FLOGIN\_DATE = sysdate

where upper(user\_name) = upper(pi\_vUserName);

check to see if we need to lock the account

if v\_nCurrAttempts >= v\_nMAX\_ACCOUT\_LOGIN\_ATTEMPTS then

update tbicds.fx\_user

set is\_locked = 1

where upper(user\_name) = upper(pi\_vUserName);

commit;

po\_nUserID := '0';

po\_nStatusCode := 7;

po\_vStatusComment := 'Invalid user name/password, Account Locked!';

end if;

else

attempting to login with a bad user name

need to track ip address and lock that ip

for a specified period of time

v\_nCount := 0;

select count()

into v\_nCount

from tbicds.FX\_IPLOGIN\_ATTEMPTS t

where client\_ip = pi\_vSessionClientIP;

if v\_nCount > 0 then

get the number of attempts from this ip

begin

select login\_attempts

into v\_nCurrIPAttempts

from tbicds.FX\_IPLOGIN\_ATTEMPTS

where client\_ip = pi\_vSessionClientIP;

exception

when others then

v\_nCurrIPAttempts := 0;

end;

increment attempts

v\_nCurrIPAttempts := v\_nCurrIPAttempts + 1;

update last attempt and date

update tbicds.FX\_IPLOGIN\_ATTEMPTS

set date\_last\_attempt = sysdate,

login\_attempts = v\_nCurrIPAttempts

where client\_ip = pi\_vSessionClientIP;

commit;

check and lock if necessary

if v\_nCurrIPAttempts >= v\_nMAX\_IP\_LOGIN\_ATTEMPTS then

update tbicds.FX\_IPLOGIN\_ATTEMPTS

set is\_locked = 1

where client\_ip = pi\_vSessionClientIP;

commit;

po\_nUserID := '0';

po\_nStatusCode := 3;

po\_vStatusComment := 'Invalid user name/password, IP Address Locked!';

email the SA and let them know...

v\_vContent := '';

v\_vContent := 'IP Address ' || pi\_vSessionClientIP || ' ';

v\_vContent := v\_vContent ||

'attempted to login 10 times with invalid user account information! ';

end if;

else

first bad ip attempt

insert into tbicds.FX\_IPLOGIN\_ATTEMPTS

(client\_ip,

login\_attempts,

date\_last\_attempt,

is\_locked,

lock\_period)

values

(pi\_vSessionClientIP, 1, sysdate, 0, v\_nMAX\_IP\_LOGIN\_TIMEOUT);

commit;

end if;

end if;

add an audit record for the fail

insert into tbicds.fx\_audit

(db\_session\_id,

client\_ip,

fx\_user\_id,

audit\_date,

audit\_name,

audit\_data)

values

(v\_vDBSessionID,

pi\_vSessionClientIP,

1,

sysdate,

'LOGIN',

'FAIL=' || pi\_vUserName);

end if;

exception

when others then

po\_nStatusCode := 1;

po\_vStatusComment := '165 An error occurred while logging in, Please contact your system administrator.';

add an audit record

insert into tbicds.fx\_audit

(db\_session\_id,

client\_ip,

fx\_user\_id,

audit\_date,

audit\_name,

audit\_data)

values

(v\_vDBSessionID,

pi\_vSessionClientIP,

1,

sysdate,

'LOGIN',

'FAIL=' || pi\_vUserName);

end;

* login this is the version that requires a username and password after a successful login the persons CAC will be associated with this account

procedure AuditPageAccess(pi\_vSessionID in varchar2,

pi\_vSessionClientIP in varchar2,

pi\_nUserID in number,

pi\_vPageName in varchar2,

po\_nStatusCode out number,

po\_vStatusComment out varchar2)

* audit page acceess

procedure AuditTransaction(pi\_vSessionID in varchar2,

pi\_vSessionClientIP in varchar2,

pi\_nUserID in number,

pi\_vSPName in varchar2,

pi\_clAuditXML in clob,

po\_nStatusCode out number,

po\_vStatusComment out varchar2)

* audits a transaction by encrypting the clob passed in and storing the params and clob in the fx\_audit table

procedure LogOff(pi\_vSessionID in varchar2,

pi\_vSessionClientIP in varchar2,

pi\_nUserID in number,

po\_nStatusCode out number,

po\_vStatusComment out varchar2)

* Log Off, clears session from the db

procedure GetFXUserRS(pi\_vSessionID in varchar2,

pi\_vSessionClientIP in varchar2,

pi\_nUserID in number,

pi\_vEncUID in varchar2,

po\_nStatusCode out number,

po\_vStatusComment out varchar2,

rs out RetRefCursor) is

begin

po\_nStatusCode := 0; 0 = success

po\_vStatusComment := '';

open recordset

open rs for

select t.,

less than or equal to zero means we are expired!

floor(60 (trunc(sysdate) t.date\_password\_changed)) as days\_till\_expiration

from tbicds.fx\_user t

where user\_name = pi\_vEncUID;

exception

when others then

po\_nStatusCode := 1;

po\_vStatusComment := '169 An error occurred while retrieving a user record, Please contact your system administrator.';

end;

* gets an fx\_user record given the encrypted uid

procedure InsertFXUser(pi\_vSessionID in varchar2,

pi\_vSessionClientIP in varchar2,

pi\_nUserID in number,

pi\_vKey in varchar2,

pi\_vProviderID in varchar2,

pi\_vUserName in varchar2,

pi\_vPassword in varchar2,

pi\_nAccountLocked in number,

pi\_nAccountInactive in number,

pi\_vCOldPassword in varchar2,

pi\_vCPassword in varchar2,

pi\_vCUserName in varchar2,

po\_nFXUserID out number,

po\_nStatusCode out number,

po\_vStatusComment out varchar2)

* insert a record into the fx\_user table and update the fx\_user\_id in the app\_user table

procedure UpdateFXUser(pi\_vSessionID in varchar2,

pi\_vSessionClientIP in varchar2,

pi\_nUserID in number,

pi\_nFXUserID in number,

pi\_vProviderID in varchar2,

pi\_vUserName in varchar2,

pi\_vPassword in varchar2,

pi\_nAccountLocked in number,

pi\_nAccountInactive in number,

po\_nStatusCode out number,

po\_vStatusComment out varchar2)

* update an fx\_user record

procedure UpdateFXUserRights(pi\_vSessionID in varchar2,

pi\_vSessionClientIP in varchar2,

pi\_nUserID in number,

pi\_nFxUserID in number,

pi\_nUserType in number,

pi\_nUserRights in number,

pi\_nUserReadOnly in number,

po\_nStatusCode out number,

po\_vStatusComment out varchar2) is

begin

po\_nStatusCode := 0; 0 = success

po\_vStatusComment := '';

update tbicds.fx\_user\_rights

set user\_rights = pi\_nUserRights,

read\_only = pi\_nUserReadOnly,

user\_type = pi\_nUserType

where fx\_user\_id = pi\_nFxUserID;

commit;

exception

when others then

po\_nStatusCode := 1;

po\_vStatusComment := 'PCK\_FX\_SEC.UpdateSuatUserRights(): ' || sqlErrm;

end;

* update fx\_user\_rights

procedure UpdateFXUserPWD(pi\_vSessionID in varchar2,

pi\_vSessionClientIP in varchar2,

pi\_nUserID in number,

pi\_vKey in varchar2,

pi\_nFXUserID in number,

pi\_vUserName in varchar2,

pi\_vPassword in varchar2,

pi\_nAccountLocked in number,

pi\_nAccountInactive in number,

pi\_vCPassword in varchar2,

pi\_vCUserName in varchar2,

po\_nStatusCode out number,

po\_vStatusComment out varchar2)

* update an fx\_user record pwd

procedure UpdateFXUserOptions(pi\_vSessionID in varchar2,

pi\_vSessionClientIP in varchar2,

pi\_nUserID in number,

pi\_nFXUserID in number,

pi\_nAccountLocked in number,

pi\_nAccountInactive in number,

po\_nStatusCode out number,

po\_vStatusComment out varchar2)

* update an fx\_user options

procedure GetFXUsernamePasswordRS(pi\_vSessionID in varchar2,

pi\_vSessionClientIP in varchar2,

pi\_nUserID in number,

pi\_vProviderID in varchar2,

po\_nStatusCode out number,

po\_vStatusComment out varchar2,

rs out RetRefCursor) is

begin

po\_nStatusCode := 0; 0 = success

po\_vStatusComment := '';

open recordset

open rs for

select from tbicds.fx\_user t where fx\_user\_id = pi\_nUserID;

exception

when others then

po\_nStatusCode := 1;

po\_vStatusComment := '174 An error occurred while retrieving a user account, Please contact your system administrator.';

end;

* gets an Username and Password given the providerid

procedure GetFXUserIdRS(pi\_vSessionID in varchar2,

pi\_vSessionClientIP in varchar2,

pi\_nUserID in number,

pi\_vProviderID in varchar2,

po\_nStatusCode out number,

po\_vStatusComment out varchar2,

rs out RetRefCursor)

* gets an FXUserID given the providerid

procedure CheckFXUserRecRS(pi\_vSessionID in varchar2,

pi\_vSessionClientIP in varchar2,

pi\_nUserID in number,

pi\_vProviderID in varchar2,

po\_nStatusCode out number,

po\_vStatusComment out varchar2,

rs out RetRefCursor) is

begin

po\_nStatusCode := 0; 0 = success

po\_vStatusComment := '';

open recordset

open rs for

select count(f.fx\_user\_id) as FXUserCount

from tbicds.fx\_user f

where f.fx\_user\_id =

(select s.fx\_user\_id

from tbicds.app\_user s

where s.provider\_id = pi\_vProviderID);

exception

when others then

po\_nStatusCode := 1;

po\_vStatusComment := '176 An error occurred while checking a user record, Please contact your system administrator.';

end;

* gets a FXUserID given the providerid

procedure GetUserQuestions(pi\_vSessionID in varchar2,

pi\_vSessionClientIP in varchar2,

pi\_nUserID in number,

pi\_vUsername in varchar2,

po\_nStatusCode out number,

po\_vStatusComment out varchar2,

rs out RetRefCursor)

* gets user's selected security questions

procedure UpdateSecQuestions(pi\_vSessionID in varchar2,

pi\_vSessionClientIP in varchar2,

pi\_nUserID in number,

pi\_nQuestionID\_1 in number,

pi\_vAnswer\_1 in varchar2,

pi\_nQuestionID\_2 in number,

pi\_vAnswer\_2 in varchar2,

pi\_nQuestionID\_3 in number,

pi\_vAnswer\_3 in varchar2,

po\_nStatusCode out number,

po\_vStatusComment out varchar2)

* updates user's selected security questions

procedure CheckSecurityQuestions(pi\_vSessionID in varchar2,

pi\_vSessionClientIP in varchar2,

pi\_nUserID in number,

pi\_nQuestionID\_1 in number,

pi\_vAnswer\_1 in varchar2,

pi\_nQuestionID\_2 in number,

pi\_vAnswer\_2 in varchar2,

pi\_nQuestionID\_3 in number,

pi\_vAnswer\_3 in varchar2,

po\_nStatusCode out number,

po\_vStatusComment out varchar2)

* checks answers submited for the challenge questions

procedure ResetPassword(pi\_vSessionID in varchar2,

pi\_vSessionClientIP in varchar2,

pi\_nUserID in number,

pi\_vKey in varchar2,

pi\_nFXUserID in number,

pi\_vUserName in varchar2,

pi\_vPassword in varchar2,

pi\_vCPassword in varchar2,

pi\_vCUserName in varchar2,

po\_nStatusCode out number,

po\_vStatusComment out varchar2)

* reset account password

procedure GetSecurityQuestions(pi\_vSessionID in varchar2,

pi\_vSessionClientIP in varchar2,

pi\_nUserID in number,

pi\_nQuestionGrp in number,

po\_nStatusCode out number,

po\_vStatusComment out varchar2,

rs out RetRefCursor) is

begin

po\_nStatusCode := 0; 0 = success

po\_vStatusComment := '';

open recordset

open rs for

select

from stat\_sec\_questions q

where q.active = 1

and q.question\_group = pi\_nQuestionGrp;

exception

when others then

po\_nStatusCode := 1;

po\_vStatusComment := 'PCK\_FX\_SEC.GetSecurityQuestions(): ' || sqlErrm;

end;

end;

/

* Gets security questions