Instructions:

1. When starting a new document, modify page and document headers with the case number.

2. Name the file “Case xxx *Project Name Tech Spec*.docx” using the same project name specified in the name of the BRD and Functional Spec files for the case.

3. Store the document in SharePoint folder.

4. When two or more options are separated by a slash (“/”), delete the option(s) that don’t apply (e.g. from “Object/Object.Property/Object.Method” to “Object.Property”).

5. When updating the document, add a line to the Document History table indicating what changed.

6. When done with the first draft, delete any unused sections.

7. Delete any italicized instructions before finishing the document (e.g. “*(Repeat bulleted sections as necessary per task…)*”).

8. Always update the table of contents before finishing a document or revision (right-click on the table and select “Update Field”, then “Update Entire Table”).

9. Never estimate any task’s duration to be less than .25 hours.

10. As tasks are completed, update the “Actual” hours taken to complete the task.

11. If the design changes, update the specification to reflect those changes. If a new task must be added, set its “Estimate” to zero hours (since it wasn’t originally estimated).

12. In the “Objects” and “UI Modules” sections, group multiple objects under the same file they belong to (i.e. don’t duplicate the “VS Project Name” and “File Name” bullets for multiple objects to be modified within them). Insert a blank line to separate the tables defining the actual tasks. If there are a number of simple properties that need to be added to an object, use just one table and list all the property names in the table’s first line following the object name.

13. Delete this instruction page before distributing the document for the first time.

Case Clientelling Technical Specification

**Project Name:Clientelling**

**Client Name:**

**Affected Version(s):**

**Project Summary:**

Contents

[Document History 3](#_Toc223153803)

[Technical Specification 4](#_Toc223153804)

[Database 4](#_Toc223153805)

[New Tables 4](#_Toc223153806)

[Table Changes 4](#_Toc223153807)

[New Stored Procedures 4](#_Toc223153808)

[Stored Procedure Changes 4](#_Toc223153809)

[New Functions 4](#_Toc223153810)

[Function Changes 4](#_Toc223153811)

[New Jobs 5](#_Toc223153812)

[Job Changes 5](#_Toc223153813)

[Data Changes 5](#_Toc223153814)

[Objects 5](#_Toc223153815)

[UI Modules 5](#_Toc223153816)

[Project Workflow 6](#_Toc223153817)

[Total Estimated Hours 6](#_Toc223153818)

[Total Actual Hours 6](#_Toc223153819)

[Future Considerations 6](#_Toc223153820)

# Document History

|  |  |  |  |
| --- | --- | --- | --- |
| **Ver.** | **Date** | **Author** | **Revision** |
| 1.0 | 9/26/2016 | Leon | First draft |

# Technical Specification

This section specifies the technical design for this project.

## Database

This section specifies the changes which will be made to the database as part of this project.

### New Tables

The following tables will be added:

crt.ACTIVITYASSOCIATIONS

### Table Changes

The following table changes will be made:

| **Table Name** | **Column Name** | **Data Type** | **Default**  **Value** | **Add, Chg. or Del.?** |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

### New Stored Procedures

The following stored procedures will be added:

| **Stored Proc Name** | **Est. (Hrs.)** | **Actual**  **(Hrs.)** | **Description of New Stored Proc** |
| --- | --- | --- | --- |
|  |  |  |  |

### Stored Procedure Changes

The following stored procedure changes will be made:

| **Stored Proc Name** | **Est. (Hrs.)** | **Actual**  **(Hrs.)** | **Description of Change** |
| --- | --- | --- | --- |
|  |  |  |  |

### New Functions

The following functions will be added:

| **Function Name** | **Est. (Hrs.)** | **Actual**  **(Hrs.)** | **Description of New Function** |
| --- | --- | --- | --- |
|  |  |  |  |

### Function Changes

The following function changes will be made:

| **Function Name** | **Est. (Hrs.)** | **Actual (Hrs.)** | **Description of Change** |
| --- | --- | --- | --- |
|  |  |  |  |

### New Jobs

The following jobs will be added:

| **Job Name** | **Est. (Hrs.)** | **Actual (Hrs.)** | **Description of New Job** |
| --- | --- | --- | --- |
|  |  |  |  |

### Job Changes

The following function changes will be made:

| **Job Name** | **Est. (Hrs.)** | **Actual (Hrs.)** | **Description of Change** |
| --- | --- | --- | --- |
|  |  |  |  |

### Data Changes

The following data changes will be made:

| **Table Name** | **Affected Script(s)** | **Est. (Hrs.)** | **Actual**  **(Hrs.)** | **Description of Change** |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

## Objects

This section specifies the changes which will be made to the various classes, interfaces,XSDs, and other objects as part of this project.

*(Repeattable sections as necessary per task for a given file…)*

* **VS Project Name:** 
  + **File Name: Activity.cs**

|  |  |
| --- | --- |
| **Object/**  **Object.Property/**  **Object.Method/**  **Object.Event:** |  |
| **Action:** | New |
| **Estimate (Hours):** |  |
| **Actual (Hours):** |  |
| **Unit Test Method(s):** |  |
| **Comment:** |  |
| **Task Description:** |  |

* **VS Solution Name: CommerceRuntimeSamples**
  + **File Name: Activity.cs**

|  |  |
| --- | --- |
| **Object** | Activityy |
| **Action:** | New |
| **Estimate (Hours):** | 0.25 |
| **Actual (Hours):** | 0.5 |
| **Unit Test Method(s):** | no |
| **Comment:** | Activity entity |
| **Task Description:** | Implementing a new CRT entity and use it in new CRT service |

* + **File Name: ActivityAssociations.cs**

|  |  |
| --- | --- |
| **Object** | **ActivityAssociations** |
| **Action:** | New |
| **Estimate (Hours):** | 0.25 |
| **Actual (Hours):** | 0.5 |
| **Unit Test Method(s):** | no |
| **Comment:** | **ActivityAssociations** entity |
| **Task Description:** | Implementing a new CRT entity and use it in new CRT service |

* + **File Name: ActivityAssociationsDataService.cs**

|  |  |
| --- | --- |
| **Object** | **ActivityAssociationsDataService** |
| **Action:** | New |
| **Estimate (Hours):** | 2 |
| **Actual (Hours):** | 4 |
| **Unit Test Method(s):** | no |
| **Comment:** | **CRT service** |
| **Task Description:** | Implementing a new CRT service that handles multiple different new requests |

* + **File Name: ActivityDataService.cs**

|  |  |
| --- | --- |
| **Object** | **ActivityDataService** |
| **Action:** | New |
| **Estimate (Hours):** | 2 |
| **Actual (Hours):** | 4 |
| **Unit Test Method(s):** | no |
| **Comment:** | **CRT service** |
| **Task Description:** | Implementing a new CRT service that handles multiple different new requests |

* + **File Name: GetActivitiesDataRequest.cs**

|  |  |
| --- | --- |
| **Object** | **GetActivitiesDataRequest** |
| **Action:** | New |
| **Estimate (Hours):** | 0.3 |
| **Actual (Hours):** | 0.3 |
| **Unit Test Method(s):** | no |
| **Comment:** | to implement a new CRT service. First, new request and response classes need to be created. |
| **Task Description:** | The new Request type must implement [DataContract] and [DataMember] attributes in order for serialization to work. |

* + **File Name: GetActivitiesDataResponse.cs**

|  |  |
| --- | --- |
| **Object** | **GetActivitiesDataResponse** |
| **Action:** | New |
| **Estimate (Hours):** | 0.4 |
| **Actual (Hours):** | 0.8 |
| **Unit Test Method(s):** | no |
| **Comment:** | to implement a new CRT service. First, new request and response classes need to be created. |
| **Task Description:** | The new Request type must implement [DataContract] and [DataMember] attributes in order for serialization to work. |

* + **File Name: CloseActivityDataRequest.cs**

|  |  |
| --- | --- |
| **Object** | **CloseActivityDataRequest** |
| **Action:** | New |
| **Estimate (Hours):** | 0.3 |
| **Actual (Hours):** | 0.3 |
| **Unit Test Method(s):** | no |
| **Comment:** | to implement a new CRT service. First, new request and response classes need to becreated. |
| **Task Description:** | The new Request type must implement [DataContract] and [DataMember] attributes in order for serialization to work. |

* + **File Name: CreateActivityDataRequest.cs**

|  |  |
| --- | --- |
| **Object** | **CreateActivityDataRequest** |
| **Action:** | New |
| **Estimate (Hours):** | 0.3 |
| **Actual (Hours):** | 0.6 |
| **Unit Test Method(s):** | no |
| **Comment:** | to implement a new CRT service. First, new request and response classes need to be created. |
| **Task Description:** | The new Request type must implement [DataContract] and [DataMember] attributes in order for serialization to work. |

* + **File Name: CreateActivityDataResponse.cs**

|  |  |
| --- | --- |
| **Object** | **CreateActivityDataResponse** |
| **Action:** | New |
| **Estimate (Hours):** | 0.3 |
| **Actual (Hours):** | 0.6 |
| **Unit Test Method(s):** | no |
| **Comment:** | to implement a new CRT service. First, new request and response classes need to be created. |
| **Task Description:** | The new Request type must implement [DataContract] and [DataMember] attributes in order for serialization to work. |

* + **File Name: GetActivityAssociationsDataRequest.cs**

|  |  |
| --- | --- |
| **Object** | **GetActivityAssociationsDataRequest** |
| **Action:** | New |
| **Estimate (Hours):** | 1 |
| **Actual (Hours):** | 2 |
| **Unit Test Method(s):** | no |
| **Comment:** | to implement a new CRT service. First, new request and response classes need to be created. |
| **Task Description:** | The new Request type must implement [DataContract] and [DataMember] attributes in order for serialization to work. |

* + **File Name: GetActivityAssociationsDataResponse.cs**

|  |  |
| --- | --- |
| **Object** | **GetActivityAssociationsDataResponse** |
| **Action:** | New |
| **Estimate (Hours):** | 0.3 |
| **Actual (Hours):** | 0.3 |
| **Unit Test Method(s):** | no |
| **Comment:** | to implement a new CRT service. First, new request and response classes need to be created. |
| **Task Description:** | The new Request type must implement [DataContract] and [DataMember] attributes in order for serialization to work. |

* + **File Name: Utils.cs**

|  |  |
| --- | --- |
| **Object** | **Utils** |
| **Action:** | New |
| **Estimate (Hours):** | 1 |
| **Actual (Hours):** | 1 |
| **Unit Test Method(s):** | no |
| **Comment:** | Some help method |
| **Task Description:** | Create static method that can be reused. |

* + **File Name: App.config**

|  |  |
| --- | --- |
| **Object** | **App.config** |
| **Action:** | Change |
| **Estimate (Hours):** | 0.1 |
| **Actual (Hours):** | 0.1 |
| **Unit Test Method(s):** | no |
| **Comment:** | Fix bug |
| **Task Description:** | Change the configuration so that the CommerceRuntime Test Host can test the new CRT service. |

* + **File Name: commerceruntime.config**

|  |  |
| --- | --- |
| **Object** | **c**ommerceruntime.config |
| **Action:** | Change |
| **Estimate (Hours):** | 0.1 |
| **Actual (Hours):** | 0.1 |
| **Unit Test Method(s):** | no |
| **Comment:** | Update the commerceRuntime.Config file’s composition sectionin order to register the CRT service |
| **Task Description:** | Change the configuration so that the test program can run normally. |

* + **File Name: Program.cs**

|  |  |
| --- | --- |
| **Object** | **Program** |
| **Action:** | Change |
| **Estimate (Hours):** | 0.1 |
| **Actual (Hours):** | 0.1 |
| **Unit Test Method(s):** | no |
| **Comment:** | Create new test method to test CRT service |
| **Task Description:** | Test **ActivityDataService and ActivityAssociationsDataService** |

* **VS Solution Name: RetailServer.Extensions.StoreHoursSample**
  + **VS Project Name: RetailServer.Extensions.StoreHoursSample**
  + **File Name: ActivityAssociationsController.cs**

|  |  |
| --- | --- |
| **Object** | ActivityAssociationsController |
| **Action:** | New |
| **Estimate (Hours):** | 1 |
| **Actual (Hours):** | 1 |
| **Unit Test Method(s):** | no |
| **Comment:** | To fetch the data, A new RetailServer controller makes sense, and all the low-level work is done in the CRT (new entity, request, response, service). |
| **Task Description:** | Adding a new simple controller for entity |

* + **File Name: ActivityController.cs**

|  |  |
| --- | --- |
| **Object** | ActivityController |
| **Action:** | New |
| **Estimate (Hours):** | 1 |
| **Actual (Hours):** | 1 |
| **Unit Test Method(s):** | no |
| **Comment:** | To fetch the data, A new RetailServer controller makes sense, and all the low-level work is done in the CRT (new entity, request, response, service). |
| **Task Description:** | Adding a new simple controller for entity |

* + **File Name: CustomizedEdmModelFactory.cs**

|  |  |
| --- | --- |
| **Object** | CustomizedEdmModelFactory |
| **Action:** | New |
| **Estimate (Hours):** | 1 |
| **Actual (Hours):** | 1 |
| **Unit Test Method(s):** | no |
| **Comment:** | For any changes to the APIs to RetailServer, two steps to be done: 1. The new action or controller needs to be implemented, and 2. The model factory needs to be overridden, on order to add the new corresponding metadata. |
| **Task Description:** | overrideThe model factory |

* **VS Project Name: Proxies.RetailProxy**
  + **File Name: ActivityAssociationsManager.cs**

|  |  |
| --- | --- |
| **Object** | ActivityAssociationsManager |
| **Action:** | new |
| **Estimate (Hours):** | 1 |
| **Actual (Hours):** | 1 |
| **Unit Test Method(s):** | no |
| **Comment:** | Add new manager class in the Adapter folder and call CRT service from that manager. |
| **Task Description:** | Add ActivityAssociationsManager class in the Adapter folder and call ActivityAssociationsCRT service from the manager.. |

* + **File Name: ActivityManager.cs**

|  |  |
| --- | --- |
| **Object** | ActivityManager |
| **Action:** | new |
| **Estimate (Hours):** | 1 |
| **Actual (Hours):** | 1 |
| **Unit Test Method(s):** | no |
| **Comment:** | Add new manager class in the Adapter folder and call CRT service from that manager. |
| **Task Description:** | Add ActivityManagerclass in the Adapter folder and call ActivityCRT service from the manager.. |

* **VS Project Name: RetailServer.Extensions.TestClient**
  + **File Name: MainForm.cs**

|  |  |
| --- | --- |
| **Object** | MainForm.cs |
| **Action:** | Change |
| **Estimate (Hours):** | 1 |
| **Actual (Hours):** | 1 |
| **Unit Test Method(s):** | no |
| **Comment:** | Use RetailServer Test Client to verify customizations in the areas of Channel database, RTS, CRT, or RetailServer |
| **Task Description:** | Create test methods to verify Retailserver . |

## UI Modules

This section specifies the changes which will be made to the UI as part of this project.

*(Repeat table sections as necessary per task for a given file…)*

* **VS Project Name:**
  + **File Name:**

|  |  |
| --- | --- |
| **Object/Object.Property/Object.Method/Object.Event:** |  |
| **Action:** | New/Change/Delete |
| **Estimate (Hours):** |  |
| **Actual (Hours):** |  |
| **Comment:** |  |
| **Task Description:** |  |
| **Screen Shot:** |  |

* **VS Solution Name: CloudPos**
  + **VS Project Name: Pos.Extension.StoreHoursSample**
  + **File Name: IExtendedChannelManager.cs**

|  |  |
| --- | --- |
| **Object** | IExtendedChannelManager |
| **Action:** | Change |
| **Estimate (Hours):** | 1 |
| **Actual (Hours):** | 1 |
| **Unit Test Method(s):** | no |
| **Comment:** | To use the new retailServer API, It needs an a Manager that that extend Commerce.Model.Managers.RetailSerer.ChannelManager . |
| **Task Description:** | Adding new methods to invoke ReatilServer APIs. |

* + **VS Project Name: Pos.ViewModels**
  + **File Name: ShowJournalViewModel.ts**

|  |  |
| --- | --- |
| **Object** | **ShowJournalViewModel** |
| **Action:** | Change |
| **Estimate (Hours):** | 4 |
| **Actual (Hours):** | 4 |
| **Unit Test Method(s):** | no |
| **Comment:** | Create methods to retrieve data. |
| **Task Description:** | Adding new methods to invoke ReatilServer APIs.  1. get a page of Activity records:  public getActivities(pageSize: number, skip: number): IAsyncResult<Entities.Activity[]>  2. Get all Activities for a customer.  public getActivitiesByCustomerIdAsync(pageSize: number, skip: number): IAsyncResult<Entities.Activity[]>  3. Get all Associations for an Activity.  {IAsyncResult<Entities.ActivityAssociations[]>} The async result.  public getAssociationsByActivityId(ActivityId: string, skip: number): IAsyncResult<Entities.ActivityAssociations[]> |

* + **File Name:** CustomerActivityAddViewModel**.ts**

|  |  |
| --- | --- |
| **Object** | CustomerActivityAddViewModel |
| **Action:** | New |
| **Estimate (Hours):** | 8 |
| **Actual (Hours):** | 8 |
| **Unit Test Method(s):** | no |
| **Comment:** | Create a ViewModel. |
| **Task Description:** | Create viewMode for customer activities. It includes the following methods:  1. Validate if required fields are being put so we can save activity data  public canSaveActivity(): Boolean  2. Add a new activity to retail server  public addNewActivity(): IVoidAsyncResult  3.close selected activities  closeActivities(activities: ViewModels.ActivitiesLine[]): IVoidAsyncResult  also it defined some enum and class:  CategoryType, Priority, ActivitiesLine |

* + **File Name:** CustomerDetailsViewModel**.ts**

|  |  |
| --- | --- |
| **Object** | CustomerDetailsViewModel |
| **Action:** | Change |
| **Estimate (Hours):** | 1 |
| **Actual (Hours):** | 2 |
| **Unit Test Method(s):** | no |
| **Comment:** | Update the ViewModel. |
| **Task Description:** | Update the viewMode for customer details. It adds the following methods:  1.Gets the customer activities.  public getActivitiesByCustomer(customerAccountNumber: string, countOfActivitiesToLoad: number)  2. Populates an array of all activities.  public populateActivities(activities: Proxy.Entities.Activity[], countOfActivitiesToLoad: number)  3.Sort activities  public sortActivities() |

* + **VS Project Name: Pos.Web**
  + **File Name: resources.resjson**

|  |  |
| --- | --- |
| **Object** | **resources** |
| **Action:** | Change |
| **Estimate (Hours):** | 1 |
| **Actual (Hours):** | 1 |
| **Unit Test Method(s):** | no |
| **Comment:** | Create string resources. |
| **Task Description:** | Create string resources to implement internationalization. |

* + **VS Project Name: Pos.Web**
  + **File Name: CustomerDetailsView.html**

|  |  |
| --- | --- |
| **Object** | **CustomerDetailsView page** |
| **Action:** | Change |
| **Estimate (Hours):** | 1 |
| **Actual (Hours):** | 8 |
| **Unit Test Method(s):** | no |
| **Comment:** | Add activities area in customer details page. |
| **Task Description:** | Update the customer details page so that it add and activities area. |

* + **File Name: CustomerDetailsView.ts**

|  |  |
| --- | --- |
| **Object** | **CustomerDetailsView** |
| **Action:** | Change |
| **Estimate (Hours):** | 0.5 |
| **Actual (Hours):** | 0.5 |
| **Unit Test Method(s):** | no |
| **Comment:** | Show all customer activities. |
| **Task Description:** | Create methods to interact with user. It adds the following methods:  Refreshes activity panel with full list of activities.  public showAllActivity(): void |

* + **File Name: ShowJournalView.html**

|  |  |
| --- | --- |
| **Object** | **ShowJournalView page** |
| **Action:** | Change |
| **Estimate (Hours):** | 4 |
| **Actual (Hours):** | 8 |
| **Unit Test Method(s):** | no |
| **Comment:** | Show all customer activities. |
| **Task Description:** | Create the present layer of all customer activities. |

* + **File Name: ShowJournalView.ts**

|  |  |
| --- | --- |
| **Object** | **ShowJournalView** |
| **Action:** | Change |
| **Estimate (Hours):** | 8 |
| **Actual (Hours):** | 8 |
| **Unit Test Method(s):** | no |
| **Comment:** | Show all customer activities. |
| **Task Description:** | Create methods to interact with user. It adds the following methods:  1.activity list item selection change handler. Retrieves activity details for the customer customer details - IF - the view is not customer details mode.  public activitySelectionChangedHandler(selectedActivities: ViewModels.ActivitiesLine[]): void  2.Handler for setting activities mode.  private setCustomerActivitiesMode(customerId: string): void  3. Handler for create activity  private createActivity():void  4. Handler for close activity  private closeActivity():void  5. callback function for save customer activity  private saveCustomerActivitySuccessCallback(dataOptions: IShowJournalViewControllerOptions): void  6.callvack function for save customer ctivity when error arised.  private saveCustomerActivityErrorCallback(errors: Model.Entities.Error[]): void  7. Unwraps observable activities  private unwrapObservableActivityLines(activityLines: ViewModels.ActivitiesLine[]): ViewModels.ActivitiesLine[] |

* + **File Name: CustomerActivityAddView.html**

|  |  |
| --- | --- |
| **Object** | **CustomerActivityAddView page** |
| **Action:** | New |
| **Estimate (Hours):** | 8 |
| **Actual (Hours):** | 8 |
| **Unit Test Method(s):** | no |
| **Comment:** | Add an activity. |
| **Task Description:** | Create the present layer of new customer activity. |

* + **File Name: CustomerActivityAddView.ts**

|  |  |
| --- | --- |
| **Object** | **CustomerActivityAddView** |
| **Action:** | New |
| **Estimate (Hours):** | 8 |
| **Actual (Hours):** | 8 |
| **Unit Test Method(s):** | no |
| **Comment:** | Add a new customer activity. |
| **Task Description:** | Create methods to interact with user. It adds the following interfaces, classes and methods:  1. CustomerActivityAddViewController constructor parameters interface  interface ICustomerActivityAddViewCtorOptions  2. CustomerActivityAddViewController constructor parameters class  class CustomerActivityAddViewCtorOptions implements ICustomerActivityAddViewCtorOptions  3. Creates and instance of CustomerActivityAddViewCtorOptions class  public static CreateInstance(customer: Model.Entities.Customer, employee: Model.Entities.Employee, activity: Model.Entities.Activity, shouldSaveChanges: boolean, navigateTo: string,  destination: string, destinationOptions: any): CustomerActivityAddViewCtorOptions  4.Controller class  class CustomerActivityAddViewController extends ViewControllerBase  5. Initiate categories so the user can select the category of the new activity.  private static initCategories(): void  6. Initiate prioryties so that the user can select the priority of the new activity.  private static initPriorities(): void  7.load employees so that the user can select the activity responsible.  private loadEmployees()  8. Handler for save button click event  public saveClicked(): void  9.invoke retailServer api to persist the new actvivity  private saveCustomerActivity(dataOptions: ICustomerActivityAddViewCtorOptions): void  10. call back when new activity is saved.  private saveCustomerActivitySuccessCallback(customerActivityAddViewModel: ViewModels.CustomerActivityAddViewModel, dataOptions: Commerce.ViewControllers.IShowJournalViewControllerOptions): void  11. call back when save activity error occurs.  private saveCustomerActivityErrorCallback(errors: Model.Entities.Error[]): |

* + **Configrations**
  + **File Name: commerceruntime.config**

|  |  |
| --- | --- |
| **Object** | **commerceruntime Configration** |
| **Action:** | Change |
| **Estimate (Hours):** | 0.1 |
| **Actual (Hours):** | 0.5 |
| **Unit Test Method(s):** | no |
| **Comment:** | 1.Update the composition section with new CRT extension dll  2.copy to RetailServer\webroot\bin |
| **Task Description:** | <add source="type" value="Contoso.Commerce.Runtime.ActivityService.ActivityAssociationsDataService, OutsourceAX.Commerce.Runtime.StoreHoursSample" />  <add source="type" value="Contoso.Commerce.Runtime.ActivityService.ActivityDataService, OutsourceAX.Commerce.Runtime.StoreHoursSample" /> |

* + **File Name: commerceruntime.mposoffline.config**

|  |  |
| --- | --- |
| **Object** | **commerceruntime.mposoffline Configration** |
| **Action:** | Change |
| **Estimate (Hours):** | 0.1 |
| **Actual (Hours):** | 0.1 |
| **Unit Test Method(s):** | no |
| **Comment:** | Make sure both RetailSdk\Assets\commerceruntime.config and  RetailSdk\Assets\CommerceRuntime.MPOSOffline.config have been updated with the equivalent changes |
| **Task Description:** | <add source="type" value="Contoso.Commerce.Runtime.ActivityService.ActivityAssociationsDataService, OutsourceAX.Commerce.Runtime.StoreHoursSample" />  <add source="type" value="Contoso.Commerce.Runtime.ActivityService.ActivityDataService, OutsourceAX.Commerce.Runtime.StoreHoursSample" /> |

* + **File Name:** dllhost.exe.config

|  |  |
| --- | --- |
| **Object** | dllhost.exe.config **Configration** |
| **Action:** | Change |
| **Estimate (Hours):** | 0.1 |
| **Actual (Hours):** | 0.1 |
| **Unit Test Method(s):** | no |
| **Comment:** | Make sure dllhost.exe.config specifies the assembly and type for the RetailProxy correctly: |
| **Task Description:** | <add key="RetailProxyAssemblyName" value="Contoso.Commerce.RetailProxy"/>  <add key="AdaptorCallerFullTypeName" value ="Contoso.Commerce.RetailProxy.Adapters.AdaptorCaller"/> |

* + **File Name:** dllhost.exe.config

|  |  |
| --- | --- |
| **Object** | dllhost.exe.config **Configration** |
| **Action:** | Change |
| **Estimate (Hours):** | 0.1 |
| **Actual (Hours):** | 0.1 |
| **Unit Test Method(s):** | no |
| **Comment:** | Update Customization.settings to include this new DLL in the appropriate ISV\_\* section |
| **Task Description:** | <RetailServerLibraryPathForProxyGeneration Condition="'$(RetailServerLibraryPathForProxyGeneration)' == ''">  $(SdkReferencesPath)\OutsourceAX.RetailServer.StoreHoursSample.dll  </RetailServerLibraryPathForProxyGeneration>  <ISV\_CommerceRuntime\_CustomizableFile Include="$(SdkReferencesPath)\OutsourceAX.Commerce.Runtime.StoreHoursSample.dll" />  <ISV\_RetailServer\_CustomizableFile Include="$(SdkReferencesPath)\OutsourceAX.RetailServer.StoreHoursSample.dll" /> |

* + **File Name: Web.config**

|  |  |
| --- | --- |
| **Object** | **Web.config Configration** |
| **Action:** | Change |
| **Estimate (Hours):** | 0.1 |
| **Actual (Hours):** | 0.1 |
| **Unit Test Method(s):** | no |
| **Comment:** | Update the composition section with new RetailServer dll |
| **Task Description:** | <add source="assembly" value="OutsourceAX.RetailServer.StoreHoursSample" /> |

# Deployment Steps

|  |
| --- |
| 1. Find the physical path for RetailServer using IIS management console , Make a copy of the folder , |
| And name it as “RetailSdkRetailServer”. |
| 1. Using IIS management console, “remap” the RetailServer web application to use the RetailSdkRetailServer folder created during step 2. 2. Using IIS management console, “remap” the Cloud POS web application to use the “\RetailSDK\POS\Web” folder. 3. Open a “MSBuild Command Prompt for VS2015” window and change to the RetailSDK directory and then type msbuild /t:Rebuild  |  | | --- | | 1. Update the composition section of Web.config file at “RetailSDKRetailServer\webroot” with   this content:  <extensionComposition>  <add source="assembly" value="OutsourceAX.RetailServer.StoreHoursSample" />  </extensionComposition> | | 1. Update the composition section of commerceRuntime.config file at ” \RetailSDK\Assets” folder with this content:   <add source="type" value="Contoso.Commerce.Runtime.ActivityService.ActivityAssociationsDataService, OutsourceAX.Commerce.Runtime.StoreHoursSample" />  <add source="type" value="Contoso.Commerce.Runtime.ActivityService.ActivityDataService, OutsourceAX.Commerce.Runtime.StoreHoursSample" />  <add source="type" value="Contoso.Commerce.Runtime.ActivityService.CrossLoyaltyCardService, OutsourceAX.Commerce.Runtime.StoreHoursSample" />  <add source="type" value="Contoso.Commerce.Runtime.ActivityService.StoreHoursDataService, OutsourceAX.Commerce.Runtime.StoreHoursSample" /> |  1. Copy commerceRuntime.config updated during step 6 to “\RetailSDKRetailServer\webroot\bin” folder. |
| Project Workflow The project will be worked on in the following order: |
| Total Estimated Hours Total hours of all the above estimates:  *(Enter the total into OnTime for this incident/feature…)* Total Actual Hours Total hours spent creating technical specification:   | **Date** | **Name** | **Hours** | | --- | --- | --- | |  |  |  |   Total hours spent on all the tasks in the technical specification:  Total hours spent on system testing and rework:  Total hours spent on project: Future Considerations *(If any…)* |