20170504  
ISSUE #1: the Settings.IsMultiOperator does not update after saved

private void StartSession()

{

SessionStatusMessage = Resources.SessionStartProgress;

//if multioperator, then send selected operator

//else send default operator

if (**Settings.IsMultiOperator**)//knc

{

if (SelectedOperator != null)

{

m\_controller.SessionStart(SelectedOperator);

}

}

else

{

m\_controller.SessionStart(new Business.Operator(0, "B3 Session"));

}

while (m\_controller.IsBusy == true) { }ReportsViewModel.Instance.UpdateReportUIOnStartNewSession();

}

public B3CenterSettings

{ Settings

get

{

if (m\_controller == null)

{

return null;

}

return m\_controller.Settings;

}

}

m\_controller.Settings;

private B3Controller m\_controller;

public void Initialize(B3Controller controller)

{

if (controller == null)

throw new ArgumentNullException();

m\_controller = controller;

PropertyChangedEventManager.AddListener(m\_controller, this, string.Empty); // Listen for changes to the parent and children.

m\_controller.SessionStartCompleted += OnStartDone;

m\_controller.SessionEndCompleted += OnEndDone;

m\_controller.SessionInfoCompleted += OnInfoDone;

m\_controller.SessionOperatorListCompleted += OnOperatorListDone;

m\_controller.GetSessionList();

SelectedBalls = new List<int>(m\_controller.GameBallList);

DisableB3Features();

EnableB3Features(controller.ModuleFeatureList);

if (m\_controller.Session != null && m\_controller.Session.Active)

{

SessionStatusMessage = Resources.SessionStarted;

}

else

{

SessionStatusMessage = Resources.SessionEnded;

}

}

public MainViewModel(B3CenterController controller)

{

SessionVm = SessionViewModel.Instance;

SessionVm.Initialize(controller.B3Controller);

public MainWindow(IB3CenterController parent, MainViewModel mainViewModel, bool useAcceleration)

: base(useAcceleration)

{

MainWindow = new MainWindow(this, new MainViewModel(this), Settings.UseAcceleration);

private void GetB3Settings()

{

var message = new GetB3SettingsMessage();

message.Send();

if (message.ReturnCode == ServerReturnCode.Success)

{

Settings.IsMultiOperator = message.IsMultiOperator;

Settings.IsCommonRngBallCall = message.IsCommonRng;

Settings.AllowInSessBallChange = message.AllowInSessBallChange;

Settings.EnforceMix = message.EnforceMix;

Settings.IsDoubleAccount = message.IsDoubleAccount;

Settings.B3GlobalSettings = message.B3SettingGlobal;

Settings.NorthDakotaMode = message.NorthDakotaMode;

}

else

throw new B3CenterException(string.Format(CultureInfo.CurrentCulture,

Resources.GetB3SettingsFailed, ServerErrorTranslator.GetReturnCodeMessage(message.ReturnCode)));

}

/// <summary>

/// Gets the B3 Center's settings.

/// </summary>

public B3CenterSettings Settings

{

get

{

return m\_settings;

}

private set

{

m\_settings = value;

RaisePropertyChanged("Settings");

}

}

-----------------------------------------------------------

<CheckBox Name="ChkbxMultiOperator" Content="Multi operator" Tag="52" IsChecked Binding SystemSettings.MultiOperator ="{ }"/>

->

private void UpdateSettingsListToModel(List<B3SettingGlobal> settingsList)

{

case B3SettingType.MultiOperator:

SystemSettings.MultiOperator = setting.ConvertB3StringValueToBool();

break;

->

public SystemSettingVm(List<B3SettingGlobal> systemSettingList)

{

VolumeList = Business.Helpers.ZeroToTenList;

CurrencyList = GetCurrencyList();

AutoSessionEndList = GetAutoSessionEndItemList();

SystemSettings = new SystemSetting();

UpdateSettingsListToModel(systemSettingList);

->

private SystemSettingVm InitializeSystemSettingVm()

{

var systemSettingsFromServer = m\_controller.Settings.B3GlobalSettings.Where(l => l.B3SettingCategoryType == B3SettingCategory.System && l.IsPayTableSetting == false).ToList();

var systemSettingVm = new SystemSettingVm(systemSettingsFromServer);

m\_systemSettingView = new SystemSettingView(systemSettingVm);

return systemSettingVm;

}

->

public void Initialize(B3Controller controller)

{

if (controller == null)

throw new ArgumentNullException();

m\_controller = controller;

B3IsGameEnabledSettings = new List<B3IsGameEnabledSetting>(m\_controller.Settings.B3GameSettings);

//set commands

SaveSettingcmd = new RelayCommand(parameter => RunSavedCommand());

CancelSettingcmd = new RelayCommand(parameter => CancelSetting());

LoadSettingList(controller.Settings.NorthDakotaMode);

BtnSaveIsEnabled = true;

m\_isRngBallCall = m\_controller.Settings.IsCommonRngBallCall;

}

->

public MainViewModel(B3CenterController controller)

{

case B3ModuleFeatures.B3Settings:

SettingVm = SettingViewModel.Instance;

SettingVm.Initialize(controller.B3Controller);

HasB3SettingPermission = true;

break;