

Creating MS RMS COM Component Add-Ins

Introduction:

Microsoft Retail Management System offers numerous solutions to stores (single or multiple, small or large) in the retail industry. The various functionalities include store management, purchase order management, transaction processing, point-of-sale solutions, reports and receipts processing.

Customizing RMS:

MS RMS is a highly customizable product. We can extend the functionality or modify the design of RMS products through customization. Creating customized add-in helps in extending the functionality of POS or store operations manager helping retailers in their business process. 3 types of add-ins can be created

1. COM component add-in
2. HTML add-in
3. Standard EXE add-in

In this article we will focus on creating and invoking COM component add-ins.

Creating Custom COM Component Add-In:

Before learning about COM component add-ins, let's first discuss what [QSRules] is. QSRules includes a set of classes containing the core business and database access logic of POS (Point-Of-Sale) system. It has several built-in classes, methods, properties defined such as [Session Class], [Register Class], [Transaction Class], [Tender Class] etc.

The current type of add-in is the most useful one and uses [QSRules] to interact with transaction in progress. This add-in can be activated from custom POS button or specific points in transaction flow through hooks.

COM component add-in are created as active x dlls and used to extend the functionality of POS as per customer's requirements.

Following are the steps to create COM component add-in

- First we need to create a class library project in Visual studio which should have at least one class that contains "Process" function. For the dll, the main entry point is the process function. It has the signature as follows [in VB.NET]
 - Public Function Process (currentSession As Object) As Boolean
 - The session object is passed as the parameter. When the add-in is invoked the Store operations passes a reference to current session (session class object of QSRules) to the [currentSession] object.

- The return value of this function is used by store operations to determine whether the operation is a success.
- After compiling the dll, we need to strong name the assembly and register the dll.
 - To strong name the assemblies we need navigate to the obj\debug folder of the project in Visual Studio .NET command prompt and type `sn -k ProjectName.snk`.
 - Next step is to add the line `<Assembly: AssemblyKeyFile("ProjectName.snk")>` in the assembly file (AssemblyInfo.vb) of your project. We should recompile the project after saving assembly info file.
 - To create and register type library for the dll , we need to navigate to the bin folder in the command prompt and type
 - `regasm /tlb: ProjectName.tlb ProjectName.dll`
 - `gacutil /i ProjectName.dll`
- After doing any changes in the dll code, we need to recompile and reregister the dll. Before that we need to unregister first executing the following command.
 - `gacutil /u << ProjectName >>`
 - `regasm /u ProjectName.dll`

Following is sample code for the Process function in VB.NET. When invoked this will only display the current logged in cashier's name.

[VB.NET code starts]

```
Public Function Process(ByVal currentSession As Object) As Boolean
    MsgBox("The current cashier is " & currentSession.Cashier.Name)
    'set process true
    Process = True
End Function
```

[VB.NET code ends]

Invoking Custom COM Component Add-In:

COM component dlls can be invoke in 2 ways

1. By creating custom POS buttons
2. By creating store operations hooks

Invoking through custom POS buttons:

- To create a custom POS button that invokes the add-in dll, we need to add a record to [CustomButtons] table either through Store operations or using SQL Insert query.
- Following are the parameters we need to set to create a custom POS button used to invoke the dll.

- Number - unique ID for the button (you can use count of records in this table +1)
 - Style - COM Object (Session Object) value is 7.
 - Caption – caption of POS button
 - Command – ProjectName.ClassName of the dll to be activated
 - Description – description about the custom add-in
- Now clicking on the POS button will invoke the dll process function.

Invoking through hooks:

- Hooks are basically the specific points (events) in the whole transaction process in POS. Their information is saved in registry.
- When the add-in information is added to the hooks in the registry, the add-in will be called when the specific hook is fired in the store operations.
- There are several types of hooks such as [StartPOS Hook], [InitializeTransaction Hook], [RefreshDisplay Hook] etc, which is called during transaction process.
- Each hook has a type id and a parameter number.
- The hook information are saved in registry at
 - \HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Retail Management System\Store Operations\Hooks\
 - The hooks are numbered from 000 to 029
- To invoke the active x dll from the hook, we need to first increase the count of hooks to count + 1 at above registry key's count sub key.
- Then we need to add the following information in the first available hook key.
 - HookType – type id of calling hook
 - ObjectName – ProjectName.ClassName of the dll to be activated
 - Parameter – parameters required for calling hook
 - Caption – identifying text for hook \ add-in
 - Description –description of add-in
- Following is a sample registry entry for

Name	Type	Data
(Default)	REG_SZ	(value not set)
ab]Caption	REG_SZ	QuitPOS Hook
ab]Description	REG_SZ	Created a Hook to display message on quit POS.
ab]HookType	REG_SZ	8
ab]ObjectName	REG_SZ	QuitTransactionPos.QuitTransaction
ab]Parameter	REG_SZ	1

When user hits the ESC key to quit the POS screen, this hook is called by the Store operations and the dll is invoked for to do any action defined.

Summary:

We can easily customize RMS products as per the requirement of the customer. MS RMS allows us to customize its features and extend the functionality to fulfill need of store managers or cashiers.

