Design idea:

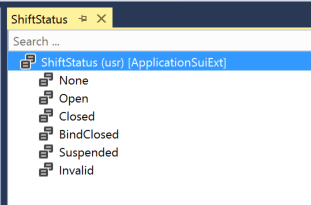
The idea of how to develop extension to show unclosed shifts information is:

Adding customization logic in CRT after open shift, suspend shift, resume shift, close shift, the customization need call real-time service to record the shift information to HQ.

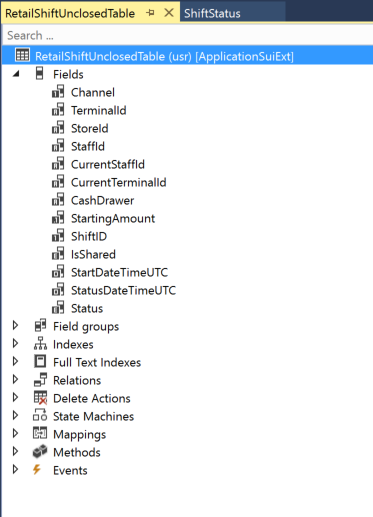
The extension development will have 2 parts, one part is in HQ, one part is in CRT.

Part I : HQ extension development

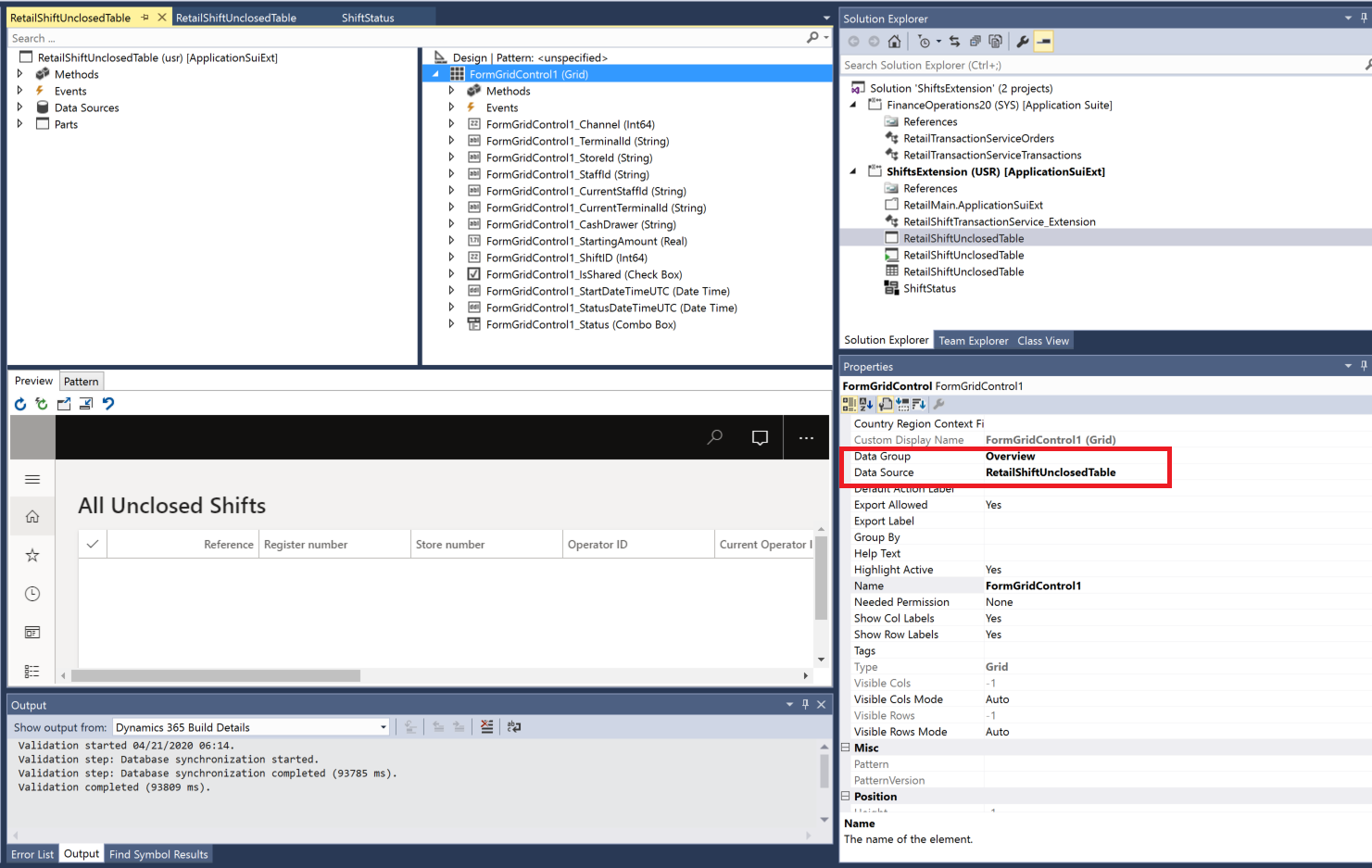
Step 1: Create a enum type to show the shift status



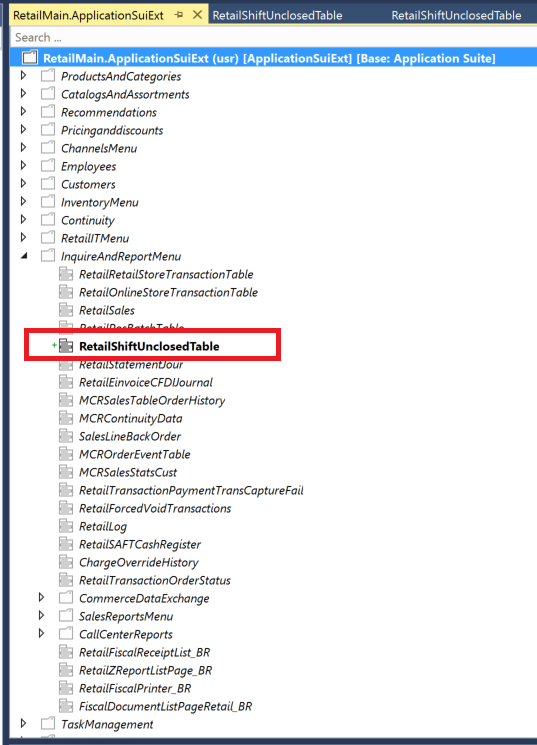
Step 2: Create a table to store the shift information



Step 3: Create a form to Display the shift information, and create a display menu item to open this form.



Step 4 add the menu item to the retail main menu:



The you should be able to open this form from menu

|  |
| --- |
|  |
|  |

Step 5, implement real-time service to receive call from CRT extension:

|  |
| --- |
| [ExtensionOf(classStr(RetailTransactionServiceEx))]  final class RetailShiftTransactionService\_Extension  {  public static container CreateNewShift(  RefRecId \_channel,  RetailTerminalId \_terminalId,  RetailStoreId \_storeId,  RetailPosBatchID \_shiftId,  RetailStaffId \_staffId,  RetailStaffId \_currentStaffId,  int \_status,  RetailTerminalId \_currentTerminalId,  NoYes \_isShared,  Name \_cashDrawer,  DataAreaId \_dataAreaId = curExt()  )  {  boolean success = true;  str error = '';  RetailShiftUnclosedTable retailShiftUnclosedTable;  select firstonly retailShiftUnclosedTable  where retailShiftUnclosedTable.Channel == \_channel &&  retailShiftUnclosedTable.TerminalId == \_terminalId &&  retailShiftUnclosedTable.ShiftID == \_shiftId;  if (!retailShiftUnclosedTable.RecId)  {  ttsbegin;  ShiftStatus \_shiftStatus = any2Enum(\_status);  retailShiftUnclosedTable.Channel = \_channel;  retailShiftUnclosedTable.TerminalId = \_terminalId;  retailShiftUnclosedTable.StoreId = \_storeId;  retailShiftUnclosedTable.ShiftID = \_shiftId;  retailShiftUnclosedTable.StaffId = \_staffId;  retailShiftUnclosedTable.CurrentStaffId = \_currentStaffId;  retailShiftUnclosedTable.Status = \_shiftStatus;  retailShiftUnclosedTable.CurrentTerminalId = \_currentTerminalId;  retailShiftUnclosedTable.IsShared = \_isShared;  retailShiftUnclosedTable.StartDateTimeUTC = DateTimeUtil::getSystemDateTime();;  retailShiftUnclosedTable.StatusDateTimeUTC = DateTimeUtil::getSystemDateTime();;  retailShiftUnclosedTable.CashDrawer = \_cashDrawer;  retailShiftUnclosedTable.insert();  ttscommit;  }  return [success, error, RetailShiftUnclosedTable.RecId];  }  public static container RemoveOpenedShift(  RefRecId \_channel,  RetailTerminalId \_terminalId,  RetailPosBatchID \_shiftId)  {  boolean success = true;  str error = "";  RetailShiftUnclosedTable retailShiftUnclosedTable;  select forupdate retailShiftUnclosedTable  where retailShiftUnclosedTable.Channel == \_channel &&  retailShiftUnclosedTable.TerminalId == \_terminalId &&  retailShiftUnclosedTable.ShiftID == \_shiftId;  if (retailShiftUnclosedTable.RecId)  {  ttsbegin;  retailShiftUnclosedTable.delete();  ttscommit;  }  return [success, error, RetailShiftUnclosedTable.RecId];  }  public static container ChangeShiftStatus(  RefRecId \_channel,  RetailTerminalId \_terminalId,  RetailPosBatchID \_shiftId,  int \_status,  RetailTerminalId \_currentTerminalId,  RetailStaffId \_currentStaffId)  {  boolean success = true;  str error = '';  RetailShiftUnclosedTable retailShiftUnclosedTable;  ShiftStatus shiftStatus = any2Enum(\_status);  select forupdate retailShiftUnclosedTable  where retailShiftUnclosedTable.Channel == \_channel &&  retailShiftUnclosedTable.TerminalId == \_terminalId &&  retailShiftUnclosedTable.ShiftID == \_shiftId;  if (retailShiftUnclosedTable.RecId)  {  ttsbegin;  retailShiftUnclosedTable.Status = shiftStatus;  retailShiftUnclosedTable.StartDateTimeUTC = DateTimeUtil::getSystemDateTime();  retailShiftUnclosedTable.CurrentTerminalId = \_currentTerminalId;  retailShiftUnclosedTable.CurrentStaffId = \_currentStaffId;  retailShiftUnclosedTable.update();  ttscommit;  }  return [success, error, RetailShiftUnclosedTable.RecId];  }  } |

Part II: CRT extension development:

Step 1: go to Retail SDK\ SampleExtensions\CommerceRuntime, find the C# solution, you can add a new C# class library project:



Step 2, create a trigger class to listen the open shift /close shift/suspend shift/close shift request/response to get the shift information, and then call real-time service to save/update/delete information to HQ, the code is as below:

|  |
| --- |
| namespace Contoso  {  namespace Commerce.Runtime.GetOpenShifts  {  using System;  using System.Collections.Generic;  using Microsoft.Dynamics.Commerce.Runtime;  using Microsoft.Dynamics.Commerce.Runtime.DataModel;  using Microsoft.Dynamics.Commerce.Runtime.Messages;  public class CreateOrUpdateShiftsTriggers : IRequestTrigger  {  public IEnumerable<Type> SupportedRequestTypes  {  get  {  return new[] {  typeof(CreateShiftRequest),  typeof(ChangeShiftStatusRequest),  typeof(ResumeShiftRequest)  };  }  }  public void OnExecuted(Request request, Response response)  {  ThrowIf.Null(request, "request");  ThrowIf.Null(response, "response");  if (request is CreateShiftRequest && response is CreateShiftResponse)  {  this.CreateShift(request as CreateShiftRequest, response as CreateShiftResponse);  }  else if (request is ChangeShiftStatusRequest && response is ChangeShiftStatusResponse)  {  Shift shift = ((ChangeShiftStatusResponse)response).Shift;  this.ChangeShift(shift, request);  }  else if (request is ResumeShiftRequest && response is ResumeShiftResponse)  {  Shift shift = ((ResumeShiftResponse)response).Shift;  this.ChangeShift(shift, request);  }  }  public void OnExecuting(Request request)  {  //Do nothing.  }  private void CreateShift(CreateShiftRequest request, CreateShiftResponse response)  {  UnclosedShiftDataServiceAsyncV3.CreateShiftAsync(request, response);  }  private void ChangeShift(Shift shift, Request request)  {  if (shift.Status == ShiftStatus.Closed)  {  UnclosedShiftDataServiceAsyncV3.CloseShiftAsync(shift, request);  }  else  {  UnclosedShiftDataServiceAsyncV3.ChangeShiftStatusAsync(shift, request);  }  }  }  }  } |

|  |
| --- |
| namespace Contoso  {  namespace Commerce.Runtime.GetOpenShifts  {  using System;  using Microsoft.Dynamics.Commerce.Runtime;  using Microsoft.Dynamics.Commerce.Runtime.DataModel;  using Microsoft.Dynamics.Commerce.Runtime.Messages;  using Microsoft.Dynamics.Commerce.Runtime.RealtimeServices.Messages;  using System.Collections.ObjectModel;  using System.Threading.Tasks;  public class UnclosedShiftDataServiceAsyncV3  {  public static void CreateShiftAsync(CreateShiftRequest request, CreateShiftResponse response)  {  ThrowIf.Null(request, "CreateShiftRequest");  ThrowIf.Null(response, "CreateShiftResponse");  string inventLocationDataAreaId = request.RequestContext.GetChannelConfiguration().InventLocationDataAreaId;  Task.Run<bool>(() => {  InvokeExtensionMethodRealtimeRequest extensionRequest = new InvokeExtensionMethodRealtimeRequest(  "CreateNewShift",  response.Shift.StoreRecordId,  response.Shift.TerminalId,  response.Shift.StoreId,  response.Shift.ShiftId,  response.Shift.StaffId,  response.Shift.CurrentStaffId,  Convert.ToInt32(response.Shift.Status),  response.Shift.CurrentTerminalId,  response.Shift.IsShared,  response.Shift.CashDrawer,  inventLocationDataAreaId);  InvokeExtensionMethodRealtimeResponse RTSResponse = request.RequestContext.Execute<InvokeExtensionMethodRealtimeResponse>(extensionRequest);  ReadOnlyCollection<object> results = RTSResponse.Result;  bool success = Convert.ToBoolean(results[0]);  return success;  });  }  public static void CloseShiftAsync(Shift shift, Request request)  {  Task.Run<bool>(() =>  {  InvokeExtensionMethodRealtimeRequest extensionRequest = new InvokeExtensionMethodRealtimeRequest(  "RemoveOpenedShift",  shift.StoreRecordId,  shift.TerminalId,  shift.ShiftId);  InvokeExtensionMethodRealtimeResponse RTSResponse = request.RequestContext.Execute<InvokeExtensionMethodRealtimeResponse>(extensionRequest);  ReadOnlyCollection<object> results = RTSResponse.Result;  bool success = Convert.ToBoolean(results[0]);  return success;  });  }  public static void ChangeShiftStatusAsync(Shift shift, Request request)  {  Task.Run<bool>(() => {  InvokeExtensionMethodRealtimeRequest extensionRequest = new InvokeExtensionMethodRealtimeRequest(  "ChangeShiftStatus",  shift.StoreRecordId,  shift.TerminalId,  shift.ShiftId,  Convert.ToInt32(shift.Status),  shift.CurrentTerminalId,  shift.CurrentStaffId);  InvokeExtensionMethodRealtimeResponse RTSResponse = request.RequestContext.Execute<InvokeExtensionMethodRealtimeResponse>(extensionRequest);  ReadOnlyCollection<object> results = RTSResponse.Result;  bool success = Convert.ToBoolean(results[0]);  return success;  });  }  }  }  } |

Part 3:

You can ask your partner developer team to review the code and update /improve the code, and then perform fully test in DEV-BOX and UAT before deploy to production, you need create retail deployable package and X++ deployable package.

But please make sure you understand that this extension is developed by myself, not by Microsoft product team, it is really out of my support scope, I just want to try my best to utilize my knowledge to unblock customer.

So neither product team nor CSS Team will provide further support in future, please make sure you do fully test before you decide to deploy it into production.