Sunrise Clinical Manager™ 5.0 XA™

ObjectsPlus/XA Guide

P/N 214853 B



For further information about this manual or other Eclipsys Corporation products, contact Product Support Services, as follows:

Email: customer.support@eclipsys.com

Telephone: 888-GET-HELP (888-438-4357)

Fax: (602) 302-8555

International customers should call (602) 302-8500 for Product Support Services.

Technology Solutions Center fax: (973) 316-8587

Technology Solutions Center email: tsc.support@eclipsys.com.

Eclipsys welcomes your opinion regarding this document. Please email your comments and suggestions to techpubs@eclipsys.com or send them to the Product Support Services email address shown above.

Copyright © 2008, Eclipsys Solutions Corporation. All rights reserved.

Eclipsys is a registered trademark of Eclipsys Solutions Corporation.

All other products are trademarks or registered trademarks of their respective holders, all rights reserved. Reference to these products is not intended to imply affiliation with or sponsorship of Eclipsys Corporation.

Proprietary Notice

This document is an unpublished work containing confidential and trade secret information, which is proprietary to Eclipsys Solutions Corporation ("Eclipsys"), and is protected by laws pertaining to such materials. This document, the information in this document, and all rights thereto are the sole and exclusive property of Eclipsys, are intended for use by customers and employees of Eclipsys, and are not to be copied, used, or disclosed to anyone else, in whole or in part, without the express written permission of Eclipsys. For authorization from Eclipsys to copy this information, please call Eclipsys Product Support Services at 888 GET-HELP or 888 438-4357.

Table of Contents

Chapter 1: About This Guide 1
1.1 Overview 1
1.2 Audience 1
1.3 What's New in This Document 2
1.4 Accessing Eclipsys Application Online Help 2
1.5 Contact Us 2
1.6 Using Adobe® Reader® 2

Chapter 2: What is ObjectsPlus/XA? 5
2.1 Overview 5
2.2 Context Sharing 6
2.3 Objects 6
2.4 MSDN Help File 7

Chapter 3: Creating Tabs 9
3.1 Overview 9
3.2 Development Tools 10
3.3 Custom Tab Interfaces 10
3.3.1 ICustomTab 10
3.4 Creating a Custom Tab UserControl 11
3.5 Building a Custom Tab Library 14
3.6 Configuring the Custom Tab 15
3.7 Getting and Setting Context 18
3.7.1 Retrieving Current Patient Information 18
3.8 Event Notifications 20
3.9 Tab Security 22

Chapter 4: Creating Tools 23
4.1 Overview 23
4.2 Development Tools 24
4.3 Calling a Custom Application Using the Tools Menu 24
4.3.1 Defining Tools Menu Items 25
4.3.2 Defining Tools Menu Items as Custom Applications 26
4.4 Calling a Custom Application from Order Entry 27
4.5 Custom Tools Application Interface 28
4.5.1 ICustomApp 28
4.6 Building a Custom Application 29
4.7 Using The ICustomApp Interface 31
4.8 CLRApp Attribute 31
4.9 ICustomFormApp Interface 32
4.10 Receiving Context Change Notifications 33
4.11 Receiving Context Change Notifications – Modeless Window Support 35

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

4.12 Accessing Form Context Information 35

Chapter 5: Calling Custom DLLs From Forms 37

5.1 Overview 37
5.2 Development Tools 38
5.3 Custom Form Applications Interface 39
5.3.1 ICustomFormApp 39
5.3.2 Form Context Information 39
5.4 Integrating Custom Form Applications 41
5.4.1 Writing a Hello World Application 42
5.5 Using the CommunicationObj Class 43
5.6 Using the ClientInfoObj Class 43
5.7 Using the FormObj Class 43
5.7.1 Accessing All Fields on a Form 44
5.7.2 Finding a Specific Form Field 45
5.7.3 Changing Field Properties 46
5.7.4 Changing Field Values 47
5.7.5 Type-Safe Field Access 48
5.8 Class Information 48
5.8.1 CommunicationObj Class 48
5.8.2 PrimaryOrderObj Class 52
5.8.3 PrimaryOrderTaskOccurrenceObj Class 54
5.8.4 PrimaryOrderSetObj Class 55
5.8.5 FormObj Class 56
5.8.6 FormFieldObj Class 56
5.8.7 FormFieldCollection Class 56
5.9 Field Types 57
5.9.1 AdditionalVerificationByFieldObj Class 58
5.9.2 AdminInstructionsFieldObj Class 58
5.9.3 BaseSolutionFieldObj Class 58
5.9.4 CalculatedDoseFieldObj Class 58
5.9.5 CollapsibleWarningFieldObj Class 59
5.9.6 CombinedMeasurementsFieldObj Class 59
5.9.7 ConditionalOrderFieldObj Class 59
5.9.8 CreatinineClearanceFieldObj Class 59
5.9.9 DosageLowFieldObj Class 60
5.9.10 FrequencyCodeFieldObj Class 60
5.9.11 IsNoTaskFieldObj Class 61
5.9.12 IsPRNFieldObj Class 61
5.9.13 IVAdditiveCalcFieldObj Class 61
5.9.14 IVAdditiveFieldObj Class 61
5.9.15 MultiOrderCheckboxFieldObj Class 61
5.9.16 MultiOrderGridFieldObj Class 62
5.9.17 MultiOrderInlineFieldObj Class 62

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

5.9.18 OrderIVRateFieldObj Class . . . 62
5.9.19 RequestedTimeFieldObj Class . . . 62
5.9.20 SpecialInstructionsFieldObj Class . . . 62
5.9.21 StopAfterFieldObj Class . . . 63
5.9.22 StopTimeFieldObj Class . . . 63
5.9.23 VariableDoseFieldObj Class . . . 63
5.10 Field Value Types . . . 63
5.10.1 AdditionalVerificationByValueObj Class . . . 63
5.10.2 BaseSolutionValueObj Class . . . 64
5.10.3 CalculatedDoseValueObj Class . . . 64
5.10.4 CollapsibleWarningValueObj Class . . . 64
5.10.5 CombinedMeasurementsValueObj Class . . . 65
5.10.6 ComplexDosingGridValueObj Class . . . 65
5.10.7 ComplexDosingGridColListObj Class . . . 65
5.10.8 ComplexDosingGridColValueObj Class . . . 66
5.10.9 ComplexDosingGridRowListObj Class . . . 66
5.10.10 ComplexDosingGridRowValueObj Class . . . 68
5.10.11 ComplexDosingGridCellListObj Class . . . 69
5.10.12 ComplexDosingGridCellValueObj Class . . . 70
5.10.13 ConditionalOrderValueObj Class . . . 72
5.10.14 CreatinineClearanceValueObj Class . . . 72
5.10.15 EnhancedTimeValueObj Class . . . 72
5.10.16 FrequencyCodeValueObj Class . . . 73
5.10.17 IVAdditiveCalcListObj Class . . . 73
5.10.18 IVAdditiveCalcValueObj Class . . . 73
5.10.19 IVAdditiveListObj Class . . . 73
5.10.20 IVAdditiveValueObj Class . . . 74
5.10.21 IVRateValueObj Class . . . 75
5.10.22 MultiOrderItemCollection Class . . . 75
5.10.23 MultiOrderItemValueObj Class . . . 76
5.10.24 MultipleFrequencyGridValueObj class . . . 76
5.10.25 MultipleFrequencyGridRowListObj Class . . . 77
5.10.26 MultipleFrequencyGridRowValueObj Class . . . 78
5.10.27 StopAfterValueObj Class . . . 80
5.10.28 VariableDoseListObj Class . . . 80
5.10.29 VariableDoseValueObj Class . . . 80
5.10.30 ClientInfoObj Class . . . 80
5.11 External Form Processing API . . . 82
5.11.1 Integrating in a Called Program . . . 82
5.11.2 IExternalFormProcessing Interface . . . 83
5.11.3 IExternalFormProcessing.Process Method . . . 83
5.11.4 IExtFormSupport Interface . . . 85
5.11.5 IExtFormSupport.GetValue Method . . . 86

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

5.11.6 IExtFormSupport.SetValue Method . . . 87
5.11.7 IExtFormSupport.IsMandatory Method . . . 88
5.11.8 IExtFormSupport.IsVisible Method . . . 89
5.11.9 IExtFormSupport.IsReadOnly Method . . . 90
5.11.10 IExtFormSupport.GetLabel Method . . . 91

Chapter 6: Extending the Notification Server . . . 93
6.1 Overview . . . 93
6.2 Writing Transports . . . 93
6.2.1 Integration . . . 93
6.2.2 Interface Details . . . 94

Chapter 7: SCMLib Objects . . . 103
7.1 Overview . . . 103
7.2 HVCLogonObj Class . . . 103
7.2.1 Namespace . . . 103
7.2.2 HVCLogonObj.GetConnectionString() . . . 104
7.2.3 HVCLogonObj.GetCurrentUser() . . . 104
7.2.4 HVCLogonObj.GetCurrentUserName() . . . 105
7.2.5 HVCLogonObj.GetSessionID() . . . 105
7.2.6 HVCLogonObj.GetEnvironmentProfileString() . . . 105
7.2.7 HVCLogonObj.GetEnvironmentProfileInt() . . . 106
7.2.8 HVCLogonObj.GetEnvWorkstationProfileString() . . . 106
7.2.9 HVCLogonObj.GetEnvWorkstationProfileInt() . . . 107
7.2.10 HVCLogonObj.GetBuild() . . . 107
7.2.11 HVCLogonObj.GetWorkstationGUID() . . . 107
7.2.12 HVCLogonObj.GetWorkstationIDCode() . . . 108
7.2.13 HVCLogonObj.CheckRightsByID() . . . 108
7.2.14 HVCLogonObj.GetApplicationName() . . . 109
7.2.15 HVCLogonObj.GetColorPreference() . . . 109
7.2.16 HVCLogonObj.GetDateDisplayFormat() . . . 111
7.2.17 HVCLogonObj.GetDateEntryFormat() . . . 111
7.2.18 HVCLogonObj.GetTimeDisplayFormat() . . . 111
7.3 MainFrameObj Class . . . 112
7.3.1 Namespace . . . 112
7.3.2 MainFrameObj.GetVersion() . . . 112
7.3.3 MainFrameObj.GetClientVersion() . . . 112
7.3.4 MainFrameObj.GetDatabaseVersion() . . . 113
7.3.5 MainFrameObj.GetCurrentDatabase() . . . 113
7.3.6 MainFrameObj.GetOperatingSystem() . . . 114
7.3.7 MainFrameObj.Handle . . . 114
7.3.8 MainFrameObj.GetApplicationMode() . . . 114
7.4 CustomContextObj Class . . . 115
7.4.1 Namespace . . . 115

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7.4.2 CustomContextObj.GetInstance() . . . 115
7.4.3 CustomContextObj.SetItem() . . . 115
7.4.4 CustomContextObj.GetItem() . . . 116
7.4.5 CustomContextObj.RemoveItem() . . . 116
7.4.6 CustomContextObj.ClientGUID . . . 116
7.4.7 CustomContextObj.ChartGUID . . . 117
7.4.8 CustomContextObj.VisitGUID . . . 117
7.4.9 CustomContextObj.UserGUID . . . 118
7.4.10 CustomContextObj.GetConnectionString() . . . 118
7.5 HVCDateObj Class . . . 118
7.5.1 Namespace . . . 118
7.5.2 HVCDateObj.ServerToday() . . . 119
7.6 PObj Class . . . 119
7.6.1 Namespace . . . 119
7.6.2 PObj.GUID . . . 119
7.6.3 PObj.Dispose . . . 120
7.7 UserPObj Class . . . 120
7.7.1 Namespace . . . 120
7.7.2 UserPObj Constructor . . . 120
7.7.3 UserPObj.DisplayName . . . 121
7.7.4 UserPObj.FirstName . . . 121
7.7.5 UserPObj.LastName . . . 121
7.7.6 UserPObj.MiddleName . . . 122
7.7.7 UserPObj.Title . . . 122
7.7.8 UserPObj.BirthDayNum . . . 123
7.7.9 UserPObj.BirthMonthNum . . . 123
7.7.10 UserPObj.BirthYearNum . . . 123
7.7.11 UserPObj.OccupationCode . . . 124
7.7.12 UserPObj.IDCode . . . 124
7.7.13 UserPObj.OrderRoleType . . . 125
7.8 ClientPObj Class . . . 125
7.8.1 Namespace . . . 125
7.8.2 ClientPObj Constructor . . . 125
7.8.3 ClientPObj.DisplayName . . . 126
7.8.4 ClientPObj.FirstName . . . 126
7.8.5 ClientPObj.LastName . . . 126
7.8.6 ClientPObj.MiddleName . . . 127
7.8.7 ClientPObj.Title . . . 127
7.8.8 ClientPObj.BirthDayNum . . . 128
7.8.9 ClientPObj.BirthMonthNum . . . 128
7.8.10 ClientPObj.BirthYearNum . . . 128
7.8.11 ClientPObj.ReligionCode . . . 129
7.8.12 ClientPObj.LanguageCode . . . 129

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7.8.13 ClientPObj.GenderCode . . . 130
7.8.14 ClientPObj.RaceCode . . . 130
7.8.15 ClientPObj.MaritalStatusCode . . . 130
7.8.16 ClientPObj.IDCode . . . 131
7.8.17 ClientPObj.IsTemporary . . . 131
7.8.18 ClientPObj.ApplicSource . . . 132
7.8.19 ClientPObj.CityOfBirth . . . 132
7.8.20 ClientPObj.CountryDvsnOfBirth . . . 132
7.8.21 ClientPObj.CountryOfBirth . . . 133
7.8.22 ClientPObj.IsVIP . . . 133
7.8.23 ClientPObj.PrivacyStatusCode . . . 134
7.8.24 ClientPObj.SecurityLevel . . . 134
7.9 ChartPObj Class . . . 134
7.9.1 Namespace . . . 134
7.9.2 ChartPObj Constructor . . . 135
7.10 ClientVisitPObj Class . . . 135
7.10.1 Namespace . . . 135
7.10.2 ClientVisitPObj Constructor . . . 135
7.10.3 ClientVisitPObj.ClientDisplayName . . . 136
7.10.4 ClientVisitPObj.VisitIDCode . . . 136
7.10.5 ClientVisitPObj.TypeCode . . . 136
7.10.6 ClientVisitPObj.CareLevelCode . . . 137
7.10.7 ClientVisitPObj.CurrentLocation . . . 137

Chapter 8: ObjectsPlus/XA Framework . . . 139
8.1 Overview . . . 139
8.2 Prerequisites . . . 140
8.3 Creating .NET Custom Applications . . . 141
8.3.1 Common Setup . . . 141
8.3.2 Using ObjectsPlus/XA Framework in Custom Tabs . . . 143
8.3.3 Creating a C# Custom Tab Project . . . 143
8.3.4 Creating a Visual Basic Custom Tab Project . . . 149
8.3.5 Using ObjectsPlus/XA Framework in Standalone Applications . . . 156
8.3.6 Creating a C# Standalone Application . . . 156
8.3.7 Creating a VB Standalone Application . . . 158
8.3.8 Best Practices . . . 160
8.4 Using ObjectsPlus/XA Framework in MLMs . . . 161
8.4.1 Key Differences . . . 162
8.4.2 MLM Examples . . . 163
8.4.3 Using Test MLMs as a Starting Point . . . 167
8.4.4 Best Practices . . . 167

Index . . . 169

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

CHAPTER 1: ABOUT THIS GUIDE

Contents

1.1 Overview 1
1.2 Audience 1
1.3 What's New in This Document 2
1.4 Accessing Eclipsys Application Online Help. 2
1.5 Contact Us 2
1.6 Using Adobe® Reader® 2

1.1 Overview

The ObjectsPlus Guide gives a detailed overview of how to use core objects within the Eclipsys systems and applications. This guide describes how a developer can use the ObjectsPlus/XA Framework to create Custom Tabs, Custom Applications and Custom Form Applications.

Note: Some graphics of application windows and the names of some applications described in this guide may be slightly different than those shown on your system due to the highly configurable nature of Eclipsys software. These graphics are for illustration purposes only and these variances do not affect the functionality of the application. The software described in this guide is based on a core or model product and may not fully represent customer modifications.

1.2 Audience

The primary audience for this guide is Developers or an Eclipsys Implementation team.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

1.3 What's New in This Document

The following table defines changes to the document for the current release.

Table 1.1: What's New

Chapter	Description
Chapter 3, 4, and 5	Parts of these chapters were taken from earlier versions of the Technical Reference guide. They were Custom .NET Tabs, Custom Applications and Custom Form Applications.
Chapter 7	This chapter was taken from an earlier version of the Technical Reference guide. Existing 4.5 Objects are listed for Custom Tabs, Custom Applications and Custom Form Applications.
Chapter 8 (new chapter)	This new chapter contains 5.0 Object information for Custom Tabs, Custom Applications and Standalone Applications.

1.4 Accessing Eclipsys Application Online Help

Eclipsys applications have online help systems that you can view to learn more about the part of the application you are using.

◆ To access online help

- On most windows, press F1.
- Click Help buttons in the applications to access context-sensitive help.
- Select Help from the toolbar. You can launch the main help, search the help, or view a page that identifies the current version of the application you are running.

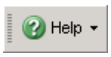
1.5 Contact Us

- If you have a problem with an Eclipsys product and need help, call Product Support at 1-888-GET-HELP or send an email to customersupport@eclipsys.com.
- If you have feedback about the documentation, contact us at userdoc@eclipsys.com.

1.6 Using Adobe® Reader®

This document is provided in PDF format, which you view using Adobe® Reader® 7.0.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

If you require help using Adobe Reader, press F1 or click  on the toolbar.

To rotate a page, select View > Rotate View > Clockwise or Counterclockwise.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

CHAPTER 2: WHAT IS OBJECTSPLUS/XA?

Contents

2.1 Overview 5
2.2 Context Sharing 6
2.3 Objects 6
2.4 MSDN Help File 7

2.1 Overview

ObjectsPlus/XA allows clients to easily develop applications hosted within Sunrise Clinical Manager (SCM) and other Eclipsys solutions on the XA platform. ObjectsPlus/XA exposes parts of the application to allow IT staff to build components that access patient medical records, or extend solutions with data integrated from other systems.

ObjectsPlus/XA gives an opportunity to use existing non-Eclipsys applications during a transition to Eclipsys solutions, allowing customers to meet site-specific requirements, such as the addition of Enterprise Defined Patient List Columns to support other customer applications. These columns are added using the ObjectsPlus/XA Framework via a Custom Tab, where previously an interface or dialog needed to be used.

ObjectsPlus/XA contains a set of components that allow for the structured development of extensions to Eclipsys systems and applications.

Through ObjectsPlus/XA customers access custom assemblies from within SCM and in some cases have SCM interact with these assemblies. These examples list some of the tasks possible using ObjectsPlus/XA:

- Call .NET assemblies from MLMs.
- Call a .NET assembly from the Call Program button on the order form.
- Create a tab for addition to SCM.
- Create an assembly or program accessible through a menu item under tools.
- Create a custom transport for the Notification Server.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

The custom assemblies can retrieve information from SCM including context type information. The currently logged in SCM user and the currently selected visit are considered context information. The context information is available through context sharing, described under "Context Sharing" on page 6.

These areas are described in more detail later in this guide. For more information see"Creating Tabs" on page 9, "Creating Tools" on page 23, "Calling Custom DLLs From Forms" on page 37, or "Extending the Notification Server" on page 93

2.2 Context Sharing

Context sharing allows custom assemblies to know which user is logged into SCM and which visit the user currently has selected. SCM can tell the assembly when the visit changes.

For more information, see "Getting and Setting Context" on page 18.

2.3 Objects

Table 2.2 describes some of the more significant objects currently available.

Table 2.2: Available Objects

Object	Description
Allergy	This object allows allergies to be added, modified, discontinued, or deleted.
Care Provider Visit Role	This object enables a care provider visit role to be added, discontinued, or deleted.
Comment	This object allows the user to add a comment or modify, discontinue, or delete an existing comment.
Enterprise-defined Patient List Column Value	This object supports setting the value of an enterprise defined patient list column.
Health Issue	This object adds a health issue as well as modify, discontinue, or delete an existing health issue.
Order	This object allows new orders to be created. Orders are limited in that only single orders can be created, and existing orders cannot be modified or updated.
Significant Event	This object allows adding, modifying, discontinuing, or deleting a significant event.
User	This object provides access to information for the logged on user.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

The Allergy, Care Provider Visit Role, Comment, Enterprise-defined Patient List Column Value, Health Issue, Order, and Significant Event are new ObjectsPlus/XA objects.

2.4 MSDN Help File

In addition to the documentation found in this guide, an MSDN Help file is also included for reference purposes. It lists the new ObjectsPlus/XA Framework namespaces with all associated objects, methods and interfaces.

The MSDN Help file includes these namespaces:

- ObjectsPlusXA.SunriseClinicalManager
- ObjectsPlusXA.SunriseClinicalManager.Common
- ObjectsPlusXA.SunriseClinicalManager.Forms

This Help file is accessed using a separate install kit available to the customer.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

CHAPTER 3: CREATING TABS

Contents

3.1 Overview 9
3.2 Development Tools 10
3.3 Custom Tab Interfaces 10
3.4 Creating a Custom Tab UserControl 11
3.5 Building a Custom Tab Library 14
3.6 Configuring the Custom Tab 15
3.7 Getting and Setting Context 18
3.8 Event Notifications 20
3.9 Tab Security 22

3.1 Overview

It is possible to add one or more Custom Tabs, known to the SCM user as chart sections, to the standard SCM user interface. Each of these tabs behaves in a manner similar to the standard tabs, has access to patient, and user context information, and event notifications.

In general, use Custom Tabs to show:

- SCM data in a custom format.
- Information from a third-party application through either:
  - A third-party's application programming interface (API).
  - Queries to the third-party's database.

In earlier releases, the user could host either an ActiveX control or a .NET UserControl within a Custom Tab. For backward-compatibility, the BasicTab.dll library continues to host ActiveX controls. Its feature set is essentially unchanged from previous versions, and is described in the *Technical Reference guide* under "Creating an ActiveX Custom Tab."

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

.NET Custom tabs are created using the BasicCLRTab.dll library. The UserControl is developed in a language, such as C# or Visual Basic (VB), that targets the Common Language Runtime (CLR). Using BasicCLRTab is the preferred method for adding Custom Tabs, using it gives access to a complete set of exposed context information and events. In addition, the .NET object library (SCMLib.dll) is used by SCM services and windows.

3.2 Development Tools

The 5.0 version of SCM has been developed using the following tools:

- Visual Studio 2005 with Service Pack 1.
- Infragistics NetAdvantage for .NET 2006 Volume 2 with Hotfix 1079.

It is recommended that any custom assemblies developed for prior versions of SCM are upgraded to use the same development tools. Using the same development tools reduces the overall memory footprint.

3.3 Custom Tab Interfaces

Two interfaces must be implemented for a Custom Tab to function. The ICustomTab interface and the IPageView interface. These two interfaces have methods that must be present in the code, the ICustomTab interface has changeable methods, while the IPageView methods are stubbed and not changed.

3.3.1 ICustomTab

This interface is required to synchronize DLLs with events occurring in SCM. It has one method that must be implemented allowing the DLL to hook up to some SCM events.

```
public void InitEvents(CustomTabEventSource eventSource)
```

This method is used in Figure 3.1 on page 14.

After a Custom Tab is created, it knows when certain events in SCM occur. Table 3.1 describes some of these events.

Table 3.1: ICustomTab Events (1 of 2)

Event	Description
OnPatientChange	In this event the majority of code is placed, along with the code to be executed when the user selects a new patient in SCM.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

Table 3.1: ICustomTab Events (2 of 2)

Event	Description
OnUserChange	In this event, place the code to be executed when a new user logs on to SCM. Put code here if user information is cached. Otherwise if one user logs off, the DLL does not realize there is a new user if first user logged off only to the logon screen.
OnCanChangePatient	This event returns TRUE unless a valid reason is given for preventing patient changes in SCM.
OnTabEnter	This event is rarely used, but if code needs to run when a user selects the Custom Tab, the code is placed in this event.
OnTabLeave	This event is rarely used, but if code needs to run when a user navigates away from the Custom Tab, the code is placed in this event.
OnApplicationTimedOut	If a third-party application is attached to the tab, code is placed in this event to indicate the action occurring when SCM times out.

3.4 Creating a Custom Tab UserControl

BasicCLRTab can host any UserControl that meets these minimum requirements:

- The control class implements the IPageView interface. This interface can be used by SCM to communicate with the UserControl.
- The control class caches the IPageViewSink interface pointer passed to it and correctly manages its reference count. This interface can be used to communicate with SCM.

These two business rules are identical to those for hosting an ActiveX control in the COM-based BasicTab.dll Custom Tabs. While BasicTab must rely solely on IPageView and IPageViewSink for bidirectional communication, BasicCLRTab provides access to additional interfaces and components.

◆ To implement the IPageView interface:

1 The Custom Tab project must first set a reference to the BASICTABLib.dll assembly that is shipped with SCM. It provides the definition of both IPageView and IPageViewSink.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

2 Declare the BASICTABLib namespace in the UserControl source file. The Figure 3.1 C# code example shows a simple UserControl class named MyFirstCustomTab that implements the IPageView interface. It also declares a private m_MySink data member that is of type IPageViewSink.

The AdviseSink() and UnadviseSink() IPageView methods are used for handshaking between SCM and the control.

3 AdviseSink() is called soon after the UserControl is instantiated and an IPageViewSink pointer is passed. This pointer is typically cached by the control for subsequent use. In the example, this value is assigned to m_MySink.

4 UnadviseSink() is called when SCM is about to shut down. This is necessary to release the IPageViewSink reference here. Releasing the IPageViewSink reference is a very important step, because you can inadvertently prevent SCM from shutting down completely. The main reason is that IPageViewSink is a COM interface pointer internal to SCM, and requires the necessary AddRef/Release calls to maintain its reference count.

```
using BASICTABLib;
using SCMLib;
using SCMLib.CustomTab;
using SCMLib.Context;

namespace MyCustomTabs

{
/// <summary>
/// Summary description for MyFirstCustomTab.
/// </summary>
public class MyFirstCustomTab : UserControl, IPageView,
ICustomTab
{

 Private IPageViewSink m_MySink = null;

 #region IPageView Members

        public void SetFocusLast()
  {
  }

public void FireActionMenuItem(int MenuID)
 {
 }
  public void UnadviseSink(int ConnectId)
  {
  // Must release the sink pointer
  Marshal.ReleaseComObject(m_MySink);
  }

        public void OnChangeVisit()
  {
  }
```


This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

```
        public void CanChangePatient()
  {
  }

        public int AdviseSink(IPageViewSink PageViewSink)
  {
       // Cache the sink pointer
       m_MySink = PageViewSink;

            return 0;
  }

        public void SetFocusFirst()
  {
  }
public void IsFirstWnd()
  {
  }

        public string[] GetActionMenu()
  {
       return null;
  }

        public void IsLastWnd()
  {
  }

        public void CanCloseTab()
  {
  }

        #endregion

#region ICustomTab Members

        public void InitEvents(CustomTabEventSource eventSource)
        {
            // We want to receive patient change notifications
            eventSource.OnPatientChange += new
SCMPatientChangeHandler(eventSource_OnPatientChange);

            // We want to receive can patient change notifications
            eventSource.OnCanChangePatient += new
SCMCanChangePatientHandler(eventSource_OnCanChangePatient);

            // We want to receive user change notifications
            eventSource.OnUserChange += new
SCMUserChangeHandler(eventSource_OnUserChange);

            // We want to receive menu notifications
            eventSource.OnSCMCommand += new
SCMCommandHandler(eventSource_OnSCMCommand);
            eventSource.OnSCMCommandUI += new
SCMCommandUIHandler(eventSource_OnSCMCommandUI);

            // We want to receive application timed out
notifications
            CustomContextObj cc = CustomContextObj.GetInstance();
```


This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

```
            cc.CustomAppEventSource.OnApplicationTimedOut += new
SCMLib.CustomApp.SCMApplicationTimedOutHandler(CustomAppEventSourc
e_OnApplicationTimedOut);
        }

        #endregion

        private void eventSource_OnPatientChange()
        {
           // The patient has changed, so obtain the latest patient
context
            CustomContextObj cc = CustomContextObj.GetInstance();
            string sClientGUID = cc.ClientGUID;
            string sChartGUID = cc.ChartGUID;
            string sVisitGUID = cc.VisitGUID;

            // Refresh my data based on the patient context
        }

        private void eventSource_OnUserChange()
        {
            // The user has changed, so obtain the latest user
context
            CustomContextObj cc = CustomContextObj.GetInstance();
            string sUserGUID = cc.UserGUID;

            // Refresh my data based on the user context
        }

        private bool eventSource_OnCanChangePatient()
        {
            // Check for unsaved data

            // Must return true unless there is a good
            // reason for stopping the pending patient change
            return true;
        }

}

}
```

Figure 3.1: Custom Tab UserControl

The MyFirstCustomTab UserControl is fully eligible to be hosted in an SCM Custom Tab. The next two sections describe the configuration steps required to make the tab appear.

3.5 Building a Custom Tab Library

Create a separate .NET class library project for each of the Custom Tabs. For example, the MyFirstCustomTab UserControl class is in a generated assembly named MyFirstCustomTab.dll, while the MySecondCustomTab UserControl resides in MySecondCustomTab.dll.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

BasicCLRTab does not have to know the name of the UserControl class to instantiate at runtime. By default, it loads the DLL configured for each Custom Tab, and then searches for the first public class that implements the IPageView interface. If the DLL contains multiple eligible Custom Tab UserControl classes, only the first is used.

To consolidate Custom Tab UserControl classes into a single DLL, or to give BasicCLRTab a hint on which class to use, apply the CLRTab custom attribute. To use this attribute, first add SCMLib.dll as a project reference and declare the SCMLib namespace.

```
using SCMLib;

/// <summary>

/// Summary description for MyFirstCustomTab.

/// </summary>

[CLRTab("MyFirstCustomTabID")]

public class MyFirstCustomTab : UserControl, IPageView
{
 private IPageViewSink m_MySink = null;
 ...
}

/// <summary>
/// Summary description for MySecondCustomTab.
/// </summary>
[CLRTab("MySecondCustomTabID")]
public class MySecondCustomTab : UserControl, IPageView
{
 private IPageViewSink m_MySink = null;
...

}
```

Figure 3.2: Applying the CLRTab Custom Attribute Using SCMLib

The CLRTab custom attribute allows assigning a unique identifier to each of the Custom Tabs. Simply pass this TabID value into the constructor of the CLRTab attribute as shown. It can be the same as the class name, or something else entirely. Note that the use of this attribute is entirely optional.

If the specified TabID is not found, BasicCLRTab gracefully falls back to using the first UserControl class that implements IPageView.

3.6 Configuring the Custom Tab

After developing the Custom Tab library, tell SCM to add one or more chart sections (tabs) to the user interface and how to compose the Custom Tab. The two mandatory pieces of information that SCM must have are the caption for the Custom Tab and the assembly name containing the UserControl. There is an optional TabID that can also be supplied

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

◆ To provide the caption and AssemblyName information using the environment profile:

1 Create a new section called CLRTab at the highest level of the environment profile. This should already be there after a successful new or upgraded installation of SCM 3.5.

2 Add new entries for the Custom Tab to the CLRTab section.

3 Add new entries to the ViewLibrary section of the environment profile. The ViewLibrary determines what type of tab is added and where it is positioned with respect to the other tabs. The libraries are numbered from 0, which is the leftmost tab.

Note: These steps show how to add a CLRTab section to the environment profile if it does not already exist. If the section already exists, go to "To add an CLRTab entry for a new Custom Tab:".

◆ To add a CLRTab section to the environment profile:

1 In the Configuration Tools window, click Dictionary Maintenance. The Dictionary Maintenance window opens.

2 In the Categories list, click Miscellaneous.

3 In the Dictionaries list, click Environment Profile.

4 Select any top-level section.

5 Click Add This Level. The Add Environment Profile window opens.

6 In the Profile Type field, select Section/Sub Section.

7 Enter a description for the section.

8 In the Code field, enter CLRTab.

9 Click OK.

◆ To add an CLRTab entry for a new Custom Tab:

1 Select the CLRTab section.

2 Click Add Sub Level. The Add Environment Profile window opens.

3 In the Profile Type field, select Entry.

4 In the Code field, enter TabCaption*N*. (*N* represents the next consecutive number from any previous TabCaption entries. If no other TabCaption entries exist, use zero.) For example, if this is the first TabCaption entry, enter TabCaption0. The entries within the CLRTab section is separate and distinct from the entries within the ActiveXTab section.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

5 In the Value field, enter the caption that identifies the Custom Tab.

6 Click OK.

7 Select the CLRTab section.

8 Click Add Sub Level. The Add Environment Profile window opens.

9 In the Profile Type field, select Entry.

10 Enter AssemblyName*N*. (*N* represents the next consecutive number from any previous AssemblyName entries. If no other TabCaption entries, exist, use zero.) For example, if this is the first AssemblyName entry, enter AssemblyName0. The *N* value for TabCaption and AssemblyName must match. They are treated as a pair.

11 In the Value field, enter the name of the assembly (DLL) that contains the UserControl to appear in the body of the Custom Tab. For example, enter "MyCustomTab.dll."

12 Click OK.

Note: Steps 13 to 18 are optional.

13 Select the CLRTab section.

14 Click Add Sub Level. The Add Environment Profile window opens.

15 In the Profile Type field, select Entry.

16 Enter TabID*N*. (*N* represents the same number from matching TabCaption*N* and AssemblyName*N* entries)

17 In the Value field, enter the string that matches the CLRTab custom attribute value for the UserControl.

18 Click OK.

◆ To add new ViewLibrary entries:

1 In the Configuration Tools window, click Dictionary Maintenance. The Dictionary Maintenance window opens.

2 In the Categories list, click Miscellaneous.

3 In the Dictionaries list, click Environment Profile. The Environment Profile Maintenance window opens.

4 Select View Library.

5 Click Add Sub Level. The Add Environment Profile window opens.

6 In the Profile Type field, select Entry.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7 In the Code field, enter ViewLibrary*N*. (*N* represents the next consecutive number from any previous ViewLibrary entries. For example, if the highest ViewLibrary entry is ViewLibrary6, enter ViewLibrary7.)

8 In the Value field, enter BasicCLRTab.dll.

9 Click OK.

10 Select the ViewLibraryCount entry.

11 Click Edit.

12 In the Value field, increase the value by 1 (for example, if the value is 7, enter 8.)

13 Click Close.

This procedure assumes that you want to add the Custom Tab as the last chart section in the row. If wanting to put the tab between two existing chart sections, edit the values in the appropriate ViewLibrary entries for the tabs to reorder. For more information, see the *General Configuration Guide*.

Note: Use and intermix both BasicTab.dll and BasicCLRTab.dll for Custom Tabs. Just remember that BasicTab.dll is only for hosting ActiveX controls, while BasicCLRTab.dll is for hosting .NET UserControls.

3.7 Getting and Setting Context

The key component in SCM context management is the CustomContextObj object. Gaining access to the CustomContextObj object functionality is necessary to get and set context.

◆ **To gain access to the CustomContextObj object:**

1 Add SCMLib.dll as a project reference.

2 Declare the SCMLib.Context namespace.

Getting and setting of context takes place from a Custom Tab or a custom Tools application. It is not possible to get and set context from a custom forms application.

3.7.1 Retrieving Current Patient Information

The currently selected patients' globally unique identifiers (GUID) are found directly on the CustomContextObj. These GUIDs are required to do database queries or to create further objects (for example, a ClientVisit object or a Client object).

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

3.7.1.1 Retrieving GUIDS

The GUIDs for the client, visit and user are read directly from the CustomContextObj.

```
using SCMLib.Context;
Retrieving patient context:
CustomContextObj cc = CustomContextObj.GetInstance();
string sClientGUID = cc.ClientGUID;
string sChartGUID = cc.ChartGUID;
string sVisitGUID = cc.VisitGUID;
```

Figure 3.3: Retrieving GUIDs Using C#

```
Dim cc As CustomContextObj = CustomContextObj.GetInstance()
Dim client_GUID As String = cc.ClientGUID
Dim client_GUID As String = cc.ChartGUID
Dim client_GUID As String = cc.VisitGUID
```

Figure 3.4: Retrieving GUIDs Using VB

3.7.1.2 Retrieving ClientVisit Object

To obtain more information about the currently selected patient visit, create a new ClientVisit object. It is necessary, if integrating with a third-party application, because it is likely to have to send the patient's medical record number (MRN) or date of birth (DOB).

```
// obtain an instance of the current Context Object
CustomContextObj cc = CustomContextObj.GetInstance();
// find the currently selected patient's visitGUID
string vguid = cc.VisitGUID;
//create a ClientVisit object for the current visitGUID
SCMLib.PObj.ClientVisitPObj clientVisit = new
ClientVisitPObj(vguid);
// then able to 'read' data for the clientvisit
string sClientMRN = clientVisit.IDCode;
string sClientName = clientVisit.ClientDisplayName;
```

Figure 3.5: Retrieving ClientVisit Object Using C#

```
' obtain an instance of the current Context Object
Dim cc As CustomContextObj = CustomContextObj.GetInstance()
' find the currently selected patient's visitGUID
Dim visit_GUID As String = cc.VisitGUID
' create a ClientVisit object for the current visitGUID
Dim clientVisit As SCMLib.PObj.ClientVisitPObj = New
ClientVisitPObj(vguid)
' then able to 'read' data for the clientvisit
Dim sClientMRN As String = clientVisit.IDCode
Dim sClientName As String = clientVisit.ClientDisplayName
```

Figure 3.6: Retrieving ClientVisit Object Using VB

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

3.7.1.3 Retrieve Current User Information

The currently selected user's GUID is found directly on the CustomContextObj. This GUID is used to:

- Perform database queries.
- Create a User object.

The GUIDs for the client, visit, and user are read directly from the CustomContextObj.

```
CustomContextObj cc = CustomContextObj.GetInstance();
string sUSerGUID = cc.UserGUID;
```

Figure 3.7: Retrieve Current User Using C#

```
Dim cc As CustomContextObj = CustomContextObj.GetInstance()
Dim user_GUID As String = cc.UserGUID
```

Figure 3.8: Retrieve Current User Using VB

3.7.1.4 Retrieving User Object

To obtain more information about the currently selected user, create a User object.

```
// obtain an instance of the current Context Object
CustomContextObj cc = CustomContextObj.GetInstance();
// find the currently logged on user
string UserGUID = cc.UserGUID;
//create a User object for the current UserGUID
SCMLib.PObj.UserPObj user = new UserPObj (UserGUID);
// then able to 'read' data for the user
string sUserOccupation = user.Occupation;
```

Figure 3.9: Retrieving User Object Using C#

```
' obtain an instance of the current Context Object
Dim cc As CustomContextObj = CustomContextObj.GetInstance()
' find the currently selected patient's visitGUID
Dim user_GUID As String = cc.UserGUID
' create a User object for the current userGUID
Dim user As SCMLib.PObj.UserPObj = New UserPObj(cc.UserGUID)
Dim sUserOccupation = user.Occupation
```

Figure 3.10: Retrieving User Object Using VB

3.8 Event Notifications

Custom tabs can receive both patient and user change notification. The preferred way to respond is to implement the ICustomTab interface from the SCMLib.CustomTab namespace. The ICustomtab interface complements the IPageView interface and does not replace it. The IPageView interface is always mandatory.

```
/// <summary>
/// Summary description for MyFirstCustomTab.
```


This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

```
/// </summary>
 [CLRTab("MyFirstCustomTabID")]
public class MyFirstCustomTab : UserControl, IPageView, ICustomTab
{
This requires you to implement the ICustomTab.InitEvents() method.

public void InitEvents(CustomTabEventSource eventSource)
{
// TODO:  Add MyFirstCustomTab.InitEvents implementation
}
The InitEvents() method is called soon after the
IPageView.AdviseSink() call, which provides you the opportunity to
hook up to some or all of the events.
public void InitEvents(CustomTabEventSource eventSource)
{
// We want to receive patient change notifications
 eventSource.OnPatientChange += new
SCMPatientChangeHandler(eventSource_OnPatientChange);

// We want to receive can patient change notifications
 eventSource.OnCanChangePatient += new
SCMCanChangePatientHandler(eventSource_OnCanChangePatient);

// We want to receive user change notifications
 eventSource.OnUserChange += new
SCMUserChangeHandler(eventSource_OnUserChange);
}

private void eventSource_OnPatientChange()
{
 // The patient has changed, so obtain the latest patient context
 CustomContextObj cc = CustomContextObj.GetInstance();
 string sClientGUID = cc.ClientGUID;
 string sChartGUID = cc.ChartGUID;
 string sVisitGUID = cc.VisitGUID;
 // Refresh my data based on the patient context
}
private void eventSource_OnUserChange()
{
 // The user has changed, so obtain the latest user context
 CustomContextObj cc = CustomContextObj.GetInstance();
 string sUserGUID = cc.UserGUID;
 // Refresh my data based on the user context
}
private bool eventSource_OnCanChangePatient()
{
 // Check for unsaved data
 // Must return true unless there is a good
 // reason for stopping the pending patient change
 return true;
}
```

Figure 3.11: Event Notifications Code

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

3.9 Tab Security

In SCM each CLR tab is responsible for specifying one (or more) security rights required to load it. These security rights are specified declaratively with the CLRTabSecurity attribute for the tab usercontrol class. The Sunrise shell is responsible for checking these rights.

If no rights are specified, the shell interprets this as always load. If the user fails the security check, the tab does not appear at all.

CLRTabSecurity contains a single property in this release: RightsGUID. This GUID can be an existing right supplied by Eclipsys, or enterprise-defined.

These sample rights are shown only as illustration.

```
[CLRTabSecurity("1234")]
public class MyFirstCustomTab : UserControl, IPageView, ICustomTab
{
```

Figure 3.12: Tab Security Rights Code

To indicate that multiple rights should be checked, simply declare additional attributes of the same name.

```
[CLRTabSecurity("1234")]  // "Can view any custom tabs"
[CLRTabSecurity("5678")]  // "Can view MyFirstCustomTab"
public class MyFirstCustomTab : UserControl, IPageView, ICustomTab
{
```

Figure 3.13: Multiple Rights Declarations Code

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

CHAPTER 4: CREATING TOOLS

Contents

4.1 Overview 23
4.2 Development Tools 24
4.5 Custom Tools Application Interface 28
4.3 Calling a Custom Application Using the Tools Menu 24
4.4 Calling a Custom Application from Order Entry 27
4.6 Building a Custom Application 29
4.7 Using The ICustomApp Interface 31
4.8 CLRApp Attribute 31
4.9 ICustomFormApp Interface 32
4.10 Receiving Context Change Notifications 33
4.11 Receiving Context Change Notifications – Modeless Window Support 35
4.12 Accessing Form Context Information 35

4.1 Overview

Custom applications launch external programs either as a separate process or as windows from within SCM. These windows can be modal or modeless and can fully integrate with the SCM context management components.

Custom applications have access to all available context information, receive context change events, and can use services exposed by ObjectsPlus.

In general custom tools applications are used for:

- Arranging SCM data in a custom format.
- A third-party application. The application appears through either:
  - The third-party's application programming interface (API).
  - Queries to the third-party's database.

Reasons to use a custom Tools application rather than a Custom Tab can include:

- Wanting to show an application that contains data not connected to a particular patient, rather than running tab that is associated with a patient.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

- Requiring more display room than a tab allows. There is more display room available when running an application from the Tools menu.

There are two ways to integrate a custom application with SCM:

- Calling a custom application using the environment profile tools menu.
- Calling a Custom Application from Order Entry.

4.2 Development Tools

The 5.0 version of SCM has been developed using the following tools:

- Visual Studio 2005 with Service Pack 1.
- Infragistics NetAdvantage for .NET 2006 Volume 2 with Hotfix 1079.

It is recommended that any custom assemblies developed for prior versions of SCM are upgraded to use the same development tools. Using the same development tools reduces the overall memory footprint.

4.3 Calling a Custom Application Using the Tools Menu

The most common way to call a custom application is to configure an item on the Tools menu. This configuration is done using the Environment Profile Maintenance window in the Configuration Tools application.

◆ **Integrating a custom application with SCM using the Environment Profile Window:**

1 Open the Environment Profile window.

2 Go to HVC Global Workstation and select Tools.

3 Edit the next available HVCApp*N* entry, and then increment the HVCAppCount value as shown in Figure 4.1.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

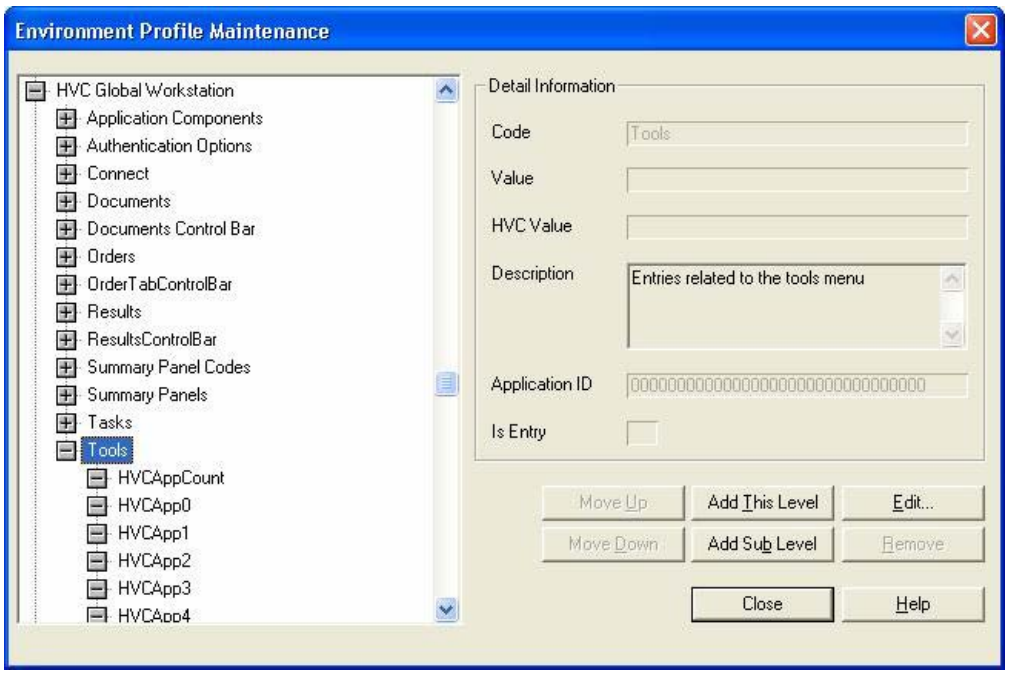


Figure 4.1: Environment Profile Maintenance

4.3.1 Defining Tools Menu Items

The syntax of the HVCApp*N* entry is defined as these tokens delimited by commas.

```
application name, application path and command, context file name,
context file type
```

In most cases, only the first two tokens are used.

- The `application name` token appears as the Tool menu item.
- The `application path and command` token contains the full path and command line used to spawn the tool as a separate running process.
- The optional `context file name` and `context file type` tokens identifies the name of the file used to pass context information to the tool. This type of context passing is a deprecated feature that could be removed in a future release.

This code example defines a tool that launches Notepad.

```
Notepad, notepad.exe
```

This next code example defines another tool.

```
My Calculator, c:\SomeFolder\MyCalc.exe
```


This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

In these examples, selecting the Notepad or My Calculator items from the Tools menu launches their respective .EXE file as a separate process.

4.3.2 Defining Tools Menu Items as Custom Applications

The HVCApp*N* entry syntax has been enhanced to include an optional fifth and sixth token.

```
application name, application path and command, context file name,
context file type, CLRApp, icon file name.ico
```

The `CLRApp` token indicates this particular menu item launches a custom application. If present, the `application path and command` token must only contain the name of the assembly without any path or command line parameters. The third and fourth parameters should also be omitted. If the `CLRApp` token is not specified, the tool is defined and used as before.

Items added to the toolbar appear with an icon that is obtained from the application itself or from a supplied icon file. The file must be in icon format. SCM uses that image for the toolbar button instead of extracting it from the tool executable itself. Token 6 is optional and is only examined if it is configured and applies to both “standard” tools and custom applications (for example, when token 5 = “CLRApp”).

Tools are defined as Environment Profile entries under the HVC Global Workstation\Tools node. The value of each HVCAppN entries is a comma-delimited string. Currently there are six supported tokens as shown in Table 4.1.

Table 4.1: Supported Tokens and Descriptions

Position/Token	Description
1	Menu item appearance text
2	Application path and name, plus optional command-line parameters. This parameter should be enclosed in double quotes if any spaces are present.
3	Context file name (deprecated - not supported in future releases)
4	Context file type (deprecated - not supported in future releases)
5	CLRApp flag (optional)
6	Icon file name (optional)

The icon file specified in token six should contain both a small (16x16) and large (32x32) size image. The toolbar can be toggled between the two sizes.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

This code launches Internet Explorer, and extracts the icon from iexplore.exe.

```
HVCApp0 = My Web App, "C:\Program Files\Internet
Explorer\iexplore.exe"
```

If token 6 specifies an icon file, that image shows in the toolbar button instead.

```
HVCApp0 = My Web App, "C:\Program Files\Internet
Explorer\iexplore.exe,,,CLRApp, c:\temp\myicon.ico
```

Note: The extra comma delimiters are required before the CLRApp token to ensure its position.

Note: The alternate image file must exist on the local machine. If the path points to a remote machine, token 6 is ignored.

Selecting the `My Web App` item from the Tools menu runs as part of the SCM process.

4.4 Calling a Custom Application from Order Entry

In addition to the Tools menu, custom applications can also be called from an order entry form. Calling a custom application in this way is an extension of the CallMLM mechanism where a special button control can be configured to call a .NET component for extra processing.

If the component is defined as a custom form application, it gains access to two new features:

- The full form context that is normally passed to a CalledMLM is also passed to a custom application. This context includes the Communication, Form, and ClientInfo objects (and child objects) that MLM authors could be familiar with.
- A new API that can be used to manipulate this form context, allowing a VB or C# component to perform the same validation done by a MLM. Unlike an MLM, however, a custom application shows one or more windows for capturing or viewing data.

There is no change in how custom applications from order entry are configured. For the control properties of a data item of type `Call MLM`, specify the assembly name of the custom application as the `MLM/Program Name` property value. Set the `Calling Type` property to `Program`. SCM examines the assembly and determines if it contains either a valid custom application component or external form processing component and uses it accordingly.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

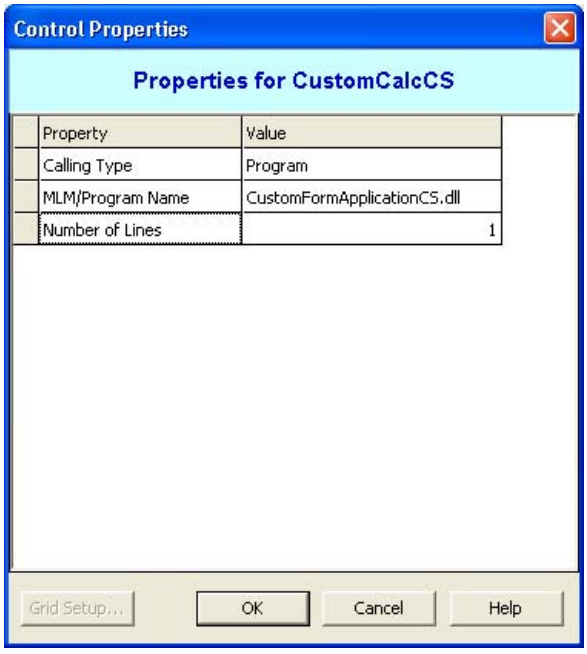


Figure 4.2: Control Properties Window

In Figure 4.2, a form field named CustomCalcCS is configured to call a custom application contained in the CustomForm ApplicationsCS.dll assembly.

4.5 Custom Tools Application Interface

In order for a custom Tools application to function, the ICustomApp interface must be implemented.

4.5.1 ICustomApp

The ICustomApp interface is required to synchronize the DLL with the events that occur in SCM. The interface has three methods that must be implemented. These methods allow the DLL to hook into some SCM events.

```
public int Process()
```

Process() is called when the call is initially instantiated.

```
public void InitEvents(CustomTabEventSource eventSource)
```

InitEvents() is used to sign up for SCM events you want to hook in with.

```
private void UnsubscribeToEvents()
```

UnsubscribeToEvents() is used to stop receiving event notifications when leaving an application.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

These methods are shown in the example under "Building a Custom Application" on page 29. More information on the ICustomApp interface is under "Using The ICustomApp Interface" on page 31

Table 4.2 describes the SCM events that are hooked into when creating a custom Tools application.

Table 4.2: Custom Tools Application Events

Event	Description
OnPatientChange	In this event the majority of code is placed, along with the code to be executed when the user selects a new patient in SCM.
OnUserChange	In this event, place the code to be executed when a new user logs on to SCM. Put code here if user information is cached. Otherwise if one user logs off, the DLL does not realize there is a new user if first user logged off only to the logon screen.
OnCanPatientChange	This event returns TRUE unless a valid reason is given for preventing patient changes in SCM.
OnApplicationSuspend	If attached to a third-party application in the tab, place code in this event to indicate what occurs when SCM suspends.
OnCanLogoff	This event is used rarely, but if code needs to run just before a user logs off, place the code in this event.
OnApplicationTimedOut	If attached to a third-party application in the tab, place the code in this event to indicate the action occurring when SCM times out.

The rest of the chapter describes how custom Tools applications are built, configured, and used.

4.6 Building a Custom Application

To build a new custom application, create a VB or C# class library project containing a public class that implements the SCMLib.CustomApp.ICustomApp interface.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

◆ To build a custom application:

1 Set a reference to the SCMLib.dll assembly. Each public class that implements the ICustomApp interface represents a distinct custom application. Setting a reference to the SCMLib.dll assembly means it is possible to package several custom applications in a single DLL.

2 Each class must also be declared with the SCMLib.CLRApp attribute. This attribute controls the identity of the custom application, as well as its usage.

The Figure 4.3 code example contains a custom application class in VB.

```
Imports System.Windows.Forms
Imports SCMLib
Imports SCMLib.CustomApp

<CLRApp("My First Custom App", CustomAppType.CustomApp)> _
public Class MyCustomApp
    Implements ICustomApp
public Sub InitEvents(ByVal eventSource As
SCMLib.CustomApp.CustomAppEventSource) Implements
SCMLib.CustomApp.ICustomApp.InitEvents
End Sub

    public Function Process() As Integer Implements
SCMLib.CustomApp.ICustomApp.Process
End Function

End Class
```

Figure 4.3: Custom Application Using VB

The Figure 4.4 code example contains a custom application class in C#.

```
using System;
using SCMLib;
using SCMLib.CustomApp;

namespace CustomFormApplicationCS
{
    [CLRApp("My First Custom App", CustomAppType.CustomApp)]
    public class MyCustomApp : ICustomApp
    {
        public MyCustomApp()
        {
//
            // TODO: Add constructor logic here
            //
}
#region ICustomApp Members

        public int Process()
        {
            // TODO:  Add MyCustomApp.Process implementation
            return 0;
        }
public void InitEvents(CustomAppEventSource eventSource)
        {
```


This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

```
            // TODO:  Add MyCustomApp.InitEvents
implementation
        }

        #endregion
    }
}
```

Figure 4.4: Custom Application Using C#

4.7 Using The ICustomApp Interface

The ICustomApp interface is used to locate, initialize, and launch the custom application. There are two methods:

- InitEvents() is called to allow the custom application to subscribe to common events, such as a patient change notification. Developers can add handlers to some, all, or none of the events defined by the CustomAppEventSource parameter inside this method.
- Process() can be called instead of ShowMainDialog(). For now, they should be considered to be identical.

4.8 CLRApp Attribute

The CLRApp attribute is usually specified with two parameters. Using the VB sample shown earlier:

```
Imports SCMLib

<CLRApp("My First Custom App", CustomAppType.CustomApp)>
```

The CLRApp attribute is part of the SCMLib namespace, so it is necessary to declare it first.

The first parameter is the display name of the application. It must match what is specified as the application name when configuring an item for the Tools menu as shown in Figure 4.5.

```
My First Custom App, SampleCustomAppsVB.dll,,,CLRApp
```



Figure 4.5: Tools Menu Example

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

The second parameter for the CLRApp attribute determines the type of application. This parameter uses the CustomAppType enumeration. Currently, there are two options:

- CustomAppType.CustomApp informs SCM that this type is a generic custom application and should only be launched from the Tools menu.
- CustomAppType.CustomFormApp informs SCM this type is a custom form application, and should only be launched from the order entry form. If this type is specified, then the class implements the ICustomFormApp interface instead of the ICustomApp interface. While not mandatory, using the ICustomFormApp interface instructs SCM to pass form context information when launching this application.

4.9 ICustomFormApp Interface

The ICustomFormApp interface contains only the Process(…) method. Unlike the Process() method on the main ICustomApp interface, SCM exchanges form context information with the custom application. Also, there is no InitEvents() method because form applications are usually modal windows.

The Figure 4.6 code example defines a custom form application in VB.

```
Imports SCMLib
Imports SCMLib.CustomApp

<CLRApp("My Custom Form App", CustomAppType.CustomFormApp)> _
public Class MyCustomFormApp
    Implements ICustomFormApp

    public Function Process(ByVal communication As
SCMLib.CustomApp.CommunicationObj,
                        ByVal form As SCMLib.CustomApp.FormObj,
                        ByVal clientInfo As
SCMLib.CustomApp.ClientInfoObj) As Integer
                Implements SCMLib.CustomApp.ICustomFormApp.Process

    End Sub
End Class
The following code example defines a custom form application in C#.

using System;

using SCMLib;
using SCMLib.CustomApp;

namespace SampleCustomAppsCS
{
    [CLRApp("My Custom Form App", CustomAppType.CustomFormApp)]
public class MyCustomFormApp : ICustomFormApp
    {
        public MyCustomFormApp()
        {
        }
```


This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

```
        #region ICustomFormApp Members

       public void Process(CommunicationObj communication, FormObj
form, ClientInfoObj clientInfo)
        {
            // TODO:  Add MyCustomFormApp.Process implementation
        }

        #endregion
    }
}
```

Figure 4.6: ICustomFormapp Interface Using VB

4.10 Receiving Context Change Notifications

When SCM calls the ICustomApp.InitEvents() method, it passes an object of type CustomAppEventSource. This object allows the custom application to subscript any of these events.

- OnCanChangePatient – This event is initiated when the current patient is about to change. Return true to accept this change request; false, otherwise.
- OnPatientChange – This event is initiated when the current patient has actually changed.
- OnUserChange – This event is typically initiated during a re-logon operation, where the current user can be different.
- OnApplicationSuspend – This event is initiated when the main application is about to auto-suspend.
- OnApplicationTimedOut – This event is initiated when the main application has timed out and is about to log off.

The Figure 4.7 code example defines how to subscribe to context change events in VB.

```
public Sub InitEvents(ByVal eventSource As
SCMLib.CustomApp.CustomAppEventSource) Implements
SCMLib.CustomApp.ICustomApp.InitEvents
    AddHandler eventSource.OnCanChangePatient, New
SCMCanChangePatientHandler(AddressOf
eventSource_OnCanChangePatient)
    AddHandler eventSource.OnPatientChange, New
SCMPatientChangeHandler(AddressOf eventSource_OnPatientChange)
    AddHandler eventSource.OnUserChange, New
SCMUserChangeHandler(AddressOf eventSource_OnUserChange)
    AddHandler eventSource.OnCanLogoff, New
SCMCanLogoffHandler(AddressOf eventSource_OnCanLogoff)
End Sub

Private Function eventSource_OnCanChangePatient() As Boolean
    ' Always return true unless there is a good reason not to.
    Return True
```


This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

```
End Function

Private Sub eventSource_OnPatientChange()
    ' TODO: retrieve the latest patient context
End Sub

Private Sub eventSource_OnUserChange()
    ' TODO: retrieve the latest user context
End Sub

Private Function eventSource_OnCanLogoff() As Boolean
    ' Always return true unless there is a good reason not to.
    Return True
End Function
The following code example defines how to subscribe to context
change events in C#.
public void InitEvents(CustomAppEventSource eventSource)
{
    eventSource.OnCanChangePatient += new
SCMCanChangePatientHandler(eventSource_OnCanChangePatient);
    eventSource.OnPatientChange += new
SCMPatientChangeHandler(eventSource_OnPatientChange);
    eventSource.OnUserChange += new
SCMUserChangeHandler(eventSource_OnUserChange);
    eventSource.OnCanLogoff += new
SCMCanLogoffHandler(eventSource_OnCanLogoff);
}
private bool eventSource_OnCanChangePatient()
{
    // Always return true unless there is a good reason not to.
    return true;
}

private void eventSource_OnPatientChange()
{
    // TODO: retrieve the latest patient context
}

private void eventSource_OnUserChange()
{
    // TODO: retrieve the latest user context
}

private bool eventSource_OnCanLogoff()
{
    // Always return true unless there is a good reason not to.
    return true;
}
```

Figure 4.7: Receiving Context Change Notifications Using VB

It is the responsibility of the custom application to unsubscribe when the user closes its main window.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

4.11 Receiving Context Change Notifications – Modeless Window Support

The CustomAppEventSource object is available to any modeless window so they can receive context change notifications.

It is not necessary to implement the ICustomApp interface and wait for the InitEvents() method to be called. Instead, retrieve it from context at any time using the CustomContextObj.CustomAppEventSource property. .

The Figure 4.8 code example defines how to retrieve and use the CustomAppEventSource object in VB.

```
Imports SCMLib.Context

Private Sub SubscribeSCMEvents()
    Dim cc As CustomContextObj = CustomContextObj.GetInstance()
    If Not cc Is Nothing Then
Dim eventSource As CustomAppEventSource = cc.CustomAppEventSource
If Not eventSource Is Nothing Then
            AddHandler eventSource.OnPatientChange, New
SCMPatientChangeHandler(AddressOf eventSource_OnPatientChange)
            ' subscribe to any other events
End If
    End If
End Sub
The following code example defines how to retrieve and use the
CustomAppEventSource object in C#.

using SCMLib.Context;

private void SubscribeSCMEvents()
{
    CustomContextObj cc = CustomContextObj.GetInstance();
    if ( null != cc )
    {
        CustomAppEventSource eventSource = cc.CustomAppEventSource;
         if ( null != eventSource )
         {
             eventSource.OnPatientChange += new
SCMPatientChangeHandler(eventSource_OnPatientChange);
             // subscribe to any other events
         }
    }
}
```

Figure 4.8: CustomAppEventSource Objects Using VB

4.12 Accessing Form Context Information

Form context refers to information that an order entry form can send to a CalledMLM for extra processing. This context includes the state of the fields on the form, such as their attributes and values. Form context is sent to custom

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

applications using these objects, as part of the ICustomFormApp.Process() method call:

- CommunicationObj – This object contains general information pertaining which business object is making this call, and why.
- FormObj – Represents the order entry form. It allows access to the collection of form field objects.
- ClientInfoObj – A helper object that provides information about the current patient. This object is mainly for convenience.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

CHAPTER 5: CALLING CUSTOM DLLS FROM FORMS

Contents

5.1 Overview 37
5.2 Development Tools 38
5.3 Custom Form Applications Interface 39
5.4 Integrating Custom Form Applications 41
5.5 Using the CommunicationObj Class 43
5.6 Using the ClientInfoObj Class 43
5.7 Using the FormObj Class 43
5.8 Class Information 48
5.9 Field Types 57
5.10 Field Value Types 63
5.11 External Form Processing API 82

5.1 Overview

Custom form applications are designed to be called from a button field on an order entry form. This feature allows Eclipsys customers to add processing and data capture into the order entry workflow. These applications gain full access to the current form context information, such as the collection of form fields and their values. The designer determines if these applications have additional windows appear to the user.

In general, custom form applications are used to :

- Create custom calculators for complex drug calculations.
- Present data from third-party systems (for example, drug dose ranges).
- Improve workflow on Order and task forms.

There are similarities between custom form applications and the SCM function CalledProgram. Both are available and can be called from the same order form. However, the custom form application API is the preferred method. Detailed information on the CalledProgram API is contained in "External Form Processing API" on page 82.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

An order entry form can include one or more fields of the type Call MLM. Each of these fields consists of a text field and a command button. The button is configured so that when it is clicked an MLM or a program is executed. The MLMValidation engine component is responsible for exchanging form context between the form (and its associated business object), and the CDS runtime. This engine builds, parses, and processes the objects that comprise the form context information.

Note: This command button is available only when an order is being created or modified and is not active when viewing an order.

In Figure 5.1, these form context objects can also be exchanged with custom form applications.

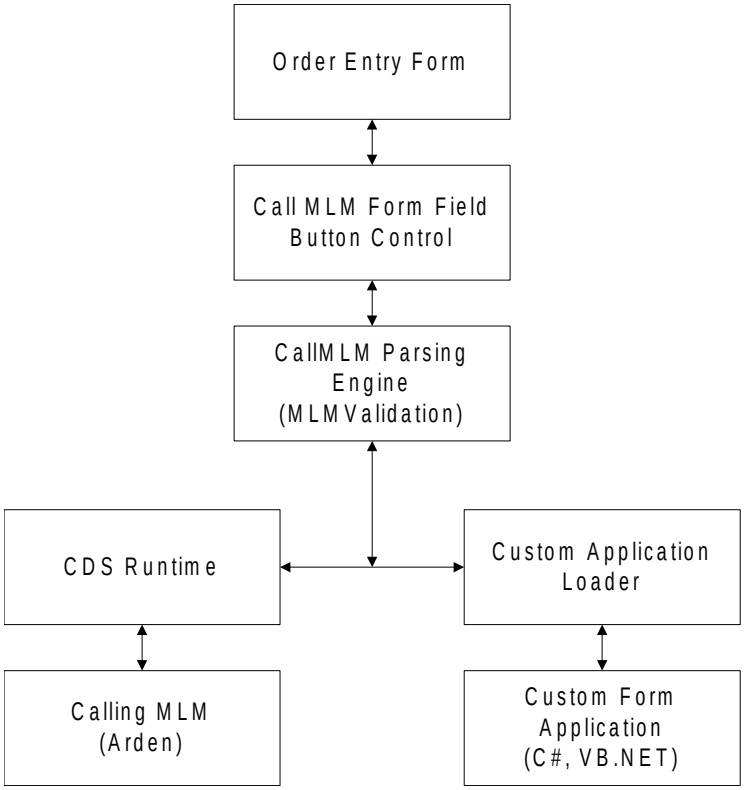


Figure 5.1: Relationship Between Called MLMs and Custom Form Applications

5.2 Development Tools

The 5.0 version of SCM has been developed using the following tools:

- Visual Studio 2005 with Service Pack 1.
- Infragistics NetAdvantage for .NET 2006 Volume 2 with Hotfix 1079.

It is recommended that any custom assemblies developed for prior versions of SCM are upgraded to use the same development tools. Using the same development tools reduces the overall memory footprint.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

5.3 Custom Form Applications Interface

In order for a custom form application to function, the ICustomFormApp interface must be implemented.

5.3.1 ICustomFormApp

The SCMLib.CustomApp.ICustomFormApp interface contains a single method, Process(...). This method is unlike the Process() method on the main ICustomApp interface because SCM exchanges form context information with the custom application. Also, there is no InitEvents() method because form applications are modal windows.

This example implements the ICustomFormApp interface.

```
using System;
using SCMLib;
using SCMLib.CustomApp;
namespace CustomFormApp
{
[CLRApp("Custom Form App", CustomAppType.CustomFormApp)]
public class CustomFormApp :
System.Windows.Forms.Form,ICustomFormApp
{
public CustomFormApp()
{
}
#region ICustomFormApp Members
Int SCMLib.CustomApp.ICustomFormApp.Process(
CommunicationObj communication,
FormObj form,
ClientInfoObj clientinfo)
{
// TODO: Process implementation
return 0;
}
#endregion
}
}
```

Figure 5.2: ICustomFormApp Interface

5.3.2 Form Context Information

Form context is information that an Order Entry form sends to a custom DLL for extra processing. Form context includes the state of the fields on the form, such as attributes and values.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

As part of the ICustomFormApp.Process() method call, form context is sent to custom form applications using these objects:

- Communication object
- Form object
- ClientInfo object

Before beginning to write applications to call from forms, Eclipsys recommends becoming familiar with the data passed between forms and called applications.

Table 5.1 describes the objects available to the called application.

Table 5.1: Available Objects

Object	Description
Communication object (CommunicationObj)	This object contains general information about which business object is making this call, and why. It contains information about the client, user, care provider, order or task, and the form. Certain fields from the Communication object can be modified and then returned to the form. The Current Primary Object is available as sub-object on the Communication Object. It includes data from Order, OrderSet, or Order Task Occurrence objects.
Form object (FormObj)	This object represents the order entry form. It includes information about the fields on the form and contains field objects for each of the form fields. Certain fields from the Form object can be modified and then returned to the form. Field objects are children of the Form object.
ClientInfo object (ClientInfoObj)	This helper object provides information about the current patient and is mainly for convenience.

For more information on the Communication, Form, or ClientInfo objects see "Using the CommunicationObj Class" on page 43, "Using the FormObj Class" on page 43, or "Using the ClientInfoObj Class" on page 43.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

Only these forms can be configured to call custom DLLs

Table 5.2: Forms to Call DLLs

Subset	Description
Order Form	A single current order data subset is available.
OrderSet Form (multi-order form)	A single OrderSet data subset is available.
Task Forms	A single Occurrence data subset is available.

5.4 Integrating Custom Form Applications

To integrate with SCM, a custom form application must provide a public class that implements the SCMLib.CustomApp.ICustomFormApp interface, and be decorated with the SCMLib.CLRApp attribute. The ICustomFormApp interface defines a Process() method that serves as the main entry point.

After configuration, clicking the form field button triggers a call to this Process() method. Figure 5.3 and Figure 5.4 contain examples of this integration.

```
using System;
using System.Windows.Forms;
using SCMLib;
using SCMLib.CustomApp;

namespace CustomFormApplicationCS
{
    [CLRApp("CustomCalculator",CustomAppType.CustomFormApp)]
    public class CustomCalculator : ICustomFormApp
    {
        public CustomCalculator()
        {
        }

        int SCMLib.CustomApp.ICustomFormApp.Process
        (
            CommunicationObj communication,
            FormObj form,
            ClientInfoObj clientInfo
        )
        {
            return 0;
        }

    }
}
```

Figure 5.3: Integration Code Example Using C#

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

```
Imports System.Windows.Forms
Imports SCMLib
Imports SCMLib.CustomApp

<CLRApp("CustomCalculator", CustomAppType.CustomFormApp)> _
public Class CustomCalculator
    Implements ICustomFormApp

    public Function Process
    (
        ByVal communication As SCMLib.CustomApp.CommunicationObj,
        ByVal form As SCMLib.CustomApp.FormObj,
        ByVal clientInfo As SCMLib.CustomApp.ClientInfoObj
    ) As Integer Implements SCMLib.CustomApp.ICustomFormApp.Process

    End Function

End Class
```

Figure 5.4: Integration Code Example Using VB

5.4.1 Writing a Hello World Application

To show an example of application form integration, a Hello World message is coded. Specifically, the CommunicationsObj context object is used to pass a message back to the form. The Message and MessageType properties are set to do this in Figure 5.5 and Figure 5.6.

```
// Test: return a message from the calculator
communication.MessageType = MessageTypes.Banner;
communication.Message = "Hello from CustomCalculator CS!";
```

Figure 5.5: Hello World Using C#

```
' Test: return a message from the calculator
communication.MessageType = MessageTypes.Banner
communication.Message = "Hello from CustomCalculator VB!"
```

Figure 5.6: Hello World Using VB

The MessageTypes enumeration has three entries:

- Banner
- Error
- Informational

An example of each is shown in Figure 5.7, Figure 5.8, and Figure 5.9 respectively.

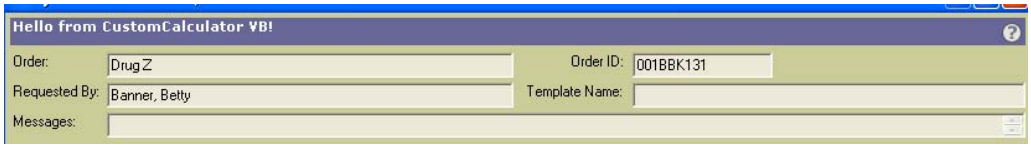


Figure 5.7: Banner Message Type

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

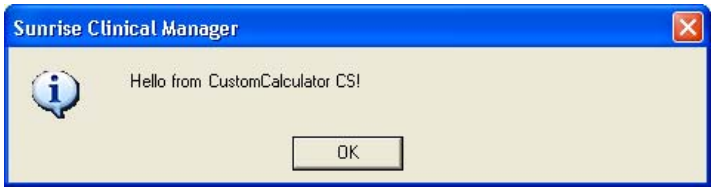


Figure 5.8: Informational Message Type



Figure 5.9: Error Message Type

5.5 Using the CommunicationObj Class

In addition to the Message and MessageType properties, the CommunicationObj class contains the identity of the field that triggered this application as well as other context information.

Use the Communication object to pass information about the:

- Field or form calling the application.
- Current patient and item (for example, the order item or task).

Class information is found under "CommunicationObj Class" on page 48.

5.6 Using the ClientInfoObj Class

An instance of the ClientInfoObj class is also used through the ICustomFormApp.Process() method. This is a collection of context information about the current patient, such as the birth date and gender.

Class information is found under "ClientInfoObj Class" on page 80.

5.7 Using the FormObj Class

The remainder of this chapter focuses on the FormObj context object, that represents the order entry form itself. This object provides access to the collection of fields on the form, as well as their properties and values.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

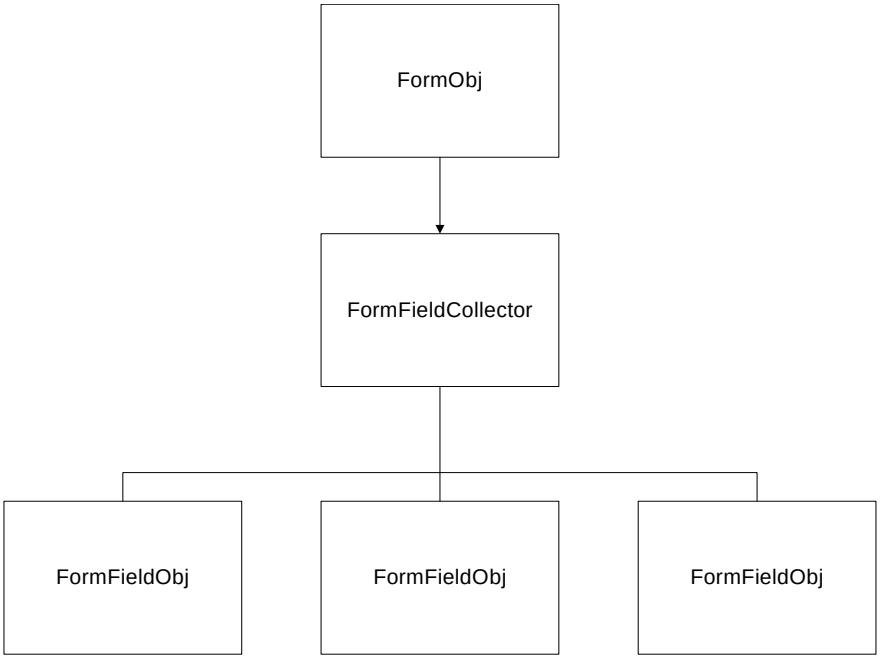


Figure 5.10: FormObj Class Hierarchy

5.7.1 Accessing All Fields on a Form

The FormObj.Fields property provides a way to iterate through the entire collection of fields. Each item in the collection is of type FormFieldObj.

```
// Examine the label and data item of each field
foreach (FormFieldObj field in form.Fields)
{
    string Label = field.Label;
    string DataItemName = field.DataItemName;
    msg = string.Format("Label: {0}   DataItemName: {1}", Label,
DataItemName);
    //MessageBox.Show(msg);
}
```

Figure 5.11: Accessing All Fields on a Form Using C#

```
' Test: examine the label and data item for each field
Dim field As FormFieldObj
For Each field In form.Fields
    Dim Label As String = field.Label
    Dim DataItemName As String = field.DataItemName
    msg = String.Format("Label: {0}  DataItemName: {1}", Label,
DataItemName)
    'MessageBox.Show(msg)
Next
```

Figure 5.12: Accessing All Fields on a Form Using VB

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

5.7.2 Finding a Specific Form Field

The FormObj class provides methods for finding a field either by its data item code or label. If successful, the FindField() and FindFieldByLabel() method returns a reference to a FormFieldObj. If the required field does not exist, these methods returns null or nothing.

```
/ Test: Find the SpecialInstructions field and update it
FormFieldObj SpecialInstructions;
SpecialInstructions = form.FindField("SpecialInstructions");
if ( null != SpecialInstructions )
{
    SpecialInstructions.Value = "Changed by CS calculator.";
}

// Test: Find the SpecialInstructions field and update it
FormFieldObj SpecialInstructions;
SpecialInstructions =
form.FindFieldByLabel("SpecialInstructions");
if ( null != SpecialInstructions )
{
    SpecialInstructions.Value = "Changed by CS calculator";
}
```

Figure 5.13: Finding a Specific Form Field Using C#

```
' Test: Find the special instructions field, and update it
Dim SpecialInstructions As FormFieldObj
SpecialInstructions = form.FindField("SpecialInstructions")
If Not field Is Nothing Then
    field.FieldValue = "Changed by VB calculator."
End If

' Test: Find the special instructions field, and update it
Dim SpecialInstructions As FormFieldObj
SpecialInstructions = form.FindFieldByLabel("SpecialInstructions")
If Not SpecialInstructions Is Nothing Then
    SpecialInstructions.Value = "Changed by VB calculator."
End If
```

Figure 5.14: Finding a Specific Form Field Using VB

If there are multiple fields with the same data item code, the FindField() method is called with an occurrence number as the second parameter.

```
FormFieldObj SecondMultiOrderGrid =
form.FindField("MultiOrderGrid",2);
if ( null != SecondMultiOrderGrid )
{
    // Found it!
}
```

Figure 5.15: FindField() Method Using C#

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

```
Dim SecondMultiOrderGrid As FormFieldObj =
form.FindField("MultiOrderGrid", 2)
If Not SecondMultiOrderGrid Is Nothing Then
    ' Found it!
End If
```

Figure 5.16: FindField() Method Using VB

The FindFieldByLabel() method does not allow an occurrence number to be passed in, because all field labels on a form should be unique.

5.7.3 Changing Field Properties

After the field is found, the FormFieldObj class allows changing of the read only, mandatory, and visible states. Change this state using the Control_Read_Only, Control_Mandatory, and Control_Visible properties.

```
FormFieldObj SpecialInstructions =
form.FindField("SpecialInstructions");
if ( null != SpecialInstructions )
{
    // Disable the field
    SpecialInstructions.Control_Read_Only = true;
    // Hide the field
    SpecialInstructions.Control_Visible = false;
    // Make field mandatory
    SpecialInstructions.Control_Mandatory = true;
}
```

Figure 5.17: Changing Field Properties Using C#

Changing Field

```
Dim SpecialInstructions As FormFieldObj =
form.FindField("SpecialInstructions")
If Not SpecialInstructions Is Nothing Then
    ' Disable the field
    SpecialInstructions.Control_Read_Only = True
    ' Hide the field
    SpecialInstructions.Control_Visible = False
    ' Make field mandatory
    SpecialInstructions.Control_Mandatory = True
End If
```

Figure 5.18: Changing Field Properties Using VB

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

5.7.4 Changing Field Values

After the wanted field is obtained, its value can be changed using its Value property. This property is of type System.Object. For fields with simple value types, such as a string, integer, or Boolean, the Value property contains the actual field value.

```
FormFieldObj field = form.FindField("RequestedDate");
if ( null != field )
{
    // try setting the date value
    field.Value = "2004-04-01";
}
```

Figure 5.19: Changing Field Values Using C#

```
Dim field As FormFieldObj = form.FindField("ResultPriorityCode")
If Not field Is Nothing Then
    field.Value = "STAT"
    field.Control_Read_Only = True
End If
```

Figure 5.20: Changing Field Values Using VB

For complex field types, however, the FormFieldObj.Value property must be cast to the appropriate date type.

```
FormFieldObj field = form.FindField("FrequencyCode");
if ( null != field )
{
    FrequencyCodeValueObj fieldValue = field.Value as
FrequencyCodeValueObj;
    if ( null != fieldValue )
    {
        double nFreqFromTime = fieldValue.FreqFromTime;
        // try decrementing by 1
        if ( nFreqFromTime > 1 )
        {
            fieldValue.FreqFromTime = nFreqFromTime - 1;
        }
    }
}
```

Figure 5.21: Changing Complex Field Values Using C#

```
Dim field As FormFieldObj = form.FindField("CollapsibleWarning")
If Not field Is Nothing Then
    Dim fieldValue As CollapsibleWarningValueObj = field.Value
    If Not fieldValue Is Nothing Then
        ' Test: toggle the IsCollapsed property
        Dim IsCollapsed As Boolean = fieldValue.IsCollapsed
        fieldValue.WarningMessage = "Testing Warning Message"
    End If
End If
```

Figure 5.22: Changing Complex Field Values Using VB

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

5.7.5 Type-Safe Field Access

The FormFieldCollection class has a set of properties that allows access to fields directly. Allowing access to fields avoids having to perform an explicit FindField() operation. These properties return a type-safe field reference, such as FrequenceCodeFieldObj, instead of the base FormFieldObj type.

In this case, use the FieldValue property instead of the Value property, for the same reason that it is type-safe.

```
FrequencyCodeFieldObj field = form.Fields.FrequencyCodeField;
if ( null != field )
{
    FrequencyCodeValueObj fieldValue = field.FieldValue;
    if ( null != fieldValue )
    {
        double nFreqFromTime = fieldValue.FreqFromTime;
        // try decrementing by 1
        if ( nFreqFromTime > 1 )
        {
            fieldValue.FreqFromTime = nFreqFromTime - 1;
        }
    }
}
```

Figure 5.23: Type-Safe field Access Using C#

```
Dim field As AdditionalVerificationByFieldObj =
form.Fields.AdditionalVerificationField
If Not field Is Nothing Then
    Dim fieldValue As AdditionalVerificationByValueObj =
field.FieldValue
    If Not fieldValue Is Nothing Then
        Dim sIDCode As String = fieldValue.IDCode
    End If
End If
```

Figure 5.24: Type-Safe field Access Using VB

Class information is found under "FormObj Class" on page 56.

5.8 Class Information

This next section contains the detailed class information for all the form context objects.

5.8.1 CommunicationObj Class

A helper object that provides context information for launching this application. Many of the properties are used by MLMs only, and are available here for consistency.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

Table 5.3: CommunicationObj Class (1 of 4)

Property Name	Required?	Data type	Description
AlertDrugFood	Yes	Boolean	The Boolean flag extracted from the environment profile CDS/Drug Interaction Alerts/Alert Drug Food.
ApplicationName	Yes	String	Name of the application that is hosting the Form SCM.
BSAFormula	Yes	String	String value extracted from the environment profile CDS/Drug Interaction Alerts/BSA Formula.
CallingEvent	Yes	String	The Event calling the application. For example, FormOpen, FormClose, FieldChange, or ButtonClick.
CallingFieldName	Yes, if called from a field or button	String	Name of the data item field calling the application. For example, DosageLow, or RequestedDate.
CareProviderGUID	Yes	String	GUID of the user requesting the order or task. If the order is entered using the interfaces (for example, HL7), this field can be 0.
ChartGUID	Yes, it at the visit level	String	GUID from the currently selected visit chart grouping or blank if this is being performed at a general level rather than a visit level (for example, Insurance).
ClientGUID	Yes	String	GUID from the currently selected client.
ClientVisitGUID	Yes, if at the visit level	String	The GUID from the currently selected visit or blank if this is being performed at a general level rather than a visit level (for example, Insurance).

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

Table 5.3: CommunicationObj Class (2 of 4)

Property Name	Required?	Data type	Description
DisplayForm	Yes	Boolean	For order forms, indicates if the form is displayed to the user. The application uses this field to force the form to appear. Valid values are Yes, No, or NoChange. For task and order set forms, this can be set only to Yes.
Fields	Yes	String	All the filed objects on the form.
FormGUID	Yes	String	The GUID of the form.
FormType	Yes	String	Type of the form (Order, Task, or OrderSet).
FormVersionGUID	Yes	String	The GUID of the current form version.
ItemEvent	Yes	String	The function the user is performing on this item. For an order, this can be UserEnter, UserModify, UserEditUnsubmitted, SystemEnter, or SystemModify. OrderSet form = Active.
ItemID	Yes	String	The identifier from the table item using this form. Order form = CV3OrderCatalogMasterItem.GUID Task form = CV3OrderCatalogMasterItem.GUID OrderSet form = CV3OrderCatalogSet.GUID
ItemStatus	Yes	String	Status of the item using the form. Order form examples can be: PEND, PCOL, or AUA1. Non-order form examples can be: Active, Inactive, or Discontinued.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

Table 5.3: CommunicationObj Class (3 of 4)

Property Name	Required?	Data type	Description
ItemTable	Yes	String	The table containing the item this form applies to. Order form = CV3OrderCatalogMasterItem Task form = CV3OrderCatalogMasterItem OrderSet form = CV3OrderCatalogSet
Message	No	String	Any message appearing to the user.
MessageType	No	Message Types	How the message is presented to the user. Informational: appears in a message box with an information icon. Error: appears in a message box with an error icon. Banner: appears at the top of a form.
Name	Yes	String	The name of the form.
OrderSet	Yes	String	Indicates whether this call is from an order that is part of an order set or a multi-order form. Valid values are: Yes, No, Linked, or Generic.
OrderType	Yes, if this is an order.	String	If the FormType is Order, this indicates the type of order form (for example, Medication, Diagnostic, or Other). This value is null if it is for Task or Order Set FormType.
ParentOrderSet	Yes	String	Defines if or what type of set the Parent Set is. Indicates whether this call is from an order contained in a set within a set. Valid values are: Yes, No, or Linked.
ParentOrderSetGUID	Yes	String	GUID of the ParentOrderSet if this call comes from an item in a set within another set, or from a multi-order form set that is contained within another set.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

Table 5.3: CommunicationObj Class (4 of 4)

Property Name	Required?	Data type	Description
PrimaryObj	Yes	Object	PrimaryObj is a reference to an object containing selected fields from an Order, OrderSet or OrderTaskOccurrence object. The nature of this object depends on which form called the MLM. For a list of the fields found in this object, see "PrimaryOrderObj Class" on page 52. The internal names of the objects are "PrimaryOrderType," "PrimaryOrderSetType," and "PrimaryOrderTaskOccurrence Type."
SeverityLevel	Yes	Integer	Numeric value extracted from the environment profile CDS/Drug Interaction Alerts/Drug Severity Level.
UserGUID	Yes	String	GUID of the user entering the order or task.
Workstation GUID	Yes	String	GUID for the workstation linked to the table Workstation in the enterprise server database.
WorkstationName	Yes	String	Name of the workstation where the Form application is being run.

5.8.2 PrimaryOrderObj Class

The current_primary_object is available as a sub-object on the Communication object. This object contains the data from one of these sources:

- OrderSet from a Multi-order form.
- Order object from Order form.
- OrderTaskOccurrence Object from Task form.

Note: Not all data is present on the Current Primary Object when the application is called.

A public class that wraps the structure of the primary order object defined under "PrimaryObj" on page 52.

The Parent class for PrimaryOrderObj class is FormContextBaseObj.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

Table 5.4 describes the attributes of the primary object.

Table 5.4: PrimaryOrderObj Class (1 of 2)

Property Name	Required?	Data type	Description
ComplexOrderType	Yes, if this is an order.	String	Defines the type of order. 0 or NULL: Regular order. 1: Sequential Complex Dose Master. 2: Sequential Complex Dose Child Order. 3: Concurrent Complex Dose Master. 4: Concurrent Complex Dose Child Order. 5: Multiple Frequencies Master. 6: Multiple Frequencies Child Order.
EnterRole	Yes, if this is an order.	String	Role of the person who entered the order. (CV3OrderRoleType)
GUID	Yes	String	Primary Key, which is a unique identifier.
HoldReasonText	Yes, if this is an order.	String	The reason for the hold order.
IsConditional	Yes, if this is an order.	Boolean	Indicates if this is a conditional order.
IsDurationChangeable	Yes, if this is an order.	Boolean	Indicates if the duration of the order can be changed.
IsforDischarge	Yes, if this is an order.	Boolean	Indicates if this is a discharge order.
IsHeld	Yes, if this is an order.	Boolean	Indicates whether the order is a hold order.
IsPartOfSet	Yes, if this is an order.	Boolean	Indicates if this order is part of a set.
IsSuspended	Yes, if this is an order.	Boolean	Indicates if this order is suspended.
IsTrackVariance	Yes, if this is an order.	Boolean	Indicates whether this order is tracked for variance charting.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

Table 5.4: PrimaryOrderObj Class (2 of 2)

Property Name	Required?	Data type	Description
Modifier	Yes, if this is an order.	String	The modifier of the order item. This is copied from the Order Catalog Item.
Name	Yes, if this is an order.	String	The name of the order item.
OrderPriorityCode	Yes, if this is an order.	String	The order priority code. (Cv3OrderPriority)
OrderSetHeading	Yes, if this is an order.	String	Heading of the order within the Order Set, if applicable.
OrderSetName	Yes, if this is an order.	String	The name of the Order Set, if applicable.
OrderSetType	Yes, if this is an order.	Integer	Stores the order catalog set type, if applicable.
OrderTemplateGUID	Yes, if this is an order.	String	GUID of the pre-filled order item that this order was created from, of applicable. (CV3Order)
SourceCode	Yes, if this is an order.	String	The source of the order. (CV3Source)
SystemOrderPriorityCode	Yes, if this is an order.	String	The internal OrderPriorityCode Valid values are: STAT, TIMECRITICAL, or ROUTINE.

5.8.3 PrimaryOrderTaskOccurrenceObj Class

A public class that wraps the structure of the primary order task occurrence object defined under "PrimaryObj" on page 52.

The PrimaryOrderTaskOccurrenceObj Parent class is FormContextBaseObj.

Table 5.5: PrimaryOrderTaskOccurrenceObj Class (1 of 2)

Property Name	Data type
GUID	String
OrderGUID	String
OrderTaskGUID	String
CatalogItemTaskGUID	String

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

Table 5.5: PrimaryOrderTaskOccurrenceObj Class (2 of 2)

Property Name	Data type
TaskName	String
ScheduledDtm	DateTime
EarliestScheduledDtm	DateTime
LatestScheduledDtm	DateTime
EnteredProviderGUID	String
PerformProviderGUID	String
EnteredDtm	DateTime
PerformedFromDtm	DateTime
PerformedToDtm	DateTime
TaskStatusCode	String
SignificantDtm	DateTime
SignificantTime	String
IsSuspended	Boolean
RequireReschedule	Boolean
OverdueMinute	Integer
OverdueMustReschedMinute	Integer
IsTrackVariance	Boolean

5.8.4 PrimaryOrderSetObj Class

A public class that wraps the structure of the primary order set object defined under "PrimaryObj" on page 52.

The PrimaryOrderSetObj parent class is FormContextBaseObj.

Table 5.6: PrimaryOrderSetObj Class

Property Name	Data type
GUID	String
OrderSetName	String
OrderSetStatusCode	String
OrderSetStatusLevelNum	Integer
OrderSetType	Integer

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

5.8.5 FormObj Class

This class represents the order entry form itself.

Table 5.7: FormObj Class

Property Name	Data type
Name	String
Fields	FormFieldCollection

5.8.6 FormFieldObj Class

This class represents a single field on a form. It is also the base class for these specialized field types.

Table 5.8: FormFieldObj Class

Property Name	Data type
Control_Mandatory	Boolean
Control_MultiFieldOccNum	Integer
Control_Read_Only	Boolean
Control_Updateable	Boolean
Control_Visible	Boolean
DataItemGUID	String
DataItemName	String
DataType	String
FieldLength	Integer
Label	String
Value	Object

5.8.7 FormFieldCollection Class

This class represents the collection of fields on a form.

Table 5.9: FormFieldCollection Class (1 of 2)

Property Name	Data type
AdditionalVerificationField	AdditionalVerificationFieldObj
AdminInstructionsField	AdminInstructionsFieldObj
BaseSolutionField	BaseSolutionFieldObj

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

Table 5.9: FormFieldCollection Class (2 of 2)

Property Name	Data type
CalculatedDoseField	CalculatedDoseFieldObj
CollapsibleWarningField	CollapsibleWarningFieldObj
CombinedMeasurementsField	CombinedMeasurementsFieldObj
ConditionalOrderField	ConditionalOrderFieldObj
CreatinineClearanceField	CreatinineClearanceFieldObj
DosageLowField	DosageLowFieldObj
EnhancedTimeField	EnhancedTimeFieldObj
FrequencyCodeField	FrequencyCodeFieldObj
IsNoTaskField	IsNoTaskFieldObj
IsPRNField	IsPRNFieldObj
IVAdditiveCalcField	IVAdditiveCalcFieldObj
IVAdditiveField	IVAdditiveFieldObj
MultiOrderCheckboxField	MultiOrderCheckboxFieldObj
MultiOrderGridField	MultiOrderGridFieldObj
MultiOrderInlineField	MultiOrderInlineFieldObj
OrderIVRateField	OrderIVRateFieldObj
RequestedTimeField	RequestedTimeFieldObj
SpecialInstructionsField	SpecialInstructionsFieldObj
StopAfterField	StopAfterFieldObj
StopTimeField	StopTimeFieldObj
VariableDoseField	VariableDoseFieldObj

5.9 Field Types

This section contains the list of specialize field types. They derive from the base FormFieldObj class and expose a new FieldValue property. Whereas the FormFieldObj.Value property is always of type System.Object, and has to be cast to the appropriate type, the FieldValue property for these classes is type safe.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

5.9.1 AdditionalVerificationByFieldObj Class

This class represents an AdditionalVerificationBy field.

Table 5.10: AdditionalVerificationByFieldObj Class

Property Name	Data type
FieldValue	AdditionalVerificationByValueObj

5.9.2 AdminInstructionsFieldObj Class

This class represents an AdminInstructions field.

Table 5.11: AdminInstructionsFieldObj Class

Property Name	Data type
FieldValue	String

5.9.3 BaseSolutionFieldObj Class

This class represents a BaseSolution field.

Table 5.12: BaseSolutionFieldObj Class

Property Name	Data type
FieldValue	BaseSolutionValueObj

5.9.4 CalculatedDoseFieldObj Class

This class represents a CalculatedDose field. The FormFieldObj.Value property for this field can be an object of type CalculatedDoseValueObj or ComplexDosingGridValueObj if complex dosing grid is shown.

If the field is blank, Value property always returns object of type CalculatedDoseValueObj. The user has to cast to the proper object before value can be retrieved. The FieldValue and ComplexOrderValue property provides type safe access to the value of this field.

Note: For regular calculated dosage, “FieldValue” property has a value and “ComplexOrderValue” is null. For complex dosing, “FieldValue” property is null and “ComplexOrderValue” returns object of type ComplexDosingGridValueObj.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

Table 5.13: CalculatedDoseFieldObj Class

Property Name	Data type
FieldValue	CalculatedDoseValueObj
ComplexOrderValue	ComplexDosingGridValueObj

5.9.5 CollapsibleWarningFieldObj Class

This class represents a CollapsibleWarning field.

Table 5.14: CollapsibleWarningFieldObj Class

Property Name	Data type
FieldValue	CollapsibleWarningValueObj

5.9.6 CombinedMeasurementsFieldObj Class

This class represents a CombinedMeasurements field.

Table 5.15: CombinedMeasurementFieldObj Class

Property Name	Data type
FieldValue	CombinedMeasurementsValueObj

5.9.7 ConditionalOrderFieldObj Class

This class represents a ConditionalOrder field.

Table 5.16: ConditionalOrderFieldObj Class

Property Name	Data type
FieldValue	ConditionalOrderValueObj

5.9.8 CreatinineClearanceFieldObj Class

This class represents a CreatinineClearance field.

Table 5.17: CreatinineClearanceFieldObj Class

Property Name	Data type
FieldValue	CreatinineClearanceValueObj

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

5.9.9 DosageLowFieldObj Class

This class represents a DosageLow field. The FormFieldObj.Value property for this field can be an object of type Double or ComplexDosingGridValueObj if complex dosing grid appears. The user has to cast to the proper object before value can be retrieved. The FieldValue and ComplexOrderValue property provides type safe access to the value of this field.

Note: For regular dosage, “FieldValue” property has a numeric value and “ComplexOrderValue” is null. For complex dosing, “FieldValue” property is 0.0 and “ComplexOrderValue” returns object of type ComplexDosingGridValueObj.

Table 5.18: DosageLowFieldObj Class

Property Name	Data type
FieldValue	Double
ComplexOrderValue	ComplexDosingGridValueObj

5.9.10 FrequencyCodeFieldObj Class

This class represents a FrequencyCode field. The FormFieldObj.Value property for this field can be object of type FrequencyCodeValueObj or MultipleFrequencyGridValueObj if complex dosing appears.

If the field is blank, Value property always returns object of type FrequencyCodeValueObj. The user has to cast to proper object before value can be retrieved. The FieldValue and ComplexOrderValue properties provide type safe access to the value of this field.

Note: For regular frequency, “FieldValue” property has a value of type FrequencyCodeValueObj and “ComplexOrderValue” property is null. For multiple frequency, “FieldValue” property is null and “ComplexOrderValue” property returns object of type MultipleFrequencyGridValueObj.

Table 5.19: FrequencyCodeFieldObj Class

Property Name	Data type
FieldValue	FrequencyCodeValueObj
ComplexOrderValue	MultipleFrequencyGridValueObj

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

5.9.11 IsNoTaskFieldObj Class

This class represents the IsNoTask field.

Table 5.20: IsNoTaskFieldObj Class

Property Name	Data type
FieldValue	Boolean

5.9.12 IsPRNFieldObj Class

This class represents the IsPRN field.

Table 5.21: IsPRNFieldObj Class

Property Name	Data type
FieldValue	Boolean

5.9.13 IVAdditiveCalcFieldObj Class

This class represents the IVAdditiveCalc field.

Table 5.22: IVAdditiveCalcFieldObj Class

Property Name	Data type
FieldValue	IVAdditiveCalcListObj

5.9.14 IVAdditiveFieldObj Class

This class represents the IVAdditive field.

Table 5.23: IVAdditiveFieldObj Class

Property Name	Data type
FieldValue	IVAdditiveListObj

5.9.15 MultiOrderCheckboxFieldObj Class

This class represents the MultiOrderCheckbox field.

Table 5.24: MultiOrderCheckboxFieldObj

Property Name	Data type
FieldValue	MultiOrderItemCollection

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

5.9.16 MultiOrderGridFieldObj Class

This class represents the MultiOrderGrid field.

Table 5.25: MultiOrderGridFieldObj Class

Property Name	Data type
FieldValue	MultiOrderItemCollection

5.9.17 MultiOrderInlineFieldObj Class

This class represents the MultiOrderInline field.

Table 5.26: MultiOrderInlineFieldObj Class

Property Name	Data type
FieldValue	MultiOrderItemCollection

5.9.18 OrderIVRateFieldObj Class

This class represents the OrderIVRate field.

Table 5.27: OrderIVRateObj Class

Property Name	Data type
FieldValue	IVRateValueObj

5.9.19 RequestedTimeFieldObj Class

This class represents the RequestedTime field.

Table 5.28: RequestedTimeFieldObj Class

Property Name	Data type
FieldValue	RequestedTimeFieldObj

5.9.20 SpecialInstructionsFieldObj Class

This class represents the SpecialInstructions field.

Table 5.29: SpecialInstructionsFieldObj Class

Property Name	Data type
FieldValue	String

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

5.9.21 StopAfterFieldObj Class

This class represents the StopAfter field.

Table 5.30: StopAfterFieldObj Class

Property Name	Data type
FieldValue	StopAfterValueObj

5.9.22 StopTimeFieldObj Class

This class represents the StopTime field.

Table 5.31: StopTimeFieldObj Class

Property Name	Data type
FieldValue	String

5.9.23 VariableDoseFieldObj Class

This class represents the VariableDose field.

Table 5.32: VariableDoseFieldObj Class

Property Name	Data type
FieldValue	VariableDoseListObj

5.10 Field Value Types

This section contains the classes that represent complex field value types.

5.10.1 AdditionalVerificationByValueObj Class

This class represents the value of an AdditionalVerificationBy field.

Table 5.33: AdditionalVerificationByValueObj Class

Property Name	DataType
IDCode	String
ProviderIDTypeCode	String

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

5.10.2 BaseSolutionValueObj Class

This class represents the value of a BaseSolution field.

Table 5.34: BaseSolutionValueObj Class

Property Name	DataType
Dose	Double
Name	String
OrderCatalogMasterItemGUID	String
UOM	String

5.10.3 CalculatedDoseValueObj Class

This class represents the value of a CalculatedDose field.

Table 5.35: CalculateDoseValueObj Class

Property Name	DataType
CalcActualDose	Double
CalcDoseMethod	String
CalcOverrideReason	String
CalcRequestedDose	Double
CalcTotalDailyDose	Double
CalcUOM	String
CalcUOMPer	String
DossageLow	Double

5.10.4 CollapsibleWarningValueObj Class

This class represents the value of a CollapsibleWarning field.

Table 5.36: CollapsibleWarningValueObj Class

Property Name	DataType
IsCollapsed	Boolean
MaxLines	Double
WarningMessage	String

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

5.10.5 CombinedMeasurementsValueObj Class

This class represents the value of a CombinedMeasurements field.

Table 5.37: CombinedMeasurementsValueObj Class

Property Name	DataType
BMI	Double
BSA	Double
Height	Double
Weight	Double
WeightType	String

5.10.6 ComplexDosingGridValueObj Class

This class represents the value of a DosageLow field or CalculatedDose field if the complex dosing option is used.

The ComplexDosingGridValueObj parent class is FormContextBaseObj.

This class allows object creation and provides the public constructor with DosingOption and ConfigAllowRowAddRemove as parameters.

```
public ComplexDosingGridValueObj ( System.String dosingoption ,
System.Boolean bConfigAllowRowAddRemove ).
```

The ComplexDosingGridValueObj class also gives Get/Set methods for all properties

Table 5.38: ComplexDosingGridValueObj Class

Property Name	Data type
DosingOption	String
ConfigAllowRowAddRemove	Boolean
AllowRowAddRemove	Boolean
Rows	ComplexDosingGridRowListObj
Cols	ComplexDosingGridColListObj

5.10.7 ComplexDosingGridColListObj Class

This class represents the value of the columns in the Complex Dosing Grid field. It is a list of ComplexDosingGridColValueObj items.

The ComplexDosingGridColListObj parent class is FormContextBaseListObj. This class Implements the IEnumerable, and IEnumerator interfaces. It also provides the ability to iterate through the column values in the list.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

This class allows finding a particular column value by DataItemCode. It also provides Add() and Remove() methods for list item manipulations.

Note: This object is read-only and the complex order grid columns cannot be removed/added using an MLM or called program.

Table 5.39: ComplexDosingGridColListObj Class

Property Name	Data type
Count	Returns the number of items in the list.

5.10.8 ComplexDosingGridColValueObj Class

This class represents the value of a column in the Complex Dosing Grid field.

The ComplexDosingGridColValueObj parent class is FormContextBaseObj.

The CustomFormApp interface does not support adding new column(s) to the grid configuration during runtime. It also provides Get methods for all read-only properties.

Table 5.40: ComplexDosingGridColValueObj Class

Property Name	Data type
DataItemCode	String
IsVisible	Boolean
IsMandatory	Boolean
IsReadOnly	Boolean

5.10.9 ComplexDosingGridRowListObj Class

This class represents the value of the rows in the Complex Dosing Grid. It is a list of ComplexDosingGridRowValueObj items.

The ComplexDosingGridRowListObj parent class is FormContextBaseListObj.

This class implements the IEnumerable, and IEnumerator interfaces. It gives the ability to iterate through the row values in the list.

This class enables finding a particular row value by SequenceNum or by OrderVariableComponentGUID. It also provides Add() and Remove() methods.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

These Methods are provided:

```
public System.Boolean Add
( SCMLib.CustomApp.ComplexDosingGridRowValueObj newRow )
```

- Add specified row to the Rows list if it is not null.
- If added successfully true is returned. Otherwise false is returned.
- If added successfully the Count is incremented by 1.

```
public System.Boolean Remove ( System.String
orderVariableComponentGUID )
```

- Marked the row item with specified OrderVariableComponentGUID in the Rows list for deletion. If the passed-in GUID string is null or empty, nothing is marked for deletion.
- The successfully deleted row has SequenceNum = -1.
- If successfully marked for deletion, true is returned. Otherwise false is returned.
- If successfully marked for deletion, the Count property is **not** decremented by 1.
- Because a newly instantiated ComplexDosingGridRowValueObj object always has empty OrderVariableComponentGUID string, this method could not be used to remove such object from the list. User must Remove by row object method in order to remove such object.

```
public System.Boolean Remove
( SCMLib.CustomApp.ComplexDosingGridRowValueObj row )
```

- Marked specified row object in the Rows list for deletion. If the passed-in row object is null, nothing is marked for deletion.
- The successfully deleted row has SequenceNum = -1.
- If successfully marked for deletion, true is returned. Otherwise false is returned.
- If successfully marked for deletion, the Count property is **not** decremented by 1.

```
public SCMLib.CustomApp.ComplexDosingGridRowValueObj Find
( System.Double seqNum )
```

- Finds the ComplexDosingGridRowValueObj item with the specified sequence number from the Rows list.
- If not found null is returned.
- If multiple rows have the same sequence number, the first row found is returned.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

```
public SCMLib.CustomApp.ComplexDosingGridRowValueObj Find
( System.String orderVariableComponentGUID )
```

- Finds the ComplexDosingGridRowValueObj item with the specified OrderVariableComponentGUID from the Rows list.
- If not found null is returned.

Table 5.41: ComplexDosingGridRowListObj Class

Property Name	Data type
Count	Returns the number of items in the list.

5.10.10 ComplexDosingGridRowValueObj Class

This class represents the value of a row in the Complex Dosing Grid field.

The ComplexDosingGridRowValueObj parent class is FormContextBaseObj.

This class allows object creation and provides a public default constructor.

```
public ComplexDosingGridRowValueObj ()
```

The newly created object has an empty string for OrderVariableComponentGUID.

Table 5.42: ComplexDosingGridRowValueObj Class (1 of 2)

Property Name	Data type
SequenceNum	Double **Note:** SequenceNum value must be a whole number.
OrderVariableComponentGUID (get only)	String
Cells	ComplexDosingGridCellListObj
DataItemCode1	String
DataItemValue1	String
DataItemCode2	String
DataItemValue2	String
DataItemCode3	String
DataItemValue3	String
DataItemCode4	String
DataItemValue4	String
DataItemCode5	String

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

Table 5.42: ComplexDosingGridRowValueObj Class (2 of 2)

Property Name	Data type
DataItemValue5	String

5.10.11 ComplexDosingGridCellListObj Class

This class represents the value of the cells in each row in the Complex Dosing Grid. It is a list of ComplexDosingGridCellValueObj items.

The ComplexDosingGridCellListObj parent class is FormContextBaseListObj.

This class provides the ability to iterate through the cell values in the list.It also allows finding a particular cell value by DataItemCode and uses Add() and Remove() methods.

These Methods are provided:

```
public System.Boolean Add
( SCMLib.CustomApp.ComplexDosingGridCellValueObj newCell )
```

- Add specified cell to the Cells list if it is not null.
- If added successfully true is returned. Otherwise false is returned.
- If added successfully the Count is incremented by 1.

```
public System.Boolean Remove
( System.String DataItemCode )
```

- Marked the cell item with specified DataItemCode in the Cells list for deletion. If the passed-in DataItemCode string is null or empty, nothing is marked for deletion.
- The successfully deleted cell item has DataItemCode = null.
- If successfully marked for deletion, true is returned. Otherwise false is returned.
- If successfully marked for deletion, the Count property is **not** decremented by 1.
- Note that Remove () method should only be used for **new** cells added for a **new** row in the same CustomFormApp running session. For existing cells in the grid, users cannot remove the cell from the grid OR clear the cell value by Remove () method.

```
public System.Boolean Remove
( SCMLib.CustomApp.ComplexDosingGridCellValueObj cell )
```

- Marked specified cell object in the Cells list for deletion. If the passed-in cell object is null, nothing is marked for deletion.
- The successfully deleted cell item has DataItemCode = null.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

- If successfully marked for deletion, true is returned. Otherwise false is returned.
- If successfully marked for deletion the Count property is **not** decremented by 1.
- Note that Remove () method should only be used for **new** cells added for a **new** row in the same CustomFormApp running session. For existing cells in the grid, users cannot remove the cell from the grid OR clear the cell value by Remove () method.

```
public SCMLib.CustomApp.ComplexDosingGridCellValueObj Find
( System.String dataItemCode )
```

- Finds the ComplexDosingGridCellValueObj item with the specified DataItemCode from the Cells list.
- If not found null is returned.

Table 5.43: ComplexDosingGridCellListObj Class

Property Name	Data type
Count	Returns the number of items in the list

5.10.12 ComplexDosingGridCellValueObj Class

This class represents the value of a row cell in the Complex Dosing Grid field.

The ComplexDosingGridCellValueObj parent class is FormContextBaseObj.

It allows object creation, and uses a public constructor with DataItemCode and IsReadOnly as parameters.

```
public ComplexDosingGridCellValueObj ( System.String dataitemcode
, System.Boolean bIsReadOnly )
```

This class also provides Get/Set methods for all properties.

Table 5.44: ComplexDosingGridCellValueObj Class

Property Name	Data type
DataItemCode	String
ItemValue	Object
IsReadOnly	Boolean
IsUpdateable	Boolean

Note: Set method for ItemValue is type specific when it is instantiated.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

Here is a list of support columns in complex order grid in SCM and their corresponding ItemValue types.

Table 5.45: Support Columns and ItemValue Types

DataItemCode value	ItemValue property type
DosageLow	Double
DosageHigh	Double
CalculatedDose	CalculatedDoseValueObj
OrderRouteCode	String
OrderRouteModifierCode	String
Uom	String
CalcActualDose	Double
CalcRequestedDose	Double
CalcUOMPer	String
CalcTotalDailyDose	Double
CalcInfo	String
FrequencyCode	FrequencyCodeValueObj
StopAfter	StopAfterValueObj
RequestedDate	String
RequestedTime	EnhancedTimeValueObj
StopDate	String
StopTime	String
IsPRN	Boolean
PRNReasonText	String

SCMLib.Common.SCMException is thrown with this message if an object of incorrect type is assigned to the ItemValue property.

"A called program on this form is incorrectly attempting to set the value of a column in the dosing option grid.
Details of the error are shown:

The ItemValue data type being assigned to the <DataItemCode> cell is not valid."

This Method is provided:

```
public SCMLib.CustomApp.ComplexDosingGridCellValueObj Clone ()
```

- Creates a deep copy of the existing cell object.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

Note: Currently there is a problem with this method for copying cells with DataItemCode equalling "StopAfter," "FrequencyCode," "RequestedTime," or "CalculatedDose." A shallow copy is performed instead of a deep copy on "ItemValue" property. This problem will be fixed in a future release.

5.10.13 ConditionalOrderValueObj Class

This class represents the value of a ConditionalOrder field.

Table 5.46: ConditionalOrderValueObj Class

Property Name	DataType
IsConditional	Boolean
Number	Double
Reason	String

5.10.14 CreatinineClearanceValueObj Class

This class represents the value of a CreatinineClearance field.

Table 5.47: CreatinineClearanceValueObj Class

Property Name	DataType
CreatinineClearance	Double
Formula	String
IsActual	Boolean
SerumCreatinine	Double

5.10.15 EnhancedTimeValueObj Class

This class represents the value of a RequestedTime field.

Table 5.48: EnhancedTimeValueObj Class

Property Name	DataType
ReqTimeCode	String
ReqTimeEventCode	String
ReqTimeEventModifier	String
ReqTimeUOM	String
ReqTimeValue	String

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

5.10.16 FrequencyCodeValueObj Class

This class represents the value of a FrequencyCode field.

Table 5.49: FrequencyCodeValueObj Class

Property Name	DataType
FreqEventCode	String
FreqEventModifier	String
FreqFromTime	Double
FreqToTime	Double
FrequencySummary	String
FreqUOM	String

5.10.17 IVAdditiveCalcListObj Class

This class represents the value of an IVAdditiveCalc field. It provides the ability to iterate through the additives in the list, as well as to find a particular additive.

5.10.18 IVAdditiveCalcValueObj Class

This class represents an additive in an IVAdditiveCalc field

Table 5.50: IVAdditiveCalcValueObj Class

Property Name	DataType
CalcActualDose	Double
CalcOverrideReason	String
CalcRequestedDose	Double
CalcTotalDailyDose	Double
CalcUOMPer	String

5.10.19 IVAdditiveListObj Class

This class represents the value of an IVAdditive field. It is a list of IVAdditiveValueObj items.

The Find() method can be used to locate a particular additive by name in the list.

```
IVAdditiveValueObj DrugA = additiveList.Find("Drug A");
if ( null != DrugA )
{
```


This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

```
    // Update this additive
    DrugA.Dose = 10;
    DrugA.UOM = "mg";
}
```

Figure 5.25: IVAdditiveListObj Class Using C#

```
Dim DrugB As IVAdditiveValueObj = additiveList.Find("Drug B")
If Not DrugB Is Nothing Then
    ' update this additive
    DrugB.Dose = 20
    DrugB.UOM = "mg"
End If
```

Figure 5.26: IVAdditiveListObj Class Using VB

The Add() method can be used to add an additive to the field.

The Remove() method can be used to remove an additive from the field.

5.10.20 IVAdditiveValueObj Class

This class represents the value of a single additive in the IVAdditive field.

Table 5.51: IVAdditiveValueObj Class

Property Name	DataType
Dose	Double
Name	String
OrderCatalogMasterItemGUID	String
UOM	String

The Clone() method can be used to create a new additive based on another. It performs a deep copy on the existing IVAdditiveValueObj object.

```
Dim DrugA As IVAdditiveValueObj = additiveList.Find("Drug A")
Dim DrugB As IVAdditiveValueObj = additiveList.Find("Drug B")
If DrugB Is Nothing Then
    If Not DrugA Is Nothing Then
        ' create a new one by cloning an existing one
        DrugB = DrugA.Clone
        DrugB.Name = "Drug B"
        DrugB.Dose = DrugA.Dose + 10
        DrugB.UOM = "mg"
        DrugB.OrderCatalogMasterItemGUID = "8000001005073001"
        ' Now add it to the list
        additiveList.Add(DrugB)
    End If
End If
```

Figure 5.27: IVAdditiveValueObj Class Using C#

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

Another way to create an additive is to instantiate one directly.

```
IVAdditiveValueObj DrugB = new IVAdditiveValueObj();
DrugB.Name = "Drug B";
DrugB.Dose = 10;
DrugB.UOM = "mg";
DrugB.OrderCatalogMasterItemGUID = "8000001005073001";

additiveList.Add( DrugB );
```

Figure 5.28: IVAdditiveValueObj Class Using VB

5.10.21 IVRateValueObj Class

This class represents the value of an OrderIVRate field.

Table 5.52: IVRateValueObj Class

Property Name	DataType
Amount	Double
UOM	String
IsKVO	Boolean
KVODisplayed (get-only)	Boolean

5.10.22 MultiOrderItemCollection Class

This class represents the value of a MultiOrderGrid, MultiOrderCheckbox, or MultiOrderInline field. It is a collection of MultiOrderItemValueObj objects.

The Find() method provides access to a specific order in the collection.

```
FormFieldObj field = form.FindField("MultiOrderInline");
MultiOrderInlineFieldObj field =
form.Fields.MultiOrderInlineField;
if ( null != field )
{
    MultiOrderItemCollection items = field.Value as
MultiOrderItemCollection;
    if ( null != items )
    {
        MultiOrderItemValueObj FullFluids = items.Find("Full
Fluids");
        if ( null != FullFluids )
        {
            // make sure it is selected
            FullFluids.IsSelected = true;
        }
    }
}
```

Figure 5.29: MultiOrderItemCollection Class Using C#

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

```
Dim field As FormFieldObj = form.FindField("MultiOrderInline")
If Not field Is Nothing Then
    Dim items As MultiOrderItemCollection = field.Value
    If Not items Is Nothing Then
       Dim FullFluids As MultiOrderItemValueObj = items.Find("Full
Fluids")
        If Not FullFluids Is Nothing Then
            FullFluids.IsSelected = False
        End If
    End If
End If
```

Figure 5.30: MultiOrderItemCollection Class Using VB

5.10.23 MultiOrderItemValueObj Class

This class represents an order item in a MultiOrderGrid, MultiOrderCheckbox, and MultiOrderInline control.

Table 5.53: MultiOrderItemValueObj Class

Property Name	DataType
IsReadOnly	Boolean
IsSelected	Boolean
Name	String
OrderCatalogMasterItemGUID	String

5.10.24 MultipleFrequencyGridValueObj class

This class represents the value of a Frequency field if multiple frequency dosing option is used.

The MultipleFrequencyGridValueObj parent class is FormContextBaseObj.

Allows object creation. Provides public constructor with DosingOption and AllowRowAddRemove as parameters.

```
public MultipleFrequencyGridValueObj ( System.String dosingoption
, System.Boolean bAllowRowAddRemove )
```

ConfigAllowRowAddRemove is always true for multiple frequency grid. This class also uses Get/Set methods for all properties.

Table 5.54: MultipleFrequencyGridValueObj Class (1 of 2)

Property Name	Data type
DosingOption	String
ConfigAllowRowAddRemove	Boolean
AllowRowAddRemove	Boolean

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

Table 5.54: MultipleFrequencyGridValueObj Class (2 of 2)

Property Name	Data type
Rows	MultipleFrequencyGridsRowListObj
Cols	ComplexDosingGridColListObj

5.10.25 MultipleFrequencyGridRowListObj Class

This class represents the value of a multiple frequencies grid field. It is a list of MultipleFrequencyGridRowValueObj items.

The MultipleFrequencyGridRowListObj parent class is FormContextBaseListObj.

This class implements the IEnumerable, and IEnumerator interfaces. It provides the ability to iterate through the frequency values in the list.

Allows finding a particular frequency value by OrderVariableComponentGUID or SequenceNum. This class uses Add() and Remove() methods.

These Methods are provided:

```
public System.Boolean Add
( SCMLib.CustomApp.MultipleFrequencyGridRowValueObj newRow )
```

- Add specified newRow to the Rows list if it is not null
- If added successfully true is returned. Otherwise false is returned.
- If added successfully the Count is incremented by 1.

```
public System.Boolean Remove
( System.String orderVariableComponentGUID )
```

- Marked the row item with specified OrderVariableComponentGUID in the Rows list for deletion. If the passed-in GUID string is null or empty, nothing is marked for deletion.
- The successfully deleted row has SequenceNum = -1.
- If successfully marked for deletion, true is returned. Otherwise false is returned.
- If successfully marked for deletion the Count property is **not** decremented by 1.
- Because a newly instantiated MultipleFrequencyGridRowValueObj object always has empty OrderVariableComponentGUID string, this method could not be used to remove such object from the list. User must Remove by row object method in order to remove such object.

```
public System.Boolean Remove
( SCMLib.CustomApp.MultipleFrequencyGridRowValueObj row )
```


This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

- Marked specified row object in the Rows list for deletion. If the passed-in row object is null, nothing is marked for deletion.
- The successfully deleted row has SequenceNum = -1.
- If successfully marked for deletion, true is returned. Otherwise false is returned.
- If successfully marked for deletion the Count property is **not** decremented by 1.

```
public SCMLib.CustomApp.MultipleFrequencyGridRowValueObj Find
( System.Double seqNum )
```

- Finds the MultipleFrequencyGridRowValueObj item with the specified sequence number from the Rows list.
- If not found null is returned.
- If multiple rows have the same sequence number, the first row found is returned.

```
public SCMLib.CustomApp.MultipleFrequencyGridRowValueObj Find
( System.String orderVariableComponentGUID )
```

- Finds the MultipleFrequencyGridRowValueObj item with the specified OrderVariableComponentGUID from the Rows list.
- If not found null is returned.

Table 5.55: MultipleFrequencyGridRowListObj Class

Property Name	Data type
Count	Returns the number of items in the list.

5.10.26 MultipleFrequencyGridRowValueObj Class

This class represents a frequency defined in each row in the Multiple Frequency Grid.

The MultipleFrequencyGridRowValueObj parent class is FormContextBaseObj.

This class allows object creation and provides a public default constructor.

```
public MultipleFrequencyGridRowValueObj ()
```


This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

The newly created object has an empty string for OrderVariableComponentGUID. This class also uses Get/Set methods for all properties.

Table 5.56: MultipleFrequencyGridRowValueObj Class

Property Name	Data type
SequenceNum	Double **Note:** SequenceNum value must be whole number.
OrderVariableComponentGUID (get only)	String
FrequencySummary	String
FreqFromTime	Double
FreqToTime	Double
FreqUOM	String
FreqEventCode	String
FreqEventModifier	String
IsFreqReadOnly	Boolean
IsFreqUpdateable	Boolean
StopAfterNumber	Double
StopAfterOption	String
IsStopAfterReadOnly	Boolean
IsStopAfterUpdateable	Boolean
DataItemCode1	String
DataItemValue1	String
DataItemCode2	String
DataItemValue2	String
DataItemCode3	String
DataItemValue3	String
DataItemCode4	String
DataItemValue4	String
DataItemCode5	String
DataItemValue5	String

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

5.10.27 StopAfterValueObj Class

This class represents the value for a StopAfter field.

Table 5.57: StopAfterValueObj Class

Property Name	DataType
Option	String
Number	Double

5.10.28 VariableDoseListObj Class

This class represents the value of a VariableDose field. It is a list of VariableDoseValueObj items.

5.10.29 VariableDoseValueObj Class

This class represents the value for a variable dose field.

Table 5.58: VariableDoseValueObj Class

Property Name	DataType
To	String
Condition	String
DoseInstructions	String
From	String
ItemID	String
UOM	String

5.10.30 ClientInfoObj Class

The ClientInfoObj helper object contains context information about the current patient. This class is primarily designed for use by MLMs only, because they cannot obtain this information another way. Custom form applications are free to use this class as well.

Eclipsys recommends that patient context should be obtained using the CustomContextObj and ClientPObj classes exposed from SCMLib.

Table 5.59: ClientInfoObj Class (1 of 3)

Property Name	DataType
Active	Boolean
ApplicSource	String

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

Table 5.59: ClientInfoObj Class (2 of 3)

Property Name	DataType
Backup	Object
BirthDayNum	Integer
BirthMonthNum	Integer
BirthYearNum	Integer
Build	Integer
CityOfBirth	String
CountryDvsnOfBirth	String
CountryOfBirth	String
CreatedBy	String
CreatedWhen	Date
DeceasedDtm	Date
DisplayName	String
FirstName	String
GenderCode	String
GenderTypeIntCode	String
GUID	String
HouseholdGUID	String
IDCode	String
IsMaterialized	Boolean
IsPermanent	Boolean
IsTemporary	Boolean
IsVIP	Boolean
LanguageCode	String
LastName	String
MaritalStatusCode	String
MiddleName	String
NAME	String
OccupationCode	String
PrivacyStatusCode	String
RaceCode	String

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

Table 5.59: ClientInfoObj Class (3 of 3)

Property Name	DataType
ReligionCode	String
SecurityLevel	Integer
Title	String
TouchedBy	String
TouchedWhen	Date

5.11 External Form Processing API

While the CalledProgram functionality is supported, the Custom Form Application API replaces it.

This section describes the CalledProgram functionality that allows a VB or C# program to be called from a button on the Order Entry Form.

Note: The CalledProgram functionality is still supported in 5.0 XA™ but with the introduction of the Custom Form Application API, Eclipsys might not support it in future releases.

5.11.1 Integrating in a Called Program

To use the Called Program, a UDDI has to be created in Config Tools which is of type Call MLM. When the field is added to the Order Entry Form the additional properties for the field have to be configured for the program to run. The Calling Type property has to be set to Program and the MLM/Program Name has to be set to the assembly (dll) that contains the code to run.

The assembly can be created in either C# or VB. The assembly (dll) must have a public class that implements the IExternalFormProcessing interface. When the button associated with the Called Program is pressed the assembly is loaded and the classes are enumerated to find the one that supports the IExternalFormProcessing interface.

The IExternalFormProcessing interface has only one method, called Process(). This method has one parameter which is a reference to the IExtFormSupport interface. The methods on the IExtFormSupport interface are used to communicate with the fields on the Order Entry Form.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

5.11.2 IExternalFormProcessing Interface

This interface provides the main entry point into an External Form Processor as shown in these code examples.

```
public interface IExternalFormProcessing
```

Figure 5.31: IExternalFormProcessing Using VB

```
public interface IExternalFormProcessing
```

Figure 5.32: IExternalFormProcessing Using C#

```
public gc_interface IExternalFormProcessing
```

Figure 5.33: IExternalFormProcessing Using C++

A least one class in an External Form Processor assembly must implement this interface.

5.11.2.1 Requirements

Table 5.60: IExternalFormProcessing Interface Requirements

Application	Eclipsys Sunrise 3.5 XA
Namespace	SCMLib.PObj
Assembly	SCMLib (in SCMLib.dll)

5.11.3 IExternalFormProcessing.Process Method

This method is the main entry point for External Form Processing as shown in these code examples.

```
Sub Process(
ByVal objExtFormSupport As IExtFormSupport
)
```

Figure 5.34: IExternalFormProcessing.Process Method Using VB

```
void Process(
IExtFormSupport objExtFormSupport
);
```

Figure 5.35: IExternalFormProcessing.Process Method Using C#

```
void Process(
IExtFormSupport* objExtFormSupport
);
```

Figure 5.36: IExternalFormProcessing.Process Method Using C++

5.11.3.1 Parameters

`objExtFormSupport`

An object that implements the IExtFormSupport interface. Used to retrieve and manipulate form data.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

This method must be implemented by all External Form Processors and is the main entry point through which external form processing occurs.

5.11.3.2 Examples

```
Imports System
Imports System.Windows.Forms
Imports SCMLib.PObj

public Class HelloWorld
Implements IExternalFormProcessing

public Sub Process(ByVal objExtFormSupport As IExtFromSupport) _
 Implements IExternalFormProcessing.Process

 MessageBox.Show("Hello, World!")
End Sub
End Class
```

Figure 5.37: IExternalFormProcessing.Process Using VB

```
using System;
using System.Windows.Forms;
using SCMLib.PObj;

public class HelloWorld : IExternalFormProcessing
{
public void Process(IExtFromSupport objExtFormSupport)
{
 MessageBox.Show("Hello, World!");
}
}
```

Figure 5.38: IExternalFormProcessing.Process Using C#

```
using namespace System;
using namespace System::Windows::Forms;
using namespace SCMLib::PObj;

public __gc class HelloWorld : public IExternalFormProcessing
{
public:
void Process(IExtFromSupport objExtFormSupport)
{
 MessageBox.Show("Hello, World!");
}
};
```

Figure 5.39: IExternalFormProcessing.Process Using C++

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

5.11.3.3 Exceptions

Table 5.61: IExternalFormProcessing.Process Exceptions

Exception Type	Condition
SCMException	The code specified can not be found on the form. -or- An error occurred while processing.

5.11.3.4 Requirements

Table 5.62: IExternalFormProcessing.Process Interface Requirements

Application	Eclipsys Sunrise 3.5 XA
Namespace	SCMLib.PObj
Assembly	SCMLib (in SCMLib.dll)

5.11.4 IExtFormSupport Interface

This interface provides access to form field information.

```
public Interface IExtFormSupport
```

Figure 5.40: IExtFormSupport Interface Using VB

```
public interface IExtFormSupport
```

Figure 5.41: ExtFormSupport Interface Using C#

```
public __gc __interface IExtFormSupport
```

Figure 5.42: IExtFormSupport Interface Using C++

External form processors do not implement this interface, but utilize an object that implements this interface.

5.11.4.1 Requirements

Table 5.63: IExtFormSupport Interface Requirements

Application	Eclipsys Sunrise 3.5 XA
Namespace	SCMLib.PObj
Assembly	SCMLib (in SCMLib.dll)

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

5.11.5 IExtFormSupport.GetValue Method

This method retrieves the value of a form field.

```
Function GetValue(
ByVal sCode As String
) As String
```

Figure 5.43: IExtFormSupport.GetValue Method Using VB

```
string GetValue(
string sCode
);
```

Figure 5.44: IExtFormSupport.GetValue Method Using C#

```
String* GetValue(
String* sCode
);
```

Figure 5.45: IExtFormSupport.GetValue Method Using C++

5.11.5.1 Parameters

sCode	The code of the field to retrieve the value for

5.11.5.2 Returns

The value for the form field that sCode represents.

5.11.5.3 Exceptions

Table 5.64: IExtFormSupport.GetValue Method Exceptions

Exception Type	Condition
SCMException	The code specified can not be found on the form. -or- An error occurred while processing.

5.11.5.4 Requirements

Table 5.65: IExtFormSupport.GetValue Method Requirements

Application	Eclipsys Sunrise 3.5 XA
Namespace	SCMLib.PObj
Assembly	SCMLib (in SCMLib.dll)

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

5.11.6 IExtFormSupport.SetValue Method

This method changes the value of a field.

```
Sub SetValue(
ByVal sCode As String,
ByVal sValue As String
)
```

Figure 5.46: IExtFormSupport.SetValue Method Using VB

```
void SetValue(
string sCode,
string sValue
);
```

Figure 5.47: IExtFormSupport.SetValue Method Using C#

```
void SetValue(
String* sCode,
String* sValue
);
```

Figure 5.48: IExtFormSupport.SetValue Method Using C++

5.11.6.1 Parameters

sCode	The code of the field to retrieve the value for.
sValue	The new field value.

5.11.6.2 Exceptions

Table 5.66: IExtFormSupport.SetValue Method Exceptions

Exception Type	Condition
SCMException	The code specified can not be found on the form. -or- An error occurred while processing.

5.11.6.3 Requirements

Table 5.67: IExtFormSupport.SetValue Method Requirements

Application	Eclipsys Sunrise 3.5 XA
Namespace	SCMLib.PObj
Assembly	SCMLib (in SCMLib.dll)

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

5.11.7 IExtFormSupport.IsMandatory Method

This method determines if a field is mandatory.

```
Function IsMandatory(
ByVal sCode As String
) As Boolean
```

Figure 5.49: IExtFormSupport.IsMandatory Method Using VB

```
bool IsMandatory(
string sCode
);
```

Figure 5.50: IExtFormSupport.IsMandatory Method Using C#

```
bool IsMandatory(
String* sCode
);
```

Figure 5.51: IExtFormSupport.IsMandatory Method Using C++

5.11.7.1 Parameters

sCode	The code of the field to retrieve the value for.

5.11.7.2 Returns

The value is true if the field is mandatory, or false otherwise.

5.11.7.3 Exceptions

Table 5.68: IExtFormSupport.IsMandatory Method Exceptions

Exception Type	Condition
SCMException	The code specified can not be found on the form. -or- An error occurred while processing.

5.11.7.4 Requirements

Table 5.69: IExtFormSupport.IsMandatory Method Requirements

Application	Eclipsys Sunrise 3.5 XA
Namespace	SCMLib.PObj
Assembly	SCMLib (in SCMLib.dll)

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

5.11.8 IExtFormSupport.IsVisible Method

This method determines if a field is visible.

```
Function IsVisible(
ByVal sCode As String
) As Boolean
```

Figure 5.52: IExtFormSupport.IsVisible Method Using VB

```
bool IsVisible(
string sCode
);
```

Figure 5.53: IExtFormSupport.IsVisible Method Using C#

```
bool IsVisible(
String* sCode
);
```

Figure 5.54: IExtFormSupport.IsVisible Method Using C++

5.11.8.1 Parameters

sCode	The code of the field to retrieve the value for.

5.11.8.2 Returns

The value is true if the field is visible, or false otherwise.

5.11.8.3 Exceptions

Table 5.70: IExtFormSupport.IsVisible Method Exceptions

Exception Type	Condition
SCMException	The code specified can not be found on the form. -or- An error occurred while processing.

5.11.8.4 Requirements

Table 5.71: IExtFormSupport.IsVisible Method Requirements

Application	Eclipsys Sunrise 3.5 XA
Namespace	SCMLib.PObj
Assembly	SCMLib (in SCMLib.dll)

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

5.11.9 IExtFormSupport.IsReadOnly Method

This method determines if a field is read-only.

```
Function IsReadOnly(
ByVal sCode As String
) As Boolean
```

Figure 5.55: IExtFormSupport.IsReadOnly Method Using VB

```
bool IsReadOnly(
string sCode
);
```

Figure 5.56: IExtFormSupport.IsReadOnly Method Using C#

```
bool IsReadOnly(
String* sCode
);
```

Figure 5.57: IExtFormSupport.IsReadOnly Method Using C++

5.11.9.1 Parameters

sCode	The code of the field to retrieve the value for.

5.11.9.2 Returns

The value is true if the field is not editable, or false otherwise.

5.11.9.3 Exceptions

Table 5.72: IExtFormSupport.IsReadOnly Method Exceptions

Exception Type	Condition
SCMException	The code specified can not be found on the form. -or- An error occurred while processing.

5.11.9.4 Requirements

Table 5.73: IExtFormSupport.IsReadOnly Method Requirements

Application	Eclipsys Sunrise 3.5 XA
Namespace	SCMLib.PObj
Assembly	SCMLib (in SCMLib.dll)

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

5.11.10 IExtFormSupport.GetLabel Method

This method retrieves the label text for a field.

```
Function GetLabel(
ByVal sCode As String
) As String
```

Figure 5.58: IExtFormSupport.GetLabel Method Using VB

```
string GetLabel(
string sCode
);
```

Figure 5.59: IExtFormSupport.GetLabel Method Using C#

```
String* GetLabel(
String* sCode
);
```

Figure 5.60: IExtFormSupport.GetLabel Method Using C++

5.11.10.1 Parameters

sCode	The code of the field to retrieve the value for.

5.11.10.2 Returns

The value is the label text for the form field that sCode represents.

5.11.10.3 Exceptions

Table 5.74: IExtFormSupport.GetLabel Method Exceptions

Exception Type	Condition
SCMException	The code specified can not be found on the form. -or- An error occurred while processing.

5.11.10.4 Requirements

Table 5.75: IExtFormSupport.GetLabel Method Requirements

Application	Eclipsys Sunrise 3.5 XA
Namespace	SCMLib.PObj
Assembly	SCMLib (in SCMLib.dll)

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

CHAPTER 6: EXTENDING THE NOTIFICATION SERVER

Contents

6.1 Overview 93
6.2 Writing Transports 93

6.1 Overview

The Notification Server provides a way of sending notification messages. The underlying components sending the actual notifications are called transports. SCM comes with two transports; one for sending e-mail messages and the other for sending pages. The Notification Server also provides customers with the ability to add their own transport or replace the Eclipsys supplied ones.

6.2 Writing Transports

Eclipsys provides two standard transports with the Notification Server; one for sending e-mail messages using SMTP and the other for sending pages over a standard modem. This section provides instructions on how to write a transport for use with the Notification Server.

6.2.1 Integration

To integrate with the Notification Server, a transport must provide a public class that implements the ISCMTransport interface. This interface implements the methods Initialize() and SendMessage() which are used by the Notification Server.

For example:

```
using System;
using System.Collections.Generic;
using System.Text;
using ObjectsPlusXA.Transports;

namespace CustomTransport.Sample
{
```


This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

```
    public class SampleTransport :
ObjectsPlusXA.Transports.ISCMTransport
    {
        public bool Initialize(string logLevel, string
logExtension)
        {
            return true;
        }

       public TransportStatusCodes SendMessage(string textMessage,
ref string returnMessage)
        {
            return TransportStatusCodes.OK;
        }

    }
}
```

Figure 6.1: Integrating With The Notification Server Using C#

6.2.2 Interface Details

The ISCMTransport interface implements two methods.

6.2.2.1 Initialize

The initialize method is called after the Notification server has started and is used to perform any initialization that the transport requires. This includes opening log files, reading configuration files or initializing any third-party code.

The logLevel parameter is a string that matches up to the accepted log levels that are configured for the Notification Server from the Service Administrator. The expected values are:

- Verbose
- Development
- Error
- Warning

The logLevel indicates how much information is sent to the log file. For Verbose everything that is happening to the data is logged. Warning logs only the events that generate a warning or error. Error logs events that generate an error. Development logs details that are used to track down issues when developing the transport.

The logExtension parameter is a string whose value is dependent on the log extension value configured for the Notification Server from the Service Administrator. The value can be null or a string of numbers that is either a date or a date and time.

If configuring the log extension to be DateTime, this string contains something like "20061027173700", giving the Year, Month, day, hour, minute and second

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

when the log file is created. This string should be appended to the end of the log file's name, for example `CustomTransport_20061027173700.log`.

If an error occurs during initialization this method should return false. This lets the Notification Server know that the transport is cannot start and does not send any messages to it.

6.2.2.2 SendMessage

The SendMessage method is called each time the Notification Server encounters a message for this transport in the queue. The server takes the message as it is stored in the database and passes it in the textMessage parameter.

The transport reads this message, parses and validates the information, then attempts to send the message. When completed, the transport returns a status code and if necessary, an error message. These status codes are outlined in "Return Status" on page 98.

SCM currently supports sending messages via E-mail and Pager using the destination clause in an MLM. There is also an Other destination that is used to send messages through a third transport if required. The application formats each message into XML before passing it to the transports.

When writing a transport it is necessary to handle the format as described in the E-mail, Pager and Other sections. Each section shows the syntax used from an MLM as well as the format of the XML passed to the transport.

6.2.2.3 Email

The MLM Syntax for the Email destination:

```
 MailAddress :=
"Yu.April@eclipsys.com;CC:Smith.Joe@Eclipsys.com;Jones.Peter@eclip
sys.com";
 mail_destination := destination
      { email: reminder, "NEW Order Alert via Email", low, chart,
MailAddress };

write "Testing the sending of e-mail via SMTP protocol" at mail
destination
```

Figure 6.2: E-mail Transport SendMessage Method for MLMs

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

The XML syntax for the Email destination:

```
<EMAIL>
     <TO>Yu.April@eclipsys.com</TO>
     <SUBJECT>NEW Order Alert via Email</SUBJECT>
     <CC>Smith.Joe@Eclipsys.com;Jones.Peter@eclipsys.com</CC>
     <BCC/>
     <FROM/>
     <BODY>Testing the sending of e-mail via SMTP protocol</BODY>
     <PRIORITY>Low</PRIORITY>
     <FORMAT>Text</FORMAT>
</EMAIL>
```

Figure 6.3: E-mail Transport SendMessage Method for XML

Table 6.1: E-mail XML Field Descriptions

XML Field	Description
TO	A semicolon-delimited field containing the e-mail addresses of who the message is to be sent to. This field is mandatory.
SUBJECT	The subject for the e-mail message. This field is optional
CC	A semicolon-delimited field containing the e-mail addresses of who the message is to be carbon copied to. This field is optional.
BCC	Not supported for this release. The field is always empty.
FROM	This field is always empty. The e-mail destination currently does not support this value. The Eclipsys supplied e-mail transport reads this field from the configuration file.
BODY	The text of the message.
PRIORITY	The priority of the mail message. Valid values are High, Low and Normal.
FORMAT	Indicates the body text format. It is always Text.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

6.2.2.4 Pager

The MLM syntax for the Pager destination:

```
 PagerNumber := "9(604)555-3269;5558737";
 pager_destination := destination
      { pager: reminder, "Alert via Pager", low, chart, PagerNumber
};

write “The body of the page as written to by the MLM” at mail
destination
```

Figure 6.4: Pager Transport SendMessage Method for MLMs

The XML syntax for the Pager destination:

```
<PAGER>
     <BODY>The body of the page as written to by the MLM</BODY>
     <NUMBER>9,(604)5553269</NUMBER>
     <PIN>5558737</PIN>
     <TYPE>alpha</TYPE>
</PAGER>
```

Figure 6.5: Pager Transport SendMessage Method for XML

Table 6.2: Pager XML Field Descriptions

XML Field	Description
BODY	The text sent over the pager. This field is optional.
NUMBER	The service provider’s pager access number, including the country code, area code, and phone number, in the format 1(999)5556666. This is a mandatory field.
PIN	The pager ID number. This is a mandatory field.
TYPE	The type of pager call to make. This field is always Alpha.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

6.2.2.5 Other

The MLM syntax for the Other destination:

```
FaxNumber := "6045553269"
other_destination := destination
 { other: reminder, "NEW Order Alert via Fax", low, chart,
FaxNumber };

write “The body of the Fax as written to by the MLM” at Other
destination
```

Other Transport SendMessage Method for MLMs and XML

The XML syntax for the Other destination:

```
<OTHER>
     <BODY>The body of the Fax as written to by the MLM</BODY>
     <NUMBER>6045553269</NUMBER>
     <SUBJECT>New Order Alert via Fax</SUBJECT>
     <PRIORITY>LOW</PRIORITY>
</OTHER>
```

Figure 6.6: Other Transport SendMessage Method for MLMs and XML

Table 6.3: Other XML Field Descriptions

XML Field	Description
BODY	The text to be sent.
NUMBER	A phone number, including the country code, area code, and phone number in the format 1(999)5556666.
SUBJECT	The short message from the destination clause.
PRIORITY	The priority string from the destination clause. This value can be High, Medium, or Low.

6.2.2.6 Return Status

The return value depends on whether the transport was successful sending the message. The TransportStatusCodes enumerator has multiple values.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

The Notification Server uses the return value to help determine the state of the transport. TransportStatusCodes has these allowable values:

```
public enum TransportStatusCodes
   {
       OK = 0,
       Retry = 1,
       Failed = 2,
       Reject = 3,
       Stop = 4
   }
```

Figure 6.7: TransportStatusCodes Values Code

If the transport is unsuccessful in sending the message one of the other four values is returned. The Notification Server uses these values to determine what state the transport is in and how to proceed with sending messages to it.

Table 6.4: TransportStatusCodes Values (1 of 2)

Value	Description
Ok	This value signifies to the Notification Server that the message was sent successfully.
Retry	This value signifies to the Notification Server the message was not sent but it should be re-tried. This value is typically returned if the number is busy. The Notification Server immediately calls the transport again in an attempt to send the message. After a configurable number of re-try attempts, AttemptsBeforeFail, the Notification Server marks the message as failed.
Failed	This value signifies to the Notification Server that the message was not sent but should be re-tried at a later time. After a configurable number of re-try attempts, FailuresBeforeReject, the Notification Server marks the message as rejected and will not try again unless the message status is reset.
Reject	This value signifies to the Notification Server that the message cannot be sent. The Notification Server marks the message as rejected and will not send unless the message's status is reset.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

Table 6.4: TransportStatusCodes Values (2 of 2)

Value	Description
Stop	This value signifies to the Notification Server that there is a problem sending any messages. The Notification Server stops sending any messages to the transport.

If the Notification Server receives too many consecutive failures, configured in StopOnFailure, or a Stop, the server stops sending messages to the transport for a configurable length of time. The length of time is configured in RetryAfterStop.

For all return values other than OK, the returnMessage string is set with a description of the problem. This returned message logs with the transport and is saved in the message queue.

It is recommended that the returned message should not exceed 1000 characters.

6.2.2.7 Installing Transports

The assembly for the transport is copied into the same folder as the Notification Server executable (NotificationServer.exe) on the Notification Server machine. If the transport has a configuration file it also has to be put into this folder.

6.2.2.8 Transport Configuration

After the transport is created it must be configured so that the Notification Server knows about it. Additional information can be found in the Notification Server Configuration Guide.

The configuration information is stored in the Environment Profile in Configuration Tools. Open the Environment profile Maintenance window and expand the Notification Server node to see the Eclipsys supplied e-mail and pager transports.

If creating a new transport for the Other destination then add a new Section/Sub Section with the name Other.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

Under this new node, create each of the configuration entries and set values based on the requirements of the transport.

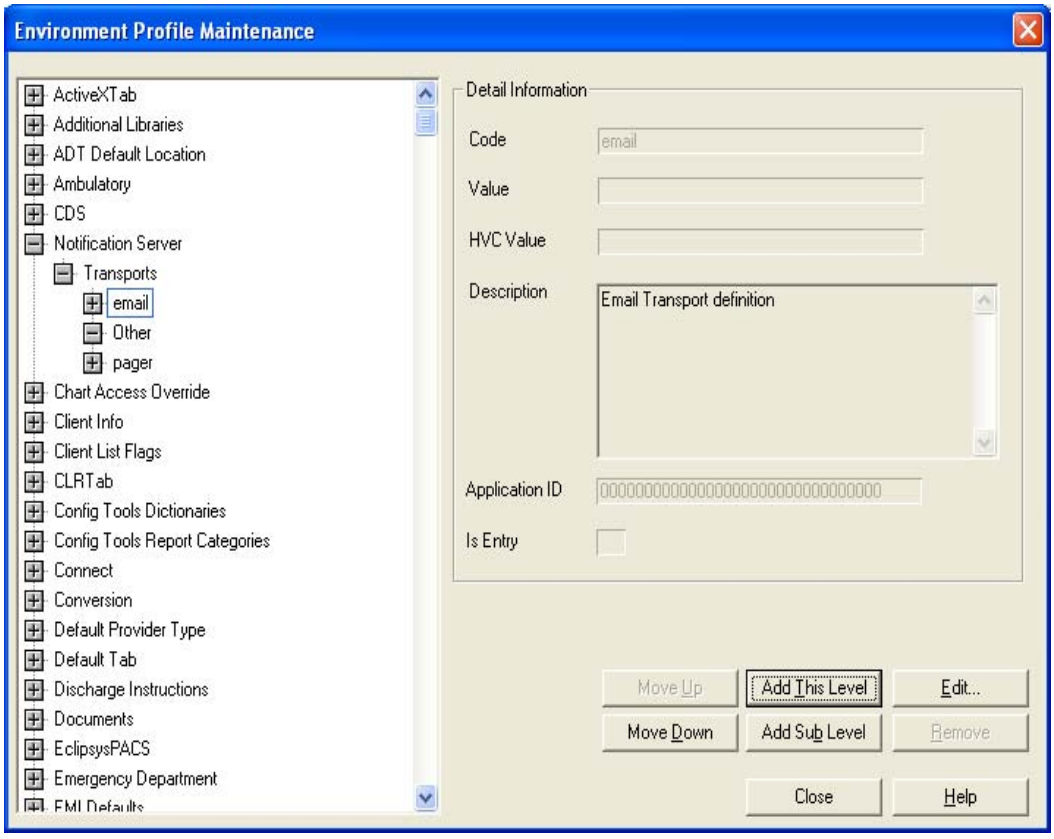


Figure 6.8: Transport Configuration Using the Environment Profile

The Assembly value is the only mandatory entry and the name of the assembly is required. The other values are optional, and if not found, revert to their default settings.

If a transport is created to replace one of the Eclipsys supplied transports then simply change the assembly name. Do not use the same assembly name as those provided by Eclipsys.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

CHAPTER 7: SCMLIB OBJECTS

Contents

7.1 Overview 103
7.2 HVCLogonObj Class 103
7.3 MainFrameObj Class 112
7.4 CustomContextObj Class 115
7.5 HVCDateObj Class 118
7.6 PObj Class 119
7.7 UserPObj Class 120
7.8 ClientPObj Class 125
7.9 ChartPObj Class 134
7.10 ClientVisitPObj Class 135

7.1 Overview

This chapter provides information on the use of objects contained in the SCMLib.dll library. These objects are used for Custom Tabs, custom applications, and custom form applications.

CAUTION



All other objects and namespaces in the SCMLib assembly are currently reserved for Eclipsys use only.

7.2 HVCLogonObj Class

The HVCLogonObj class exposes services and context data related to the current logon session.

```
System.Object
SCMLib.HVCLogon.HVCLogonObj
```

7.2.1 Namespace

```
SCMLib.HVCLogon
```


This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7.2.2 HVCLogonObj.GetConnectionString()

This method returns an ADO .NET compatible connection string that can be used to access the same database that Sunrise XA is currently running against.

7.2.2.1 Syntax

```
public static string GetConnectionString();
```

7.2.2.2 Return Value

The string returned contains database connection information. This string typically uses this syntax:

```
Data Source=;Initial Catalog=;User ID=;PWD=;Workstation
ID=;Application Name=
```

7.2.2.3 Remarks

The password portion of the connection string is encrypted for security reasons.

7.2.2.4 C# Example

```
using System.Data.SqlClient;

string sConnectionString = HVCLogonObj.GetConnectionString();
SqlConnection conn = new SqlConnection( sConnectionString );
try
{
conn.Open();
    // do something
conn.Close();
}
catch( Exception e )
{
MessageBox.Show( e.Message );
}
```

7.2.3 HVCLogonObj.GetCurrentUser()

GetCurrentUser() returns the current logged on user.

7.2.3.1 Syntax

```
public static UserPObj GetCurrentUser();
```

7.2.3.2 Return Value

This method returns a reference to a UserPObj component, which can be used to access addition context information about the user.

7.2.3.3 Remarks

The password portion of the connection string is encrypted for security reasons.

7.2.3.4 C# Example

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

```
UserPObj myUser = HVCLogonObj.GetCurrentUser();
string sIDCode = myUser.IDCode;
```

7.2.4 HVCLogonObj.GetCurrentUserName()

This method returns the display name of the current user.

7.2.4.1 Syntax

```
public static string GetCurrentUserName();
```

7.2.4.2 Return Value

The return value is a string containing the user's display name. It is typically in the form:

```
<last name>, <given names>
```

7.2.4.3 C# Example

```
string sName = HVCLogonObj.GetCurrentUserName();
```

7.2.5 HVCLogonObj.GetSessionID()

This method returns the current logon session ID

7.2.5.1 Syntax

```
public static int GetSessionID();
```

7.2.5.2 Return Value

The return value is the session ID value generated for the current logon session, if applicable. It is zero otherwise.

7.2.5.3 C# Example

```
int nSessionID = HVCLogonObj.GetSessionID();
```

7.2.6 HVCLogonObj.GetEnvironmentProfileString()

This method returns a string value from the environment profile.

7.2.6.1 Syntax

```
public static string GetEnvironmentProfileString( string sSection,
string sKey);

public static string GetEnvironmentProfileString( string sSection,
string sKey, string sDefault);
```

7.2.6.2 Return Value

The return value is the string value of the specified environment profile entry.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7.2.6.3 Remarks

If a default value is provided, it is returned if the entry is not found. If no default value is provides, an empty string is returned if the entry is not found.

7.2.6.4 C# Example

```
string sTabCaption = HVCLogonObj.GetEnvironmentProfileString(
"CLRTab", "TabCaption0" );
```

7.2.7 HVCLogonObj.GetEnvironmentProfileInt()

This method returns an integer value from the environment profile.

7.2.7.1 Syntax

```
public static int GetEnvironmentProfileInt( string sSection,
string sKey);
public static int GetEnvironmentProfileInt( string sSection,
string sKey, string sDefault);
```

7.2.7.2 Return Value

The return value is the integer value of the specified environment profile entry.

7.2.7.3 Remarks

An exception is noted if the specified entry cannot be converted to an integer value.

7.2.7.4 C# Example

```
int nViewLibraryCount = HVCLogonObj.GetEnvironmentProfileInt(
"View Library", "ViewLibraryCount");
```

7.2.8 HVCLogonObj.GetEnvWorkstationProfileString()

This method returns the string value of a workstation profile entry.

7.2.8.1 Syntax

```
public static string GetEnvWorkstationProfileString( string
sSection, string sKey);

public static string GetEnvWorkstationProfileString( string
sSection, string sKey, string sDefault);
```

7.2.8.2 Return Value

The return value is the string value of the specified workstation profile entry.

7.2.8.3 Remarks

This is similar to the GetEnvironmentProfileString() method, except it is for the user's workstation.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7.2.8.4 C# Example

```
string sCheckDuplicate =
HVCLogonObj.GetEnvWorkstationProfileString( "Orders",
"CheckDuplicates" );
```

7.2.9 HVCLogonObj.GetEnvWorkstationProfileInt()

This method returns the integer value of the specified workstation profile entry.

7.2.9.1 Syntax

```
public static int GetEnvWorkstationProfileInt( string sSection,
string sKey);

public static int GetEnvWorkstationProfileInt( string sSection,
string sKey, string sDefault);
```

7.2.9.2 Return Value

The return value is the integer value of the specified workstation profile entry.

7.2.9.3 Remarks

This is similar to the GetEnvironmentProfileInt() method, except it is for the user's workstation.

7.2.9.4 C# Example

```
int nViewLibraryCount = HVCLogonObj.GetEnvWorkstationProfileInt(
"Orders", "BackdateMaxDays");
```

7.2.10 HVCLogonObj.GetBuild()

This method returns the major build number.

7.2.10.1 Syntax

```
public static int GetBuild();
```

7.2.10.2 Return Value

The return value is the integer value of the major build number of SCM.

7.2.10.3 Remarks

Should use GetFullBuild() if the value has to be saved to the database.

7.2.10.4 C# Example

```
int nBuild = HVCLogonObj.GetBuild();
```

7.2.11 HVCLogonObj.GetWorkstationGUID()

This method returns the GUID of the current workstation.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7.2.11.1 Syntax

```
public static string GetWorkstationGUID();
```

7.2.11.2 Return Value

The return value is the string value of the workstation GUID.

7.2.11.3 C# Example

```
string sWorkstationGUID = HVCLogonObj.GetWorkstationGUID();
```

7.2.12 HVCLogonObj.GetWorkstationIDCode()

This method returns the display name of the current workstation.

7.2.12.1 Syntax

```
public static string GetWorkstationIDCode();
```

7.2.12.2 Return Value

The return value is the display name of the workstation used to logon to SCM.

7.2.12.3 C# Example

```
string sWorkstationIDCode = HVCLogonObj.GetWorkstationIDCode();
```

7.2.13 HVCLogonObj.CheckRightsByID()

Checks whether the user has the specified application right.

7.2.13.1 Syntax

```
public static bool CheckRightsByID( string sRightGUID );
```

7.2.13.2 Return Value

This method returns true if the user has the SCM application right specified by the GUID; false, otherwise.

7.2.13.3 Remarks

The input to this method is the GUID of the right, and not the display name of the right.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7.2.13.4 C# Example

```
string sRight1 = "4000001000104001";  // valid rights GUID
string sRight2 = "1234";              // invalid rights GUID
bool bHasRight1 = HVCLogonObj.CheckRightsByID(sRight1);
bool bHasRight2 = HVCLogonObj.CheckRightsByID(sRight2);
if (bHasRight1)
{
MessageBox.Show("Has right1");
}
else
{
MessageBox.Show("Does not have right1");
}

if (bHasRight2)
{
MessageBox.Show("Has right2");
}
else
{
MessageBox.Show("Does not have right2");
}
```

7.2.14 HVCLogonObj.GetApplicationName()

This method returns the display name of the running application.

7.2.14.1 Syntax

```
public static string GetApplicationName();
```

7.2.14.2 Return Value

The return value is the display name of the running application. This is determined by the /m mode command line parameter passed to hvc.exe.

```
“Sunrise Clinical Manager”
“Sunrise ED Manager”
“Sunrise Ambulatory Care Manager”
```

7.2.14.3 Remarks

Use MainFrameObj.GetApplicationMode() instead if you have to know what mode hvc.exe is running in. That is easier than parsing the application display name.

7.2.14.4 C# Example

```
string sApplicationName = HVCLogonObj.GetApplicationName();
```

7.2.15 HVCLogonObj.GetColorPreference()

This method returns the specified SCM color. This is important for maintaining a consistent color schema across all SCM tabs and windows.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7.2.15.1 Syntax

```
public static Color GetColorPreference( ColorTypeEnum enumType );
```

7.2.15.2 Return Value

This value returns the System.Drawing.Color defined by the ColorTypeEnum parameter.

```
public enum ColorTypeEnum
{
    SysColor_SCROLLBAR          = 0,
    SysColor_BACKGROUND         = 1,
    SysColor_ACTIVECAPTION      = 2,
    SysColor_INACTIVECAPTION    = 3,
    SysColor_MENU               = 4,
    SysColor_WINDOW             = 5,
    SysColor_WINDOWFRAME        = 6,
    SysColor_MENUTEXT           = 7,
    SysColor_WINDOWTEXT         = 8,
    SysColor_CAPTIONTEXT        = 9,
    SysColor_ACTIVEBORDER       = 10,
    SysColor_INACTIVEBORDER     = 11,
    SysColor_APPWORKSPACE       = 12,
    SysColor_HIGHLIGHT          = 13,
    SysColor_HIGHLIGHTTEXT      = 14,
    SysColor_BTNFACE            = 15,
    SysColor_BTNSHADOW          = 16,
    SysColor_GRAYTEXT           = 17,
    SysColor_BTNTEXT            = 18,
    SysColor_INACTIVECAPTIONTEXT = 19,
    SysColor_BTNHIGHLIGHT       = 20,
    SysColor_3DDKSHADOW         = 21,
    SysColor_3DLIGHT            = 22,
    SysColor_INFOTEXT           = 23,
    SysColor_INFOBK             = 24,
    SysColor_HOTLIGHT           = 26,
    SysColor_GRADIENTACTIVECAPTION = 27,
    SysColor_GRADIENTINACTIVECAPTION = 28,
    SysColor_MENUHILIGHT        = 29,
    SysColor_MENUBAR            = 30,
    PriWindowBackground         = APP_SYSCOLOR(0),
    SecWindowBackground         = APP_SYSCOLOR(1),
    DialogCaptionForeground     = APP_SYSCOLOR(2),
    DialogCaptionBackground     = APP_SYSCOLOR(3),
    GroupHeaderForeground       = APP_SYSCOLOR(4),
    GroupHeaderBackground       = APP_SYSCOLOR(5),
    SecWindowGroupBackground    = APP_SYSCOLOR(6),
    GridForeground              = APP_SYSCOLOR(7),
    GridBorder                  = APP_SYSCOLOR(8),
    GridOddRowsBackground       = APP_SYSCOLOR(9),
    GridEvenRowsBackground      = APP_SYSCOLOR(10),
    GridColumnHeaderForeground  = APP_SYSCOLOR(11),
    GridColumnHeaderBackground  = APP_SYSCOLOR(12),
    EditableFieldForeground     = APP_SYSCOLOR(13),
    EditableFieldBackground     = APP_SYSCOLOR(14),
    ReadOnlyFieldForeground     = APP_SYSCOLOR(15),
    ReadOnlyFieldBackground     = APP_SYSCOLOR(16),
```


This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

```
    RequiredFieldIndicator      = APP_SYSCOLOR(17),
    ButtonForeground            = APP_SYSCOLOR(18),
    ButtonBackground            = APP_SYSCOLOR(19),
    CaptionForeground           = APP_SYSCOLOR(20)
};
```

7.2.15.3 Remarks

If the window/control class derives from one of the base classes supplied by SCM, you typically not have to set any color preferences. It can already be handled.

7.2.15.4 C# Example

```
Color myCaptionColor = HVCLogonObj.GetColorPreference(
CaptionForeground);
```

7.2.16 HVCLogonObj.GetDateDisplayFormat()

This method returns the format string for showing a date in SCM.

7.2.16.1 Syntax

```
public static string GetDateDisplayFormat();
```

7.2.16.2 Return Value

The return value is the date format string defined as “DisplayGeneral” in the Environment Profiles.

7.2.16.3 C# Example

```
string sFormat = HVCLogonObj.GetDateDisplayFormat();
// Example: sFormat = “dd-MMM-yyyy”
```

7.2.17 HVCLogonObj.GetDateEntryFormat()

This method returns the date format string for use during data entry.

7.2.17.1 Syntax

```
public static string GetDateEntryFormat();
```

7.2.17.2 Return Value

The return value is the date format string defined as “DateEntryFormat” in the Environment Profiles.

7.2.17.3 C# Example

```
string sFormat = HVCLogonObj.GetDateEntryFormat();
// Example: sFormat = “dd-MMM-yyyy”
```

7.2.18 HVCLogonObj.GetTimeDisplayFormat()

This method returns the format string for a time value in SCM.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7.2.18.1 Syntax

```
public static string GetTimeDisplayFormat();
```

7.2.18.2 Return Value

The return value is the time format string defined as “TimeFormat” in the Environment Profiles.

7.2.18.3 C# Example

```
string sFormat = HVCLogonObj.GetTimeDisplayFormat();
// Example: sFormat = “HH:mm”
```

7.3 MainFrameObj Class

The MainFrameObj class exposes selected services and context data from the SCM MainFrame class. This is the main window that contains the toolbar, patient header, and tabs.

```
System.Object
SCMLib.Common.MainFrameObj
```

7.3.1 Namespace

```
SCMLib.Common
```

7.3.2 MainFrameObj.GetVersion()

This method returns the version of SCM.

7.3.2.1 Syntax

```
public static string GetVersion();
```

7.3.2.2 Return Value

The return value is the display name of the SCM version that is running.

7.3.2.3 Remarks

This is the same information that appears in the “Help / About Sunrise Clinical Manager” window.

7.3.2.4 C# Example

```
string sVersion = MainFrameObj.GetVersion();
```

7.3.3 MainFrameObj.GetClientVersion()

This method returns the client version of SCM.

7.3.3.1 Syntax

```
public static string GetClientVersion();
```


This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7.3.3.2 Return Value

The return value is the client version of SCM, which is the full build number.

7.3.3.3 Remarks

This is the same information that appears in the "Help / About Sunrise Clinical Manager" window.

7.3.3.4 C# Example

```
string sClientVersion = MainFrameObj.GetClientVersion();
```

7.3.4 MainFrameObj.GetDatabaseVersion()

This method returns the version of SQL Server that SCM is running against.

7.3.4.1 Syntax

```
public static string GetDatabaseVersion();
```

7.3.4.2 Return Value

The return value is the version number of Microsoft SQL Server.

7.3.4.3 Remarks

This is the same information that appears in the "Help / About Sunrise Clinical Manager" window.

7.3.4.4 C# Example

```
string sDatabaseVersion = MainFrameObj.GetDatabaseVersion();
```

7.3.5 MainFrameObj.GetCurrentDatabase()

This method returns the name of the SCM database.

7.3.5.1 Syntax

```
public static string GetCurrentDatabase();
```

7.3.5.2 Return Value

The return value is the name of the SCM database.

7.3.5.3 Remarks

This is the same information that appears in the "Help / About Sunrise Clinical Manager" window.

7.3.5.4 C# Example

```
string sCurrentDatabase = MainFrameObj.GetCurrentDatabase();
```


This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7.3.6 MainFrameObj.GetOperatingSystem()

This method returns the name of the operating system.

7.3.6.1 Syntax

```
public static string GetOperatingSystem();
```

7.3.6.2 Return Value

The return value is the name of the operating system.

7.3.6.3 Remarks

This is the same information that appears in the "Help / About Sunrise Clinical Manager" window.

7.3.6.4 C# Example

```
string sOperatingSystem = MainFrameObj.GetOperatingSystem();
```

7.3.7 MainFrameObj.Handle

This method returns the window handle of the main SCM mainframe.

7.3.7.1 Syntax

```
public static IntPtr Handle [get];
```

7.3.7.2 Return Value

The return value is the window handle of the main SCM mainframe.

7.3.7.3 Remarks

This is for internal use only.

7.3.7.4 C# Example

```
IntPtr pHandle = MainFrameObj.Handle;
```

7.3.8 MainFrameObj.GetApplicationMode()

This method returns the application mode of hvc.exe.

7.3.8.1 Syntax

```
public static string GetApplicationMode();
```

7.3.8.2 Return Value

The return value is the application mode string, which corresponds to the /m mode command line parameter passes to hvc.exe. It is blank for SCM, because that is the default mode.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7.3.8.3 C# Example

```
string sApplicationMode = MainFrameObj.GetApplicationMode();
```

7.4 CustomContextObj Class

This is the key component in SCM for context management. It maintains the current patient context, current user context, and custom context.

```
System.Object
SCMLib.Context.CustomContextObj
```

7.4.1 Namespace

```
SCMLib.Context
```

7.4.2 CustomContextObj.GetInstance()

Using this method is the only permitted way to access the CustomContextObj component.

7.4.2.1 Syntax

```
public static CustomContextObj GetInstance();
```

7.4.2.2 Return Value

The return value is a reference to the CustomContextObj singleton component.

7.4.2.3 C# Example

```
CustomContextObj cc = CustomContextObj.GetInstance();
string sClientGUID = cc.ClientGUID;
```

7.4.3 CustomContextObj.SetItem()

Place a string value into context

7.4.3.1 Syntax

```
public static void SetItem( string sName, string sValue );
```

7.4.3.2 Remarks

If the sName context item already exists, it is overwritten by the new value. There is no warning or error issued. This is by design.

7.4.3.3 C# Example

```
CustomContextObj cc = CustomContextObj.GetInstance();
String sName = txtContextName.Text;
String sValue = txtContextValue.Text;
cc.SetItem(sName, sValue);
```


This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7.4.4 CustomContextObj.GetItem()

Retrieves an item from context.

7.4.4.1 Syntax

```
public static string GetItem( string sName );
```

7.4.4.2 Return Value

The return value is the context item identified by sName.

7.4.4.3 Remarks

If the context item does not exist, an empty string is returned. No error is issued. This is by design.

7.4.4.4 C# Example

```
CustomContextObj cc = CustomContextObj.GetInstance();
String sName = txtContextName.Text;
String sValue = cc.GetItem(sName);
txtContextValue.Text = sValue;
```

7.4.5 CustomContextObj.RemoveItem()

Remove a specific item from context.

7.4.5.1 Syntax

```
public static void RemoveItem( string sName );
```

7.4.5.2 Remarks

Nothing happens if the context item does not exist. No error is issued. This is by design.

7.4.5.3 C# Example

```
CustomContextObj cc = CustomContextObj.GetInstance();
String sName = txtContextName.Text;
cc.RemoveItem(sName);
```

7.4.6 CustomContextObj.ClientGUID

This method returns the current ClientGUID.

7.4.6.1 Syntax

```
public static string ClientGUID [get];
```

7.4.6.2 Return Value

The return value is the current ClientGUID.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7.4.6.3 Remarks

This value should be retrieved after a patient change notification. It represents the current patient.

7.4.6.4 C# Example

```
CustomContextObj cc = CustomContextObj.GetInstance();
string sClientGUID = cc.ClientGUID;
```

7.4.7 CustomContextObj.ChartGUID

This method returns the current ChartGUID.

7.4.7.1 Syntax

```
public static string ChartGUID [get];
```

7.4.7.2 Return Value

The return value is the current ChartGUID.

7.4.7.3 Remarks

This value should be retrieved after a patient change notification. It represents the current chart.

7.4.7.4 C# Example

```
CustomContextObj cc = CustomContextObj.GetInstance();
string sChartGUID = cc.ChartGUID;
```

7.4.8 CustomContextObj.VisitGUID

This method returns the current VisitGUID.

7.4.8.1 Syntax

```
public static string VisitGUID [get];
```

7.4.8.2 Return Value

The return value is the current VisitGUID.

7.4.8.3 Remarks

This value should be retrieved after a patient change notification. It represents the current client visit.

7.4.8.4 C# Example

```
CustomContextObj cc = CustomContextObj.GetInstance();
string sVisitGUID = cc.VisitGUID;
```


This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7.4.9 CustomContextObj.UserGUID

This method returns the current UserGUID.

7.4.9.1 Syntax

```
public static string UserGUID [get];
```

7.4.9.2 Return Value

The return value is the current UserGUID.

7.4.9.3 Remarks

This value should be retrieved after a user change notification. It represents the current logged on user.

7.4.9.4 C# Example

```
CustomContextObj cc = CustomContextObj.GetInstance();
string sUserGUID = cc.UserGUID;
```

7.4.10 CustomContextObj.GetConnectionString()

This method returns the connection string

7.4.10.1 Syntax

```
public static string GetConnectionString();
```

7.4.10.2 Return Value

The return value is the connection string used to access the SCM database.

7.4.10.3 Remarks

This is the same as calling HVCLogonObj.GetConnectionString(). It is available here for convenience.

7.4.10.4 C# Example

```
CustomContextObj cc = CustomContextObj.GetInstance();
string sConnectionString = cc.GetConnectionString();
```

7.5 HVCDateObj Class

The HVCDateObj helper class exposes selected datetime services.

```
System.Object
SCMLib.Common.HVCDateObj
```

7.5.1 Namespace

```
SCMLib.Common
```


This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7.5.2 HVCDateObj.ServerToday()

This method returns the current datetime value from the server.

7.5.2.1 Syntax

```
public static Datetime ServerToday();
```

7.5.2.2 Return Value

The return value is the current datetime at the server.

7.5.2.3 Remarks

This should be used instead of the local workstation datetime.

7.5.2.4 C# Example

```
Datetime dtNow = HVCDateObj.ServerToday();
```

7.6 PObj Class

The PObj class is the base class for all of the subsequent classes. It provides common functionality that is inherent for all classes derived from it. This class cannot be created, only classes derived from it can be created. This class is not typically used directly the sample code shows usage through one of the derived level objects.

```
System.Object
SCMLib.PObj.PObj
```

7.6.1 Namespace

```
SCMLib.PObj
```

7.6.2 PObj.GUID

This method returns the GUID for the object.

7.6.2.1 Syntax

```
public string GUID [get];
```

7.6.2.2 Return Value

The return value is the GUID for the object.

7.6.2.3 Remarks

This method is a read-only property.

7.6.2.4 C# Example

```
string sGUID = UserPObj.GUID;
```


This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7.6.3 PObj.Dispose

This method releases unmanaged references within the class.

7.6.3.1 Syntax

```
public void Dispose();
```

7.6.3.2 Remarks

This method is called when the object is no longer required or used.

7.6.3.3 C# Example

```
// Once finished with the object call Dispose
UserPObj.Dispose();
```

7.7 UserPObj Class

The PObj class is the base class for all of the subsequent classes. It provides common functionality that is inherent for all classes derived from it. This class cannot be created, only classes derived from it can be created. This class is not typically used directly the sample code shows usage through one of the derived level objects.

```
System.Object
SCMLib.PObj.PObj
SCMLib.PObj.UserPObj
```

7.7.1 Namespace

```
SCMLib.PObj.
```

7.7.2 UserPObj Constructor

This constructor creates a UserPObj object and attaches to an existing Sunrise XA user.

7.7.2.1 Syntax

```
UserPObj( string sUserGUID );
```

7.7.2.2 Remarks

The constructor takes a UserGUID to attach to an existing user in the SCM database.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7.7.2.3 C# Example

```
// Get reference to the custom context singleton
CustomContextObj cc = CustomContextObj.GetInstance();
// Get the GUID of the logon user from context
string sCurrentUserGUID = cc.UserGUID;
// Create a UserPObj object and attach
UserPObj myCurrentUser = new UserPObj( sCurrentUserGUID );
// Get the user's display name
string sCurrentUserDisplayName = myCurrentUser.DisplayName;
```

7.7.3 UserPObj.DisplayName

This method returns the display name of the user.

7.7.3.1 Syntax

```
public static string DisplayName [get];
```

7.7.3.2 Return Value

The return value is the display name of the user.

7.7.3.3 Remarks

This method is a read-only property.

7.7.3.4 C# Example

```
UserPObj myUser = HVCLogonObj.GetCurrentUser();
string sDisplayName = myUser.DisplayName;
```

7.7.4 UserPObj.FirstName

This method returns the user's first name.

7.7.4.1 Syntax

```
public static string FirstName [get];
```

7.7.4.2 Return Value

The return value is the user's first name.

7.7.4.3 Remarks

This method is a read-only property.

7.7.4.4 C# Example

```
UserPObj myUser = HVCLogonObj.GetCurrentUser();
string sFirstName = myUser.FirstName;
```

7.7.5 UserPObj.LastName

This method returns the user's last name.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7.7.5.1 Syntax

```
public static string LastName [get];
```

7.7.5.2 Return Value

The return value is the user's last name.

7.7.5.3 Remarks

This method is a read-only property.

7.7.5.4 C# Example

```
UserPObj myUser = HVCLogonObj.GetCurrentUser();
string sLastName = myUser.LastName;
```

7.7.6 UserPObj.MiddleName

This method returns the user's middle name.

7.7.6.1 Syntax

```
public static string MiddleName [get];
```

7.7.6.2 Return Value

The return value is the user's middle name, if any.

7.7.6.3 Remarks

This method is a read-only property.

7.7.6.4 C# Example

```
UserPObj myUser = HVCLogonObj.GetCurrentUser();
string sMiddleName = myUser.MiddleName;
```

7.7.7 UserPObj.Title

This method returns the user's title string.

7.7.7.1 Syntax

```
public static string Title [get];
```

7.7.7.2 Return Value

The return value is the user's title string, if any. For example, "Mr.", or "Ms.".

7.7.7.3 Remarks

This method is a read-only property.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7.7.7.4 C# Example

```
UserPObj myUser = HVCLogonObj.GetCurrentUser();
string sTitle = myUser.Title;
```

7.7.8 UserPObj.BirthDayNum

This method returns the day value of the user's birthdate.

7.7.8.1 Syntax

```
public static int BirthDayNum [get];
```

7.7.8.2 Return Value

The return value is the day integer value of the user's birth date.

7.7.8.3 Remarks

This method is a read-only property.

7.7.8.4 C# Example

```
UserPObj myUser = HVCLogonObj.GetCurrentUser();
int nDay = myUser.BirthDayNum;
```

7.7.9 UserPObj.BirthMonthNum

This method returns the month value of the user's birthdate.

7.7.9.1 Syntax

```
public static int BirthMonthNum [get];
```

7.7.9.2 Return Value

The return value is the month integer value of the user's birth date.

7.7.9.3 Remarks

This method is a read-only property.

7.7.9.4 C# Example

```
UserPObj myUser = HVCLogonObj.GetCurrentUser();
int nMonth = myUser.BirthMonthNum;
```

7.7.10 UserPObj.BirthYearNum

This method returns the year value of the user's birthdate.

7.7.10.1 Syntax

```
public static int BirthYearNum [get];
```


This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7.7.10.2 Return Value

The return value is the year integer value of the user's birth date.

7.7.10.3 Remarks

This method is a read-only property.

7.7.10.4 C# Example

```
UserPObj myUser = HVCLogonObj.GetCurrentUser();
int nYear = myUser.BirthYearNum;
```

7.7.11 UserPObj.OccupationCode

This method returns the user's occupation code.

7.7.11.1 Syntax

```
public static string OccupationCode [get];
```

7.7.11.2 Return Value

The return value is the user's occupation code, as defined in the "Organization Information \ Occupation" dictionary.

7.7.11.3 Remarks

This method is a read-only property.

7.7.11.4 C# Example

```
UserPObj myUser = HVCLogonObj.GetCurrentUser();
string sOccupationCode = myUser.OccupationCode;
```

7.7.12 UserPObj.IDCode

This method returns the user's logon ID code.

7.7.12.1 Syntax

```
public static string IDCode [get];
```

7.7.12.2 Return Value

The return value is the ID code that used to logon to an SCM database.

7.7.12.3 Remarks

This method is a read-only property.

7.7.12.4 C# Example

```
UserPObj myUser = HVCLogonObj.GetCurrentUser();
string sIDCode = myUser.IDCode;
```


This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7.7.13 UserPObj.OrderRoleType

This method returns the assigned role that the user regarding orders.

7.7.13.1 Syntax

```
public static string OrderRoleType [get];
```

7.7.13.2 Return Value

The return value is the user's role for order, as defined in the "Organization Information \ Order Role Type" dictionary.

7.7.13.3 Remarks

This method is a read-only property.

7.7.13.4 C# Example

```
UserPObj myUser = HVCLogonObj.GetCurrentUser();
string sOrderRoleType = myUser.OrderRoleType;
```

7.8 ClientPObj Class

The ClientPObj class exposes context for an SCM client (patient).

```
System.Object
SCMLib.PObj.PObj
SCMLib.PObj.ClientPObj
```

7.8.1 Namespace

```
SCMLib.PObj
```

7.8.2 ClientPObj Constructor

Create a ClientPObj object and attach to an existing SCM client (patient).

7.8.2.1 Syntax

```
ClientPObj( string sClientGUID );
```

7.8.2.2 Remarks

The constructor takes a ClientGUID to attach to an existing client in the SCM database.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7.8.2.3 C# Example

```
// Get reference to the custom context singleton
CustomContextObj cc = CustomContextObj.GetInstance();
// Get the client GUID from context
string sClientGUID = cc.ClientGUID;
// Create a ClientPObj object and attach
ClientPObj myClient = new ClientPObj( sClientGUID );
// Get the client's display name
string sDisplayName = myClient.DisplayName;
```

7.8.3 ClientPObj.DisplayName

This method returns the display name of the client.

7.8.3.1 Syntax

```
public static string DisplayName [get];
```

7.8.3.2 Return Value

The return value is the display name of the user.

7.8.3.3 Remarks

This method is a read-only property.

7.8.3.4 C# Example

```
ClientPObj myClient = new ClientPObj( sClientGUID );
string sDisplayName = myClient.DisplayName;
```

7.8.4 ClientPObj.FirstName

This method returns the client's first name.

7.8.4.1 Syntax

```
public static string FirstName [get];
```

7.8.4.2 Return Value

The return value is the client's first name.

7.8.4.3 Remarks

This method is a read-only property.

7.8.4.4 C# Example

```
ClientPObj myClient = new ClientPObj( sClientGUID );
string sFirstName = myClient.FirstName;
```

7.8.5 ClientPObj.LastName

This method returns the client's last name.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7.8.5.1 Syntax

```
public static string LastName [get];
```

7.8.5.2 Return Value

The return value is the client's last name.

7.8.5.3 Remarks

This method is a read-only property.

7.8.5.4 C# Example

```
ClientPObj myClient = new ClientPObj( sClientGUID );
string sLastName = myClient.LastName;
```

7.8.6 ClientPObj.MiddleName

This method returns the client's middle name.

7.8.6.1 Syntax

```
public static string MiddleName [get];
```

7.8.6.2 Return Value

The return value is the user's middle name, if any.

7.8.6.3 Remarks

This method is a read-only property.

7.8.6.4 C# Example

```
ClientPObj myClient = new ClientPObj( sClientGUID );
string sMiddleName = myClient.MiddleName;
```

7.8.7 ClientPObj.Title

This method returns the client's title string.

7.8.7.1 Syntax

```
public static string Title [get];
```

7.8.7.2 Return Value

The return value is the client's title string, if any. For example, "Mr.", or "Ms."

7.8.7.3 Remarks

This method is a read-only property.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7.8.7.4 C# Example

```
ClientPObj myClient = new ClientPObj( sClientGUID );
string sTitle = myClient.Title;
```

7.8.8 ClientPObj.BirthDayNum

This method returns the day value of the client's birthdate.

7.8.8.1 Syntax

```
public static int BirthDayNum [get];
```

7.8.8.2 Return Value

The return value is the day integer value of the client's birth date.

7.8.8.3 Remarks

This method is a read-only property.

7.8.8.4 C# Example

```
ClientPObj myClient = new ClientPObj( sClientGUID );
int nBirthDayNum = myClient.BirthDayNum;
```

7.8.9 ClientPObj.BirthMonthNum

This method returns the month value of the client's birthdate.

7.8.9.1 Syntax

```
public static int BirthMonthNum [get];
```

7.8.9.2 Return Value

The return value is the month integer value of the client's birth date.

7.8.9.3 Remarks

This method is a read-only property.

7.8.9.4 C# Example

```
ClientPObj myClient = new ClientPObj( sClientGUID );
int nBirthMonthNum = myClient.BirthMonthNum;
```

7.8.10 ClientPObj.BirthYearNum

This method returns the year value of the client's birthdate.

7.8.10.1 Syntax

```
public static int BirthYearNum [get];
```


This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7.8.10.2 Return Value

The return value is the year integer value of the client's birth date.

7.8.10.3 Remarks

This method is a read-only property.

7.8.10.4 C# Example

```
ClientPObj myClient = new ClientPObj( sClientGUID );
int nBirthYearNum = myClient.BirthYearNum;
```

7.8.11 ClientPObj.ReligionCode

This method returns the client's religion code.

7.8.11.1 Syntax

```
public static string ReligionCode [get];
```

7.8.11.2 Return Value

The return value is this client's religion code as defined in the "Demographics \ Religion" dictionary.

7.8.11.3 Remarks

This method is a read-only property.

7.8.11.4 C# Example

```
ClientPObj myClient = new ClientPObj( sClientGUID );
string sReligionCode = myClient.ReligionCode;
```

7.8.12 ClientPObj.LanguageCode

This method returns the client's language code.

7.8.12.1 Syntax

```
public static string LanguageCode [get];
```

7.8.12.2 Return Value

The return value is the client's language code, as defined in the "Demographic \ Language" dictionary.

7.8.12.3 Remarks

This method is a read-only property.

7.8.12.4 C# Example

```
ClientPObj myClient = new ClientPObj( sClientGUID );
string sLanguageCode = myClient.LanguageCode;
```


This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7.8.13 ClientPObj.GenderCode

This method returns the client's gender code

7.8.13.1 Syntax

```
public static string GenderCode [get];
```

7.8.13.2 Return Value

The return value is the client's gender code, as defined in the "Demographics \ Gender" dictionary.

7.8.13.3 Remarks

This method is a read-only property.

7.8.13.4 C# Example

```
ClientPObj myClient = new ClientPObj( sClientGUID );
string sGenderCode = myClient.GenderCode;
```

7.8.14 ClientPObj.RaceCode

This method returns the client's race code.

7.8.14.1 Syntax

```
public static string RaceCode [get];
```

7.8.14.2 Return Value

The return value is the client's race code, as defined in the "Demographics \ Race" dictionary.

7.8.14.3 Remarks

This method is a read-only property.

7.8.14.4 C# Example

```
ClientPObj myClient = new ClientPObj( sClientGUID );
string sRaceCode = myClient.RaceCode;
```

7.8.15 ClientPObj.MaritalStatusCode

This method returns the client's marital status code.

7.8.15.1 Syntax

```
public static string MaritalStatusCode [get];
```


This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7.8.15.2 Return Value

The return value is the client's marital status code, as defined in the "Demographics \ Marital Status" dictionary.

7.8.15.3 Remarks

This method is a read-only property.

7.8.15.4 C# Example

```
ClientPObj myClient = new ClientPObj( sClientGUID );
string sMaritalStatusCode = myClient.MaritalStatusCode;
```

7.8.16 ClientPObj.IDCode

This method returns the ID code of the client.

7.8.16.1 Syntax

```
public static string IDCode [get];
```

7.8.16.2 Return Value

The return value is the client's ID code.

7.8.16.3 Remarks

This method is a read-only property.

7.8.16.4 C# Example

```
ClientPObj myClient = new ClientPObj( sClientGUID );
string sIDCode = myClient.IDCode;
```

7.8.17 ClientPObj.IsTemporary

This method checks if this client record is temporary.

7.8.17.1 Syntax

```
public static bool IsTemporary [get];
```

7.8.17.2 Return Value

Return true if this client record is temporary; false, otherwise. This can occur if the ADT system is down, and a temporary record and ID is created for the patient. Later, it is merged with the real client record.

7.8.17.3 Remarks

This method is a read-only property.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7.8.17.4 C# Example

```
ClientPObj myClient = new ClientPObj( sClientGUID );
bool bTemporary = myClient.IsTemporary;
```

7.8.18 ClientPObj.ApplicSource

This method identifies the application that created this client record.

7.8.18.1 Syntax

```
public static string ApplicSource [get];
```

7.8.18.2 Return Value

This method returns the application mnemonic that created this client record. It is typically “ADT” or “LAB” if the patient was admitted outside of SCM.

7.8.18.3 Remarks

This method is a read-only property.

7.8.18.4 C# Example

```
ClientPObj myClient = new ClientPObj( sClientGUID );
string sApplicSource = myClient.ApplicSource;
```

7.8.19 ClientPObj.CityOfBirth

This method returns the client’s city of birth.

7.8.19.1 Syntax

```
public static string CityOfBirth [get];
```

7.8.19.2 Return Value

This method returns the name of the city where the client was born.

7.8.19.3 Remarks

This method is a read-only property.

7.8.19.4 C# Example

```
ClientPObj myClient = new ClientPObj( sClientGUID );
string sCityOfBirth = myClient.CityOfBirth;
```

7.8.20 ClientPObj.CountryDvsnOfBirth

This method returns the country division where the client was born.

7.8.20.1 Syntax

```
public static string CountryDvsnOfBirth [get];
```


This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7.8.20.2 Return Value

The return value is the state or province where the patient was born.

7.8.20.3 Remarks

This method is a read-only property.

7.8.20.4 C# Example

```
ClientPObj myClient = new ClientPObj( sClientGUID );
string sCountryDvsnOfBirth = myClient.CountryDvsnOfBirth;
```

7.8.21 ClientPObj.CountryOfBirth

This method returns the country where the client was born.

7.8.21.1 Syntax

```
public static string CountryOfBirth [get];
```

7.8.21.2 Return Value

The return value is the name of the country where the patient was born.

7.8.21.3 Remarks

This method is a read-only property.

7.8.21.4 C# Example

```
ClientPObj myClient = new ClientPObj( sClientGUID );
string sCountryOfBirth = myClient.CountryOfBirth;
```

7.8.22 ClientPObj.IsVIP

This method determines if this client has VIP status.

7.8.22.1 Syntax

```
public static bool IsVIP [get];
```

7.8.22.2 Return Value

This method returns true if this client is a VIP; false, otherwise.

7.8.22.3 Remarks

This method is a read-only property.

7.8.22.4 C# Example

```
ClientPObj myClient = new ClientPObj( sClientGUID );
bool bVIP = myClient.IsVIP;
```


This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7.8.23 ClientPObj.PrivacyStatusCode

This method returns the client's privacy status code.

7.8.23.1 Syntax

```
public static string PrivacyStatusCode [get];
```

7.8.23.2 Return Value

The return value is the privacy status code of the client, as defined in the "Client Info \ Privacy Status" dictionary.

7.8.23.3 Remarks

This method is a read-only property.

7.8.23.4 C# Example

```
ClientPObj myClient = new ClientPObj( sClientGUID );
string sPrivacyStatusCode = myClient.PrivacyStatusCode;
```

7.8.24 ClientPObj.SecurityLevel

This method returns the security level of the client.

7.8.24.1 Syntax

```
public static int SecurityLevel [get];
```

7.8.24.2 Return Value

The return value is the security level of the client.

7.8.24.3 Remarks

The security level is determined by the corresponding privacy status code. It can be Standard (0), Enhanced (1), or Ultra (2).

7.8.24.4 C# Example

```
ClientPObj myClient = new ClientPObj( sClientGUID );
int nSecurityLevel = myClient.SecurityLevel;
```

7.9 ChartPObj Class

The ChartPObj class exposes context for an SCM chart.

```
System.Object
SCMLib.PObj.PObj
SCMLib.PObj.ChartPObj
```

7.9.1 Namespace

```
SCMLib.PObj
```


This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7.9.2 ChartPObj Constructor

This constructor creates a ChartPObj object and attaches to an existing SCM chart.

7.9.2.1 Syntax

```
ChartPObj( string sChartGUID );
```

7.9.2.2 Remarks

The constructor takes a ChartGUID to attach to an existing chart in the SCM database.

7.9.2.3 C# Example

```
// Get reference to the custom context singleton
CustomContextObj cc = CustomContextObj.GetInstance();
// Get the ChartGUID from context
string sChartGUID = cc.ChartGUID;
// Create a ChartPObj object and attach
ChartPObj myChart = new ChartPObj( sChartGUID );
// Get the user's status
string sStatus = myChart.Status;
```

7.10 ClientVisitPObj Class

The ClientVisitPObj class exposes context for an SCM client visit.

```
System.Object
SCMLib.PObj.PObj
SCMLib.PObj.ClientVisitPObj
```

7.10.1 Namespace

```
SCMLib.PObj
```

7.10.2 ClientVisitPObj Constructor

This constructor creates a ClientVisitPObj object and attaches to an existing SCM client visit.

7.10.2.1 Syntax

```
public ClientVisitPObj( string sClientVisitGUID );
```

7.10.2.2 Remarks

The constructor takes a ClientVisitGUID to attach to an existing client visit in the SCM database.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7.10.2.3 C# Example

```
// Get reference to the custom context singleton
CustomContextObj cc = CustomContextObj.GetInstance();
// Get the client GUID from context
string sVisitGUID = cc.ClientVisitGUID;
// Create a ClientPObj object and attach
ClientVisitPObj myVisit = new ClientVisitPObj( sVisitGUID );
// Get the client's display name
string sDisplayName = myVisit.ClientDisplayName;
```

7.10.3 ClientVisitPObj.ClientDisplayName

This method returns the display name of the client for the visit.

7.10.3.1 Syntax

```
public string ClientDisplayName [get];
```

7.10.3.2 Return Value

The return value is the display name of the client that this visit is related to.

7.10.3.3 Remarks

This method is a read-only property.

7.10.3.4 C# Example

```
ClientVisitPObj myVisit = new ClientVisitPObj( sVisitGUID );
string sDisplayName = myVisit.ClientDisplayName;
```

7.10.4 ClientVisitPObj.VisitIDCode

This method returns the ID Code for the visit.

7.10.4.1 Syntax

```
public string VisitIDCode [get];
```

7.10.4.2 Return Value

The ID Code for the visit.

7.10.4.3 Remarks

This method is a read-only property.

7.10.4.4 C# Example

```
ClientVisitPObj myVisit = new ClientVisitPObj( sVisitGUID );
string sIDCode = myVisit.VisitIDCode;
```

7.10.5 ClientVisitPObj.TypeCode

This method returns the type code for the type of visit.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7.10.5.1 Syntax

```
public string TypeCode [get];
```

7.10.5.2 Return Value

The return value is the type code for the type of visit.

7.10.5.3 Remarks

This method is a read-only property.

7.10.5.4 C# Example

```
ClientVisitPObj myVisit = new ClientVisitPObj( sVisitGUID );
string sTypeCode = myVisit.TypeCode;
```

7.10.6 ClientVisitPObj.CareLevelCode

This method returns the care level code for the visit.

7.10.6.1 Syntax

```
public string CareLevelCode [get];
```

7.10.6.2 Return Value

The return value is the care level code for the visit.

7.10.6.3 Remarks

This method is a read-only property.

7.10.6.4 C# Example

```
ClientVisitPObj myVisit = new ClientVisitPObj( sVisitGUID );
string sCareLevelCode = myVisit.CareLevelCode;
```

7.10.7 ClientVisitPObj.CurrentLocation

This method returns the current location for the client's visit.

7.10.7.1 Syntax

```
public string CurrentLocation [get];
```

7.10.7.2 Return Value

The return value is the current location for the client's visit.

7.10.7.3 Remarks

This method is a read-only property.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

7.10.7.4 C# Example

```
ClientVisitPObj myVisit = new ClientVisitPObj( sVisitGUID );
string sCurrentLocation = myVisit.CurrentLocation;
```


This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

CHAPTER 8: OBJECTSPLUS/XA FRAMEWORK

Contents

8.1 Overview 139
8.2 Prerequisites 140
8.3 Creating .NET Custom Applications 141
8.4 Using ObjectsPlus/XA Framework in MLMs 161

8.1 Overview

The ObjectsPlus/XA Framework consists of classes and interfaces that allow for the development of extensions to SCM applications. The ObjectsPlus/XA Framework types are CLS-compliant allowing these types to be used from any programming language conforming to the common language specification (CLS).

The ObjectsPlus/XA Framework is made up of three namespaces:

- ObjectsPlusXA.SunriseClinicalManager
- ObjectsPlusXA.SunriseClinicalManager.Common
- ObjectsPlusXA.SunriseClinicalManager.Forms

The ObjectsPlusXA.SunriseClinicalManager namespace contains classes that are used for:

- Creating Orders.
- Creating and maintaining Allergies.
- Creating and maintaining Care provider visit roles.
- Creating and maintaining Health issues.
- Creating and maintaining Patient comments.
- Creating and maintaining Significant events.
- Setting enterprise defined column values.

The ObjectsPlusXA.SunriseClinicalManager namespace also contains a number of classes used to support these main classes.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

The ObjectsPlusXA.SunriseClinicalManager.Common namespace contains classes and interfaces used by ObjectsPlusXA.SunriseClinicalManager for advanced handling of CDS Alerts, Warning messages, and Informational messages.

The ObjectsPlusXA.SunriseClinicalManager.Forms namespace contains classes allowing the Signature Manager and Order Reconciliation Manager forms to be shown. The Signature Manager class also allows items that must be signed or verified to be queried.

The API for the classes and interfaces included in the ObjectsPlus/XA Framework are found in the ObjectPlus/XA Framework class library help, installed with the ObjectsPlus/XA Framework SDK. The ObjectsPlus/XA Framework when installed, integrates with the MSDN Library.

The ObjectsPlus/XA Framework class library gives detailed information on all the classes and types plus some additional code examples.

8.2 Prerequisites

These prerequisites must be installed prior to using the ObjectsPlus/XA Framework to extend SCM:

- Microsoft Visual Studio 2005 with Service Pack 1.
- Infragistics NetAdvantage for .Net 2006 Volume 2 with HotFix 1079, if necessary.

Eclipsys recommends using the same tools when creating extensions to SCM in order to minimize the amount of memory consumed.

- One or more of these Eclipsys SCM applications:
  - Sunrise Acute Care.
  - Sunrise Critical Care.
  - Sunrise Ambulatory.
  - Sunrise Pharmacy.
  - Sunrise Emergency Department.
  - Sunrise MLM Editor if MLMs are going to be created that use the ObjectsPlus/XA Framework.

If a user does not have Microsoft Visual Studio 2005, a free version of Visual Studio 2005 Express Edition can be downloaded from the Microsoft website.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

8.3 Creating .NET Custom Applications

There are four ways for Custom Applications to interact with SCM:

- Custom Tabs.
- Custom Applications.
- Custom Form Applications.
- Standalone Applications.

Within these Custom Applications, developers can instantiate and use the ObjectsPlus/XA Framework classes.

The rest of this chapter explains how to develop these types of applications that use the classes in the ObjectsPlus/XA Framework. All of the examples show how to add and delete patient comments, how to create an order, and how to launch Signature Manager.

8.3.1 Common Setup

These steps are common to all types of Custom Applications.

◆ **C# Setup:**

1 Create a new C# project in Visual Studio. If creating a Custom Tab, select Windows Control Library as the project type. If creating a Custom Application or Custom Form Application, select Class Library as the project type. If creating a Standalone Application, select Windows Application as the project type.

2 Change the working directory for the project. Go to Solution Explorer, right-click on the project and select Properties. Select the Debug tab and set the Working directory to the SCM install directory.

3 Change the output path for the project. Go to Solution Explorer, right-click on the project and select Properties. Select the Build tab, and set the Output path to the SCM install directory.

Note: Some of the assemblies referenced can have additional dependencies. It is recommended to set the output and working folder to the directory where SCM is installed, or copy all files to a separate directory and reference the files from there.

4 Reference the required assemblies in the ObjectsPlus/XA Framework. Go to Solution Explorer, right-click on References and select Add Reference.

5 Click on the Browse tab, navigate to the SCM install directory, and select the required assemblies.

Some assemblies have dependencies on other assemblies, so multiple

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

assemblies might have to be referenced. For each assembly referenced, the Copy Local property for the assembly reference must be set to False.

6 To set the Copy Local property click the + sign to expand References, right-click on the assembly reference and select Properties. Set the Copy Local property to False.

◆ Visual Basic Setup:

1 Create a new VB project in Visual Studio. If creating a Custom Tab, select Windows Control Library as the project type. If creating a Custom Application or Custom Form Application, select Class Library as the project type. If creating a Standalone Application, select Windows Application as the project type.

2 Change the working directory. Go to Solution Explorer, right-click on project, and select Properties. Select the Debug tab and set the Working directory to the SCM install directory.

3 Change the output path. Using Solution Explorer, right-click on Project and select Properties. Select the Compile tab and set the Build output path to the SCM install directory.

Note: Some of the assemblies referenced might have additional dependencies. It is recommended to set the output and working folder to the SCM install directory, or copy all the files to a separate directory and reference the files there.

4 Reference the required assemblies in the ObjectsPlus/XA Framework. Using Solution Explorer, right-click on the project and select Properties.

5 Select the References Tab and click on Add Reference. Click on the Browse Tab, navigate to the directory where SCM is installed and select the required assemblies.

Some assemblies have dependencies on other assemblies, so multiple assemblies might have to be referenced. For each assembly referenced, the Copy Local property for the assembly reference must be set to False.

6 To set the Copy Local property right-click on the project in Solution Explorer and select Properties. Select the References Tab, then double-click on the reference. In the Properties window set the Copy Local property to False.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

8.3.2 Using ObjectsPlus/XA Framework in Custom Tabs

It is possible to add one or more Custom Tabs to the standard SCM user interface. Each of these tabs behaves in a manner similar to the standard tabs, has access to patient and user context information, and event notifications.

"Creating a C# Custom Tab Project" on page 143 and "Creating a Visual Basic Custom Tab Project" on page 149 contain a C# and VB example using the ObjectsPlusXA Framework in a Custom Tab. For detailed information on creating Custom Tabs, see "Creating Tabs" on page 9.

8.3.3 Creating a C# Custom Tab Project

◆ **To create a Custom Tab project:**

1 Create a new Windows Control Library Project, creating a new class called UserControl1.

2 Reference these six assemblies:

- BasicTabLib.dll
- ObjectsPlusXA.SCM
- ObjectsPlusXA.SCM.Common
- ObjectsPlusXA.SCM.Forms
- SCM.Common.Interfaces
- SCMLib.dll

Ensure that Copy Local is set to False for all of the references. See "Common Setup" on page 141 for more details.

3 Have the UserControl1 class implement the BasicTabLib.IPageView and SCMLib.CustomTab.ICustomTab interfaces. This requires several methods to be stubbed in, right-click on the interface and select Implement Interface.

4 Add these six controls to UserControls1 so it resembles Figure 8.9.

Table 8.5: CustomTab Controls

Control Type	Control Name
TextBox (Multiline)	textBox1
Label	label1
Four Buttons	button1, button2, button3, and button4

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

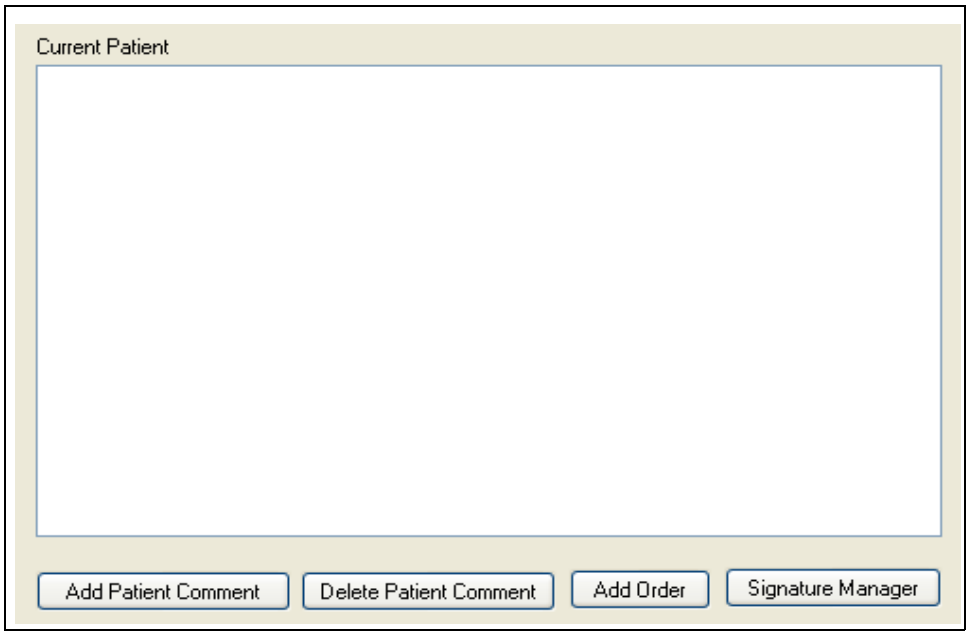


Figure 8.9: Custom Tab Control Form

5 Declare these private instance variables, as they are used later in the process.

```
private BASICTABLib.IPageViewSink _mySink;
private ObjectsPlusXA.SunriseClinicalManager.ClientVisit _visit;
private long _lastAddedCommentGUID;
```

Figure 8.10: Custom Tab Private Instance Variables Using C#

8.3.3.1 Implementing the IPageView Methods

The AdviseSink() and UnadviseSink() IPageView methods are used for establishing a connection between SCM and your control. AdviseSink() is called soon after SCM instantiates the UserControl, then passes an IPageViewSink interface as the parameter.

This IPageViewSink interface is cached by your control for subsequent use. In the example, this value is assigned to _mySink.

UnadviseSink() is called when SCM is about to shut down. This is necessary to release the IPageViewSink reference that the UserControl retains. Calling UnadviseSink() is an important step, because you can inadvertently prevent SCM from shutting down completely.

IPageViewSink is a COM interface pointer internal to SCM, and requires the necessary AddRef/Release calls to maintain its reference count.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

This C# code shows how to implement AdviseSink and UnadviseSink for our example.

```
public int AdviseSink(BASICTABLib.IPageViewSink PageViewSink)
{
    this._mySink = PageViewSink;
    return 0;
}

public void UnadviseSink(int ConnectId)
{

System.Runtime.InteropServices.Marshal.ReleaseComObject(this._mySi
nk);
    this._mySink = null;
}
```

Figure 8.11: Implementing AdviseSink and UnadviseSink IPageView Methods

The code that throws the exception should be removed from all stubbed in methods. The only other change for this interface's implementation is to have GetActionMenu return an empty string array, as shown.

```
public string[] GetActionMenu()
{
    return new string[0];
}
```

Figure 8.12: Returning an Empty String Array

8.3.3.2 Implementing the ICustomTab Methods

InitEvents is the only method on ICustomTab. In this example, InitEvents has code for handling the OnCanChangePatient and OnPatientChange events from SCM. In this code example the referenced event handlers are added.

```
public void InitEvents(SCMLib.CustomTab.CustomTabEventSource
eventSource)
{
    eventSource.OnCanChangePatient += new

SCMLib.CustomTab.SCMCanChangePatientHandler(eventSource_OnCanChang
ePatient);

    eventSource.OnPatientChange += new

SCMLib.CustomTab.SCMPatientChangeHandler(eventSource_OnPatientChan
ge);
}
```

Figure 8.13: InitEvents Method Code

The code for the event handlers for the OnCanChangePatient and OnPatientChange events is added.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

OnCanChangePatient

The OnCanChangePatient event handler returns true so that SCM can switch patients.

```
private bool eventSource_OnCanChangePatient()
{
    return true;
}
```

Figure 8.14: OnCanChangePatient Using C#

OnPatientChange

The OnPatientChange event handler gets the current visit in context, then shows the information for it. It also creates the ClientVisit object.

```
private void eventSource_OnPatientChange()
{
    // the patient has changed so get the related visit
    SCMLib.Context.CustomContextObj context =
        SCMLib.Context.CustomContextObj.GetInstance();
    long visitGUID = long.Parse(context.VisitGUID);

    // if there is already a ClientVisit object then call Dispose
on it
    if (this._visit != null)
    {
        this._visit.Dispose();
        this._visit = null;
    }

    // get the new ClientVisit object
    this._visit =

ObjectsPlusXA.SunriseClinicalManager.ClientVisit.FindByPrimaryKey(
visitGUID);

    UpdateAllControls();
}

private void UpdateAllControls()
{
   textBox1.Text =  "Client Name:  ";
   textBox1.Text +=  this._visit.ClientName + Environment.NewLine;
   textBox1.Text += "Visit ID:     ";
   textBox1.Text += this._visit.VisitId + Environment.NewLine;
   textBox1.Text += "Visit Status: ";
   textBox1.Text += this._visit.VisitStatus + Environment.NewLine;
   textBox1.Text += "Visit Reason: ";
   textBox1.Text += this._visit.VisitReason + Environment.NewLine;
}
```

Figure 8.15: OnPatientChange Using C#

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

8.3.3.3 Implement Click Handler for Button1

The click event handler for button1 creates a new Patient Comment using the PatientComment class in the ObjectsPlus/XA Framework. After the new Patient Comment is saved, the primary key is kept so it can be used later to delete the Patient Comment when button2 is clicked.

```
private void button1_Click(object sender, EventArgs e)
{
    // create new patient comment and save
    ObjectsPlusXA.SunriseClinicalManager.PatientComment comment =

ObjectsPlusXA.SunriseClinicalManager.PatientComment.CreatePatientC
omment(this._visit, "HEADER1");
    comment.Text = "ObjectPlus Test Comment";
    comment.Save();

    // save the guid of the newly created patient comment
    this._lastAddedCommentGUID = comment.PrimaryKey;

    // clean up
    comment.Dispose();
}
```

Figure 8.16: Button1 Click Handler Using C#

8.3.3.4 Implement Click Handler for Button2

The click event handler for button2 deletes the last Patient Comment created when button1 was clicked. The Delete() method on the PatientComment class marks the patient comment for removal, but the patient comment is not removed until Save() is called.

```
private void button2_Click(object sender, EventArgs e)
{
    // remove the last created patient comment
    if (this._lastAddedCommentGUID != 0)
    {
       ObjectsPlusXA.SunriseClinicalManager.PatientComment comment
=

ObjectsPlusXA.SunriseClinicalManager.PatientComment.FindByPrimaryK
ey(this._lastAddedCommentGUID);
        comment.Delete();
        comment.Save();

        this._lastAddedCommentGUID = 0;

        // clean up
        comment.Dispose();
    }
}
```

Figure 8.17: Button2 Click Handler Using C#

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

8.3.3.5 Implement Click Handler for Button3

The click event handler for button3 adds an ampicillin order. It looks up ampicillin in the catalog, then creates the order using the currently logged on user as the requesting provider for the order. This code example also shows how to set several properties on the new order.

```
private void button3_Click(object sender, EventArgs e)
{
    // find ampicillin
    ObjectsPlusXA.SunriseClinicalManager.OrderCatalogMasterItem
item =

ObjectsPlusXA.SunriseClinicalManager.OrderCatalogMasterItem.FindBy
Name("ampicillin");

    // get care provider of logged on user
    SCMLib.Context.CustomContextObj context =
SCMLib.Context.CustomContextObj.GetInstance();
    long providerGUID = long.Parse(context.UserGUID);
    ObjectsPlusXA.SunriseClinicalManager.CareProvider provider =

ObjectsPlusXA.SunriseClinicalManager.CareProvider.FindByPrimaryKey
(providerGUID);

    // create medication order
    ObjectsPlusXA.SunriseClinicalManager.MedicationOrder order =

ObjectsPlusXA.SunriseClinicalManager.MedicationOrder.CreateMedicat
ionOrder(this._visit,
            item, "Example of order creation", provider, "",
"Standard", "", null,

ObjectsPlusXA.SunriseClinicalManager.AvailabilityOverride.Prompt,

ObjectsPlusXA.SunriseClinicalManager.Common.SuppressNotificationOp
tions.Warnings);

// now set requested date to today
order.RequestedDate = "T";

    // set requested time to coded time of PM
    ObjectsPlusXA.SunriseClinicalManager.RequestedTime
requestedTime;
    requestedTime = new
ObjectsPlusXA.SunriseClinicalManager.RequestedTime("PM");
    order.RequestedTime = requestedTime;

    // set frequency to TID
    ObjectsPlusXA.SunriseClinicalManager.Frequency frequency;
    frequency = new
ObjectsPlusXA.SunriseClinicalManager.Frequency("TID");
    order.Frequency = frequency;

    // set dose and uom
    order.DosageLow = 15;
    order.Uom = "mg";
```


This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

```
    // now save order
    order.Save();

    // clean up
    item.Dispose();
    provider.Dispose();
    order.Dispose();

}
```

Figure 8.18: Button3 Click Handler Using C#

8.3.3.6 Implement Click Handler for Button4

The click event handler for button4 displays the Signature Manager. Signature Manager is a visual form and should be shown only from code running from within the SCM application. Signature Manager should not be shown from Time Delayed MLMs or from Standalone Applications.

This code example assumes that a using statement has already been added for ObjectsPlusXA.SunriseClinicalManager.Forms.

```
private void button4_Click(object sender, EventArgs e)
{
    // create SignatureManager
    SignatureManager signMgr = new SignatureManager();

    // configure the filter criteria through its properties
    signMgr.SignProviderSelection = SignProviderSelection.Own;
    signMgr.SignItemAction = SignItemAction.Reassign;
    signMgr.SignItemTypeFilters = SignItemTypeFilters.All;
    signMgr.SignItemStatusFilters = SignItemStatusFilters.Active;
    signMgr.ShowInTaskbar = false;

    // see how many orders, tasks, documents are to be signed
    int numOrders = signMgr.GetNumberOfOrdersToSign();
    int numTasks = signMgr.GetNumberOfTasksToSign();
    int numDocs = signMgr.GetNumberOfDocumentsToSign();

    // show the signature manager form
    signMgr.ShowSignDialog(this);

    // clean up
    signMgr.Dispose();
}
```

Figure 8.19: Button4 Click Handler Using C#

8.3.4 Creating a Visual Basic Custom Tab Project

◆ **To create a Visual Basic Custom Tab project:**

1 Create a new Windows Control Library Project, creating a new class called UserControl1.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

2 Reference the following six assemblies:

- BasicTabLib.dll
- ObjectsPlusXA.SCM
- ObjectsPlusXA.SCM.Common
- ObjectsPlusXA.SCM.Forms
- SCM.Common.Interfaces
- SCMLib.dll

Ensure that Copy Local is set to False for all of the references. See "Common Setup" on page 141 for more details.

3 Have UserControl1 class implement the BasicTabLib.IPageView and SCMLib.CustomTab.ICustomTab interfaces. When the Implements statement is added the methods are automatically stubbed in.

4 Add these six controls to UserControls1 so it resembles Figure 8.20.

Table 8.6: CustomTab Controls

Control Type	Control Name
TextBox (Multiline)	TextBox1
Label	Label1
Four Buttons	Button1, Button2, Button3, and Button4

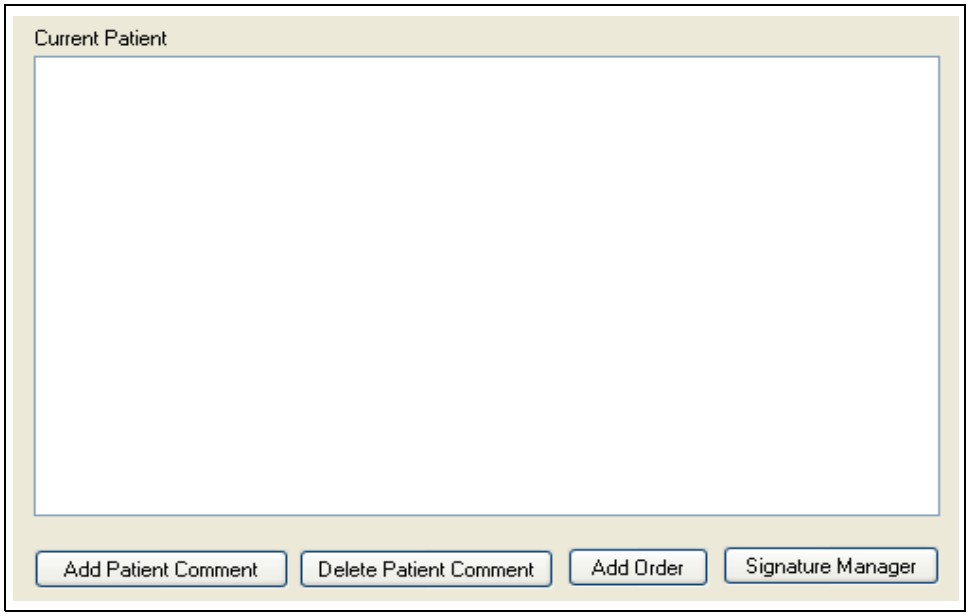


Figure 8.20: Custom Tab Control Form

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

5 Declare these private instance variables, as they are used later.

```
Private _mySink As BASICTABLib.IPageViewSink
Private _visit As ObjectsPlusXA.SunriseClinicalManager.ClientVisit
Private _lastAddedCommentGUID As Long
```

Figure 8.21: Custom Tab Private Instance Variables Using VB

8.3.4.1 Implementing the IPageView Methods

The AdviseSink() and UnadviseSink() IPageView methods are used for establishing a connection between SCM and your control. AdviseSink() is called soon after SCM instantiates the UserControl, then passes an IPageViewSink interface as the parameter.

This IPageViewSink interface is cached by your control for subsequent use. In the example, this value is assigned to _mySink.

UnadviseSink() is called when SCM is about to shut down. Calling UnadviseSink() is necessary to release the IPageViewSink reference that the UserControl retains. This is an important step, because you can inadvertently prevent SCM from shutting down completely.

IPageViewSink is a COM interface pointer internal to SCM, and requires the necessary AddRef/Release calls to maintain its reference count.

This code example shows how to implement AdviseSink and UnadviseSink for our example.

```
Public Function AdviseSink(ByVal PageViewSink As
BASICTABLib.IPageViewSink) As Integer Implements
BASICTABLib.IPageView.AdviseSink
    Me._mySink = PageViewSink
    Return 0
End Function

Public Sub UnadviseSink(ByVal ConnectId As Integer) Implements
BASICTABLib.IPageView.UnadviseSink

System.Runtime.InteropServices.Marshal.ReleaseComObject(Me._mySink
)
    Me._mySink = Nothing
End Sub

The only other change for this interface's implementation is to
have GetActionMenu return an empty string array as shown below.

Public Function GetActionMenu() As String() Implements
BASICTABLib.IPageView.GetActionMenu
    Return New String() {}
End Function
```

Figure 8.22: Implementing AdviseSink and UnadviseSink IPageView Methods Using VB

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

8.3.4.2 Implementing the ICustomTab methods

InitEvents is the only method on ICustomTab. In this example, InitEvents has code for handling the OnCanChangePatient and OnPatientChange events from SCM. In this code example the referenced event handlers are added.

```
Public Sub InitEvents(ByVal eventSource As
SCMLib.CustomTab.CustomTabEventSource) Implements
SCMLib.CustomTab.ICustomTab.InitEvents
    AddHandler eventSource.OnCanChangePatient, AddressOf
eventSource_OnCanChangePatient
    AddHandler eventSource.OnPatientChange, AddressOf
eventSource_OnPatientChange
End Sub
```

Figure 8.23: InitEvent Method Using VB

The code for the event handlers for the OnCanChangePatient and OnPatientChange events is added.

ONCANCHANGEPATIENT

The OnCanChangePatient event handler returns true so that SCM can switch patients.

```
Public Function eventSource_OnCanChangePatient() As Boolean
    Return True
End Function
```

Figure 8.24: OnCanChangePatient Using VB

ONPATIENTCHANGE

The OnPatientChange event handler gets the current visit in context, then shows the information for it. It also creates the ClientVisit object.

```
Public Sub eventSource_OnPatientChange()
    ' the patient has changed so get the related visit
    Dim context As SCMLib.Context.CustomContextObj =
SCMLib.Context.CustomContextObj.GetInstance()
    Dim visitGUID As Long = Long.Parse(context.VisitGUID)

   ' if there is already a ClientVisit object then call Dispose on
it
    If Me._visit IsNot Nothing Then
        Me._visit.Dispose()
        Me._visit = Nothing
    End If

    ' get the new ClientVisit object
    Me._visit =
ObjectsPlusXA.SunriseClinicalManager.ClientVisit.FindByPrimaryKey(
visitGUID)

    UpdateAllControls()
End Sub

Private Sub UpdateAllControls()
```


This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

```
    Me.TextBox1.Text = "Client Name:  " & Me._visit.ClientName & _
                       Environment.NewLine
    Me.TextBox1.Text = Me.TextBox1.Text & "Visit ID:     " & _
                       Me._visit.VisitId & Environment.NewLine
    Me.TextBox1.Text = Me.TextBox1.Text & "Visit Status: " & _
                       Me._visit.VisitStatus & Environment.NewLine
    Me.TextBox1.Text = Me.TextBox1.Text & "Visit Reason: " & _
                       Me._visit.VisitReason & Environment.NewLine
End Sub
```

Figure 8.25: OnPatientChange Using VB

8.3.4.3 Implement Click Handler for Button1

The click event handler for Button1 creates a new Patient Comment using the PatientComment class in the ObjectsPlus/XA Framework. After the new Patient Comment is saved, the primary key is kept so it can be used later to delete the Patient Comment when Button2 is clicked.

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
    ' create new patient comment and save
    Dim comment As
ObjectsPlusXA.SunriseClinicalManager.PatientComment = _

ObjectsPlusXA.SunriseClinicalManager.PatientComment.CreatePatientC
omment(Me._visit, "HEADER1")
    comment.Text = "ObjectPlus Test Comment"
    comment.Save()

    ' save the guid of the newly created patient comment
    Me._lastAddedCommentGUID = comment.PrimaryKey
End Sub
```

Figure 8.26: Button1 Click Handler Using VB

8.3.4.4 Implement Click Handler for Button2

The click event handler for Button2 deletes the last Patient Comment created when Button1 was clicked. The Delete() method on the PatientComment class marks the patient comment for removal, but the patient comment is not removed until Save() is called.

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
    ' remove the last created patient comment
    If Me._lastAddedCommentGUID <> 0 Then
        Dim comment As
ObjectsPlusXA.SunriseClinicalManager.PatientComment = _

ObjectsPlusXA.SunriseClinicalManager.PatientComment.FindByPrimaryK
ey(Me._lastAddedCommentGUID)
        comment.Delete()
        comment.Save()
```


This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

```
            Me._lastAddedCommentGUID = 0
        End If
End Sub
```

Figure 8.27: Button2 Click Handler Using VB

8.3.4.5 Implement Click Handler for Button3

The click event handler for Button3 adds an ampicillin order. It looks up ampicillin in the catalog, then creates the order using the currently logged on user as the requesting provider for the order. This example also shows how to set several properties on the new order.

```
Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button3.Click
    ' find ampicillin
    Dim item As
ObjectsPlusXA.SunriseClinicalManager.OrderCatalogMasterItem = _

ObjectsPlusXA.SunriseClinicalManager.OrderCatalogMasterItem.FindBy
Name("ampicillin")

    ' get care provider of logged on user
    Dim context As SCMLib.Context.CustomContextObj = _
        SCMLib.Context.CustomContextObj.GetInstance()
    Dim providerGUID As Long = Long.Parse(context.UserGUID)
    Dim provider As
ObjectsPlusXA.SunriseClinicalManager.CareProvider = _

ObjectsPlusXA.SunriseClinicalManager.CareProvider.FindByPrimaryKey
(providerGUID)

    ' create medication order
    Dim order As
ObjectsPlusXA.SunriseClinicalManager.MedicationOrder = _

ObjectsPlusXA.SunriseClinicalManager.MedicationOrder.CreateMedicat
ionOrder(Me._visit, _
            item, "Example of order creation", provider, "",
"Standard", "", Nothing, _

ObjectsPlusXA.SunriseClinicalManager.AvailabilityOverride.Prompt,
_

ObjectsPlusXA.SunriseClinicalManager.Common.SuppressNotificationOp
tions.Warnings)

    ' now set requested date to today
    Dim requestedDate As String =
DateTime.Today.ToString("yyyyMMdd")
    order.RequestedDate = requestedDate

    ' set requested time to coded time of PM
    Dim requestedTime As
ObjectsPlusXA.SunriseClinicalManager.RequestedTime = _
        New
ObjectsPlusXA.SunriseClinicalManager.RequestedTime("PM")
```


This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

```
    order.RequestedTime = requestedTime

    ' set frequency to TID
    Dim frequency As
ObjectsPlusXA.SunriseClinicalManager.Frequency = _
        New ObjectsPlusXA.SunriseClinicalManager.Frequency("TID")
    order.Frequency = frequency

    ' set dose and uom
    order.DosageLow = 15
    order.Uom = "mg"

    ' now save order
    order.Save()
End Sub
```

Figure 8.28: Button3 Click Handler Using VB

8.3.4.6 Implement Click Handler for Button4

The click event handler for Button4 displays the Signature Manager. Signature Manager is a visual form and should be shown only from code running from within the SCM application. Signature Manager should not be shown from Time Delayed MLMs or from Standalone Applications.

This code example assumes that a using statement has already been added for ObjectsPlusXA.SunriseClinicalManager.Forms.

```
Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button4.Click
    ' create SignatureManager
    Dim signMgr As SignatureManager = New SignatureManager()

    ' configure the filter criteria through its properties
    signMgr.SignProviderSelection = SignProviderSelection.Own
    signMgr.SignItemAction = SignItemAction.Reassign
    signMgr.SignItemTypeFilters = SignItemTypeFilters.All
    signMgr.SignItemStatusFilters = SignItemStatusFilters.Active
    signMgr.ShowInTaskbar = False

    ' see how many orders, tasks, documents are to be signed
    Dim numOrders As Integer = signMgr.GetNumberOfOrdersToSign()
    Dim numTasks As Integer = signMgr.GetNumberOfTasksToSign()
    Dim numDocs As Integer = signMgr.GetNumberOfDocumentsToSign()

    ' show the signature manager form
    signMgr.ShowSignDialog(Me)

    ' clean up
    signMgr.Dispose()
End Sub
```

Figure 8.29: Button4 Click Handler Using VB

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

8.3.5 Using ObjectsPlus/XA Framework in Standalone Applications

Standalone Applications are launched as a separate process, so the application must be a windows application (.EXE) and is required to explicitly logon. The other restriction for Standalone Applications is that the ObjectsPlusXA.SCM.Forms assembly cannot be used.

A valid Standalone Application does not have to inherit from any classes or implement any interfaces. Developers have flexibility in its inheritance and interfaces.

"Creating a C# Standalone Application" on page 156 and "Creating a VB Standalone Application" on page 158 contain a C# and VB example creating a Standalone Application to add a Patient Comment.

8.3.6 Creating a C# Standalone Application

◆ **To create a Standalone Application project:**

1 Create a new Windows Application Project, creating a new form named Form1.

2 Reference these three assemblies:

- ObjectsPlusXA.SCM
- ObjectsPlusXA.SCM.Common
- SCM.Common.Interfaces

Ensure that Copy Local is set to False for all of the references. See "Common Setup" on page 141 for more details.

3 Add a TextBox (multiline) and Button to Form1 called textBox1 and button1.

8.3.6.1 Private Instance Variables _logon and _visit

Add these instance variables to Form1:

```
private ObjectsPlusXA.SunriseClinicalManager.ScmLogon _logon;
private ObjectsPlusXA.SunriseClinicalManager.ClientVisit _visit;
```

Figure 8.30: _logon and _visit Private Instance Variables Using C#

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

8.3.6.2 Logon, Logoff, and Create ClientVisit

Unlike other types of Custom Applications, there is no easy way to obtain context information from SCM because a Standalone Application is completely independent. Developers can run SQL queries against the database after logging on with ScmLogon.

```
private void Form1_Load(object sender, EventArgs e)
{
    // create the ScmLogon object and logon
   _logon = new ObjectsPlusXA.SunriseClinicalManager.ScmLogon();
    logon.Logon("Test App", true, "username", "password");

    // create a ClientVisit object
   _visit =
ObjectsPlusXA.SunriseClinicalManager.ClientVisit.FindByPrimaryKey(
12345);

    UpdateAllControls();
}

private void UpdateAllControls()
{
   textBox1.Text = "Client Name:  ";
   textBox1.Text += this._visit.ClientName + Environment.NewLine;
   textBox1.Text += "Visit ID:     ";
   textBox1.Text += this._visit.VisitId + Environment.NewLine;
   textBox1.Text += "Visit Status: ";
   textBox1.Text += this._visit.VisitStatus + Environment.NewLine;
   textBox1.Text += "Visit Reason: ";
   textBox1.Text += this._visit.VisitReason + Environment.NewLine;
}
```

Figure 8.31: ScmLogon Using C#

For every Logon, a corresponding Logoff is required. In this example Logoff is called when the form closes.

```
private void Form1_FormClosed(object sender, FormClosedEventArgs
e)
{
    // clean up
   _visit.Dispose();

    // now logoff
   _logon.Logoff();
}
```

Figure 8.32: Logoff Using C#

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

8.3.6.3 Implement Event Handler for Button1

Button1 when clicked adds a Patient Comment in the same manner as the sample for the Custom Tab button1.

```
private void button1_Click(object sender, EventArgs e)
{
    // create new patient comment and save
    ObjectsPlusXA.SunriseClinicalManager.PatientComment comment =

ObjectsPlusXA.SunriseClinicalManager.PatientComment.CreatePatientC
omment(this._visit, "HEADER1");
    comment.Text = "ObjectPlus Test Comment";
    comment.Save();

    // clean up
    comment.Dispose();
}
```

Figure 8.33: Button1_Click Using C#

8.3.7 Creating a VB Standalone Application

◆ **To create a Standalone Application project:**

1 Create a new Windows Application Project, creating a new form called Form1.

2 Reference these three assemblies:

- ObjectsPlusXA.SCM
- ObjectsPlusXA.SCM.Common
- SCM.Common.Interfaces

Ensure that Copy Local is set to False for all of the references. See "Common Setup" on page 141 for more details.

3 Add a TextBox (multiline) and Button to Form1 called TextBox1 and Button1.

8.3.7.1 Private Instance Variables _logon and _visit

Add the following instance variables to Form1.

```
Private _logon As ObjectsPlusXA.SunriseClinicalManager.ScmLogon
Private _visit As ObjectsPlusXA.SunriseClinicalManager.ClientVisit
```

Figure 8.34: _logon and _visit Private Instance Variables Using VB

8.3.7.2 Logon, Logoff and Create ClientVisit

Unlike other types of Custom Applications, there is no easy way to obtain context information from SCM because a Standalone Application is completely independent. Developers can run SQL queries against the database after logging on with ScmLogon.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    ' create the ScmLogon object and logon
   _logon = New ObjectsPlusXA.SunriseClinicalManager.ScmLogon()
   _logon.Logon("Test App", True, "username", "password")

    ' create a ClientVisit object
   _visit =
ObjectsPlusXA.SunriseClinicalManager.ClientVisit.FindByPrimaryKey(
12345)

    UpdateAllControls()
End Sub

Private Sub UpdateAllControls()
    Me.TextBox1.Text = "Client Name:  " & Me._visit.ClientName & _
                       Environment.NewLine
    Me.TextBox1.Text = Me.TextBox1.Text & "Visit ID:     " & _
                       Me._visit.VisitId & Environment.NewLine
    Me.TextBox1.Text = Me.TextBox1.Text & "Visit Status: " & _
                       Me._visit.VisitStatus & Environment.NewLine
    Me.TextBox1.Text = Me.TextBox1.Text & "Visit Reason: " & _
                       Me._visit.VisitReason & Environment.NewLine
End Sub
```

Figure 8.35: ScmLogon Using VB

For every Logon, a corresponding Logoff is required. In this example Logoff is called when the form closes.

```
Private Sub Form1_FormClosed(ByVal sender As System.Object, ByVal
e As System.Windows.Forms.FormClosedEventArgs) Handles
MyBase.FormClosed
    ' clean up
   _visit.Dispose()

    ' now logoff
   _logon.Logoff()
End Sub
```

Figure 8.36: Figure 8.103: Logoff Using VB

8.3.7.3 Implement Event Handler for Button1

Just like the sample for the Custom Tab, clicking Button1 adds a Patient Comment.

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
    ' create new patient comment and save
    Dim comment As
ObjectsPlusXA.SunriseClinicalManager.PatientComment = _

ObjectsPlusXA.SunriseClinicalManager.PatientComment.CreatePatientC
omment(Me._visit, "HEADER1")
    comment.Text = "ObjectPlus Test Comment"
    comment.Save()
```


This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

```
    ' clean up
    comment.Dispose()
End Sub
```

Figure 8.37: Button1_Click Using VB

8.3.8 Best Practices

8.3.8.1 Dispose

Some ObjectsPlus/XA Framework objects implement the IDisposable interface. Implementing the IDisposable interface advertises to the consumers of the objects that the Dispose method should be called after the object is no longer required. Calling Dispose() frees up resources right away, rather than relying on the garbage collector to free them up.

Eclipsys recommends calling Dispose after you have finished with the object. This frees the underlying unmanaged objects. The underlying unmanaged objects are not thread safe, so require all calls to them to come from the same thread. The garbage collector runs on a different thread, so it is best if the objects are explicitly disposed.

8.3.8.2 Exceptions

The ObjectsPlus/XA Framework objects do not return error status codes. Instead, the objects throw exceptions when errors occur or when pre-conditions are not met. The most likely case for exceptions is when an ObjectsPlus/XA object requires configuration through its setter properties before calling methods like ShowDialog() or Save().

The statements accessing object properties and method calls need to be surrounded with try-catch-finally statements in C# and Try-Catch-Finally blocks in VB.

Handling exceptions is essential to working with the objects in the ObjectsPlus/XA Framework. Not only are the errors handled, but objects can also be disposed properly.

The ObjectsPlus/XA Framework has its own exception class called ObjectsPlusXAException. This exception class is thrown from many places, and contains an ExceptionCategory.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

When catching an ObjectsPlusXAException, the ExceptionCategory is used to determine a course of action. Table 8.7 outlines the five exception categories and the suggested handling for each exception.

Table 8.7: Exception Categories

Exception Category	Suggested Handling
UserCancelled	This exception category is thrown when the user has been prompted to continue or cancel an operation, and the user has decided to cancel. The exception should not be shown to the user but simply eaten.
CDSCancelled	This exception category is thrown when CDS Alerts are shown to the user and the user has decided to Go Back. The exception should not be shown to the user but simply eaten.
ApplicationError	This exception category is thrown for errors that the user should be able to fix themselves. The message in the exception is shown to the user so that the user can take corrective action.
SystemError	This exception category is thrown when system errors are encountered, but the user cannot correct the system error. The message in the exception should be shown to the user so that it can be reported.
FatalError	This exception category is thrown when extreme errors are encountered and the user likely cannot continue. The message in the exception should be shown to the user so that it can be reported.

8.4 Using ObjectsPlus/XA Framework in MLMs

.NET support is now available for use in MLM Coding. The .NET object model has been integrated with the Arden 2.5 object model, resulting in what is referred to as Arden.NET. This capability allows MLMs to work with the ObjectsPlus/XA Framework as well as any other .NET objects.

Arden.NET supports full integration with the ObjectsPlus/XA Framework, including some additional features specific for MLMs. The MLM author uses an MLM to create, manipulate, store, and query objects and call methods on any public .NET types. It is no longer necessary to write wrapper .DLL functions to utilize .NET capabilities.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

New examples of Arden.NET are found in the *Clinical Decision Support Configuration Guide* and the *Clinical Decision Support MLM Authoring Guide.*

8.4.1 Key Differences

Aside from the syntax difference, development with MLMs differ from C# or VB in several ways.

8.4.1.1 Arden.NET Syntax

Though there are many similarities between Arden.NET and languages such as C# and VB, there are differences in how MLMs work with the ObjectsPlus/XA Framework. Some of those differences can be noted by comparing the C# and VB examples for adding a Patient Comment earlier in this chapter, and those examples provided in the MLM Example section.

You can also compare the methods described in the Visual Studio help file for C# and VB code examples to those used in the Test ObjectsPlus MLMs.

8.4.1.2 Accessibility

Not all of the functionality in ObjectsPlus/XA Framework is accessible from MLMs . MLMs that are User, Batch or Any run modes use only two of the assemblies in the ObjectsPlus/XA Framework: ObjectsPlusXA.SCM and ObjectsPlusXA.SCM.Common.

For MLMs that are User run mode, the ObjectsPlusXA.SCM.Forms assembly can also be used but only if the MLM is not run in Time Delay Mode. The ObjectsPlusXA.SCM.Forms assembly contains UI functionality requiring it to be run from SCM applications.

8.4.1.3 Time Delay Mode

The majority of the ObjectsPlus/XA Framework objects require a ClientVisit to already exist in order to create them. If the MLM is triggered when a new visit is being created and an ObjectsPlus/XA Framework object is being created through the Create static methods, the MLM must run in Time Delay mode. Running in Time Delay mode ensures that the ClientVisit is already present in the database before the MLM is run.

In addition to this there are four ObjectsPlus/XA Framework objects that if used with an ObjectsPlus Destination, must also be run in Time Delay Mode.

These four objects are:

- Allergy
- CareProviderVisitRole
- HealthIssue
- PatientComment

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

Modifying these objects can trigger an HL7 ADT Message to be sent out. To ensure that all information is sent correctly in the message, the MLM that creates or maintains these objects must be run in Time Delay Mode.

8.4.1.4 Writing to the ObjectsPlus Destination

An MLM can be used to manipulate objects during the execution of an MLM or can be used to manipulate objects only when an MLM concludes TRUE. The latter can be done by writing the object data to the ObjectsPlus Destination. When data is written to the database using a CDS Destination statement, additional audit information is also written.

The audit trail provides information about what was done, and which user was logged on. The audit information is stored in the CV3MLMDestinationAudit table.

It is recommended that ObjectsPlus data is written to the database using the Destination statement, in order to ensure that that data is stored only after a user completes an action, and to ensure that audit information is kept.

For example, when an ObjectsPlus Destination is used to create a new Patient Comment in the OrderEnter or HealthIssueEnter Trigger Event, the save for the Patient Comment occurs only if the user decides not to submit the order. If an ObjectsPlus Destination is used to create an Order in the HealthIssueEnter Trigger Event, and there are CDS Alerts related to the Health Issue, then the new Order is saved only if the user decides to proceed with the saving of the Health Issue after viewing the CDS Alerts.

When an instance of an object has been created and fields are populated, the data can be written to the ObjectPlus Destination.

More on ObjectsPlus Destinations can be found in the *Clinical Decision Support MLM Authoring Guide*.

Note: Applets using the ObjectsPlus/XA Framework objects and Called MLMs cannot use the ObjectsPlus Destination, meaning that an audit trail is not maintained for the changes.

8.4.2 MLM Examples

This section contains an example of how an MLM can be used to add a Patient Comment similar to the earlier examples provided for C# and VB.

In order to create a comment using an ObjectsPlus Destination, the MLM has to have an ObjectsPlus Destination, create the object, associate the object with the destination, and then write to the ObjectsPlus Destination when the MLM concludes TRUE. In this example note the differences between the C# and the VB examples previously given in this chapter.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

To Add a Patient Comment:

1 Create a new MLM using the Sunrise MLM Editor, logon using an Eclipsys User Account, then from the File menu, select New MLM.

2 Add information as required to the Maintenance, Library and Explanation sections of the MLM.

3 Include the ObjectsPlus/XA Framework assemblies and namespaces in the Knowledge: Data section of the MLM.

For convenience a standard MLM, std_include_libs, is available that references the ObjectsPlusXA.SCM and ObjectsPlusXA.SCM.Common assemblies through the using statement. This is equivalent to referencing the assemblies in C# or VB. The standard MLM also includes the using namespace statement for these assemblies. The using namespace statement is equivalent to the using keyword in C# and the Imports keyword in VB.

The std_include_libs MLM is included using this arden syntax:

```
// Load the ObjectsPlus assemblies
standard_libs := MLM 'std_include_libs';
include standard_libs;
```

Figure 8.38: Include Assemblies Using Arden

4 Declare what type of event must occur to create the new comment. This is coded in the Knowledge: Data section of the MLM

For this example a comment is created when an Admission Type Health Issue is created.

```
// Trigger when new Health Issue is entered
health_issue_enter_event := event { HealthIssueEnter User
            HealthIssue: where TypeCode="Admission" };
```

Figure 8.39: HealthIssueEnter Event Using Arden

5 Declare the ObjectsPlus Destination for the PatientComment object in the Knowledge: Data section of the MLM.

Using the ObjectsPlus Destination ensures the PatientComment object is saved only after the user has decided to proceed if CDS Alerts have been generated.

If CDS Alerts have been generated, then the PatientComment is not saved unless the user decides to save the Health Issue.

When the destination is coded, the site's preferences for any Alert messages that could be generated on error are configured.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

```
new_comment_dest := destination { ObjectsPlus } with
                  [ alert_type := "Warning",
                    short_message := "Object created by MLM",
                    priority := "low",
                    scope := "chart",
                    rule_group := "Comment Object",
                    rule_number := 2010 ];
```

Figure 8.40: ObjectsPlus Destination Using Arden

6 Specify the event(s) that trigger the execution of the MLM. Set the evoking event in the Knowledge: Evoke section of the MLM to the variable that was declared in the Knowledge: Data section of the MLM.

In this example, the MLM executes when the health_issue_enter_event, declared earlier, occurs.

```
evoke:
       health_issue_enter_event;
```

Figure 8.41: Evoke Statement Using Arden

7 Instantiate the ClientVisit object used when creating the new Patient Comment.

In this example the try-catch blocks are added around the ObjectsPlus/XA Framework calls to capture any thrown exceptions. Adding these blocks is a good idea, as calls to .NET can throw exceptions that require handling. In this example the exception handling has been kept simplistic, for more advanced exception handling consult the test MLMs included with SCM.

```
try
 // get the .NET version of the Client Visit object.
 // Needed to create new ObjectsPlus objects
 client_visit_obj := call 'ClientVisit'.FindByPrimaryKey
     with ((EvokingObject.ClientVisitGUID as number)
         as 'Int64');
endtry;
catch Exception ex
 // If the client object cannot be retrieved,
 // do not attempt to create any ObjectsPlus objects.
 conclude true;
endcatch;
```

Figure 8.42: Instantiating the ClientVisit Object Using Arden

8 Create the PatientComment, set the Text property, then assign to the ObjectsPlus Destination. This is the core data gather and object preparation section of the MLM. In the Logic section you can review data and then decide whether to complete the action or not.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

```
try
 // The comment type that will be created. This has to
 // exist in the system already, otherwise an exception
 // will be thrown in this try block
 comment_type := "HEADER1";

 // Create an instance of PatientComment
 new_comment_obj := call 'PatientComment'.CreatePatientComment
      with (client_visit_obj, //ClientVisit
            comment_type);    //TypeCode
 // Set the Text property
 new_comment_obj.Text := "0bjects Plus Test " || now;

 // Set the destination ObjectsPlus
 new_comment_dest.ObjectsPlus := new_comment_obj;

 // Dispose the ClientVisit object
 return_value:= call client_visit_obj.Dispose;
endtry;
catch exception ex
 new_comment_dest := null;
endcatch
```

Figure 8.43: Creating the PatientComment Object Using Arden

9 Conclude TRUE in the Knowledge: Logic section of the MLM.

Perform any required comparisons in the Logic slot and conclude TRUE to perform that save action.

```
// Conclude True to perform any actions in the Actions slot.
conclude TRUE;
```

Figure 8.44: Conclude True in the Logic Slot Using Arden

10 Define the actions to take in the Knowledge: Action section of the MLM.

The action slot gets executed only if the logic slot concludes TRUE. Write to the destination only if the associated object requires saving.

```
//only save destination if there's something to be saved
if EXISTS new_comment_dest then
        write true at new_comment_dest;
endif;
```

Figure 8.45: Executing the Action Slot Using Arden

◆ To Deploy the MLM:

1 From the View menu, select Syntax Check View to check the syntax of the newly created MLM. This catches any errors such as missing semicolons. Any errors encountered are listed in the Syntax Check tab in the lower left corner of the window.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

2 From the View menu, select Test Clients to run the MLM against in the editor Doing so is required so the MLM has database information and a patient visit context to run the MLM.

3 From the View menu, select Execute View to run the MLM against the test client. The execution results are listed in the Execute MLM tab in the lower left corner of the window. In addition, a CDS dialog shows any objects about to be created.

4 Click on the Edit MLM Information tab on lower left corner, and set the Production check box. This setting tells the CDS Controller this is an active MLM to be evoked when the trigger event listed in the MLM occurs.

5 From the File menu, select Store MLM.

6 Logon to SCM and enter a Health Issue with a TypeCode of Admission. A new comment with TypeCode HEADER1 is created.

8.4.3 Using Test MLMs as a Starting Point

The Test MLMs for Objects Plus provided with the release show examples for creating and manipulating each of the ObjectsPlus/XA Framework objects using MLMs.

Code can be copied from any of these MLMs to your MLM, or the MLM can be used as a starting point and modified to specific requirements.

Detailed information on these MLMs is presented in the *Clinical Decision Support Example MLM Guide*.

8.4.4 Best Practices

8.4.4.1 Dispose

Arden.NET implementations in an MLM follow the same recommendations provided by Microsoft for other .NET programming. It is recommended that an MLM explicitly call Dispose() on the object, if the object supports the IDisposable interface, after the object is no longer required. Calling Dispose() frees up resources immediately. Many of the classes in the ObjectsPlus/XA Framework support the IDisposable interface.

The exception to this rule is when an object is assigned to the ObjectsPlus Destination. If an ObjectsPlus/XA Framework object is assigned to the ObjectPlus Destination, Dispose() cannot be called, as it is used by the CDS subsystem. The CDS subsystem is responsible for calling Dispose().

This Arden code shows an example of calling the Dispose() method.

```
return_value:= call some_obj.Dispose;
```

Note: Arden.NET syntax requires that a variable be assigned to this call even though Dispose() does not return a value.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

8.4.4.2 Exceptions

Errors that occur in calls to .NET methods are not shown during the syntax or execution checks performed in the MLM Editor. In order to ensure that any exception that is thrown from a .NET call is captured, be sure to wrap all .NET calls with a try-catch block.

Examples of try-catch blocks, and handling exceptions in the MLMs is found in the Test MLMs for ObjectsPlus in the *Clinical Decision Support Example MLM Guide*.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

Index

.NET Custom Tab
  building a custom tab library, 14
  configuring a custom tab, 15
  custom tab UserControl, 11
  event notifications, 20
  getting and setting context, 18
  overview, 9

A

AdditionalVerificationByFieldObj class, 58
AdditionalVerificationByValueObj class, 63
AdminInstructionsFieldObj class, 58
Adobe Reader, 2
ApplicSource, 132

B

BaseSolutionFieldObj class, 58
BaseSolutionValueObj class, 64
Best practices, 160
  MLMs, 167
BirthDayNum, 123, 128
BirthMonthNum, 123, 128
BirthYearNum, 123, 128

C

C# custom tab project, 143
C# standalone project, 156
CalculatedDoseFieldObj class, 58
CalculatedDoseValueObj class, 64
Calling custom DLLs from forms, 37
CareLevelCode, 137
ChartGUID, 117
ChartPObj Class, 134
ChartPObj Class Constructor, 135
CheckRightsByID(), 108
CityOfBirth, 132
ClientDisplayName, 136
ClientGUID, 116
ClientInfoObj class, 80
ClientPObj Class, 125
  ClientPObj.ApplicSource, 132
  ClientPObj.BirthDayNum, 128
  ClientPObj.BirthMonthNum, 128
  ClientPObj.BirthYearNum, 128
  ClientPObj.CityOfBirth, 132
  ClientPObj.CountryDvsnOfBirth, 132
  ClientPObj.CountryOfBirth, 133
  ClientPObj.DisplayName, 126
  ClientPObj.First Name, 126
  ClientPObj.GenderCode, 130
  ClientPObj.IDCode, 131
  ClientPObj.IsTermporary, 131
  ClientPObj.IsVIP, 133
  ClientPObj.LanguageCode, 129
  ClientPObj.LastName, 126
  ClientPObj.MaritalStatusCode, 130
  ClientPObj.MiddleName, 127
  ClientPObj.PrivacyStatusCode, 134
  ClientPObj.RaceCode, 130
  ClientPObj.ReligionCode, 129
  ClientPObj.SecurityLevel, 134
  ClientPObj.Title, 127
ClientPObj Class Constructor, 125
ClientVisitPObj Class, 135
  ClientDisplayName, 136
  constructor, 135
  CurrentLocation, 137
  TypeCode, 136
  VisitIDCode, 136
ClientVisitPObj ClassLCareLevelCode, 137
CollapsibleWarningFieldObj class, 59
CollapsibleWarningValueObj class, 64
CombinedMeasurementsFieldObj class, 59
CombinedMeasurementsValueObj class, 65
Common Setup
  custom applications, 141
CommunicationObj class, 48
ConditionalOrderFieldObj class, 59
ConditionalOrderValueObj class, 72
Configuring custom applications
  accessing form context information, 35
  building a custom application, 29
  calling a custom application from order entry, 27
  CLRApp attribute, 31
  defining tools menu items, 25
  defining tools menu items as custom applications, 26

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

ICustomApp interface, 31
ICustomFormApp interface, 32
overview, 29
receiving context change notifications, 33
modeless window support, 35
CountryDvsnOfBirth, 132
CountryOfBirth, 133
Creating, 9
creating .NET custom applications, 141
common setup, 141
creating a C# custom tab project, 143
creating a C# standalone project, 156
creating a VB custom tab project, 149
creating a VB standalone project, 158
CreatinineClearanceFieldObj class, 59
CreatinineClearanceValueObj class, 72
CurrentLocation, 137
Custom Form Applications
overview, 37
Custom form applications
accessing all fields on a form, 44
changing field properties, 46
changing field values, 47
class information, 48
CommunicationObj class, 48
FormFieldCollection class, 52, 54, 55, 56
FormFieldObj class, 56
FormObj class, 56
ClientInfoObj class, 80
field types, 57
AdditionalVerificationByFieldObj class, 58
AdminInstructionFieldObj class, 58
BaseSolutionFieldObj class, 58
CalculatedDoseFieldObj class, 58
CollapsibleWarningFieldObj class, 59
CombinedMeasurementsFieldObj class, 59
ConditionalOrderFieldObj class, 59
CreatinineClearanceFieldObj class, 59
FrequencyCodeFieldObj class, 60
IsNoTaskFieldObj class, 61
IsPRNFieldObj class, 61
IVAdditiveCalcFieldObj class, 61
IVAdditiveFiledObj class, 61
MultiOrderCheckboxFieldObj class, 61
MultiOrderGridFieldObj class, 62
MultiOrderInlineFieldObj class, 62
OrderIVRateFieldObj class, 62
RequestedTimeFieldObj Class, 62
StopAfterFieldObj class, 63
StopTimeFieldObj Class, 63
VariableDoseFieldObj class, 63
field typesSpecialInstructionsFieldObj class, 62
field value types, 63
AdditionalVerificationByValueObj class, 63
BaseSolutionValueObj class, 64
CalculatedDoseValueObj class, 64
CollapsibleWarningValueObj class, 64
CombinedMeasurementsValueObj class, 65
ConditionalOrderValueObj class, 72
CreatinineClearanceValueObj class, 72
EnhancedTimeValueObj class, 72
FrequencyCodeValueObj class, 73
IVAdditiveCalcListObj class, 73
IVAdditiveCalcValueObj class, 73
IVAdditiveListObj class, 73
IVAdditiveValueObj class, 74
IVRateValueObj class, 75
MultiOrderItemCollection class, 75
MultiOrderItemValueObj class, 76
StopAfterValueObj class, 80
VariableDoseListObj class, 80
VariableDoseValueObj class, 80
finding a specific form field, 45
integrating, 41
type-safe access, 48
using ClientInfoObj class, 43
using CommunicationObj class, 43
using FormObj class, 43
writing an application, 42
CustomContextObj Class, 115
CustomContextObj.ChartGUID, 117
CustomContextObj.ClientGUID, 116
CustomContextObj.GetConnectionString(), 118
CustomContextObj.GetInstance(), 115
CustomContextObj.GetItem(), 116
CustomContextObj.RemoveItem(), 116
CustomContextObj.SetItem(), 115
CustomContextObj.UserGUID, 118
CustomContextObj.VisitGUID, 117

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

D

DisplayName, 121, 126
Dispose, 120

E

EnhancedTimeValueObj class, 72
External form processing API, 82
  IExternalFormProcessing interface, 83
  IExternalFormProcessing.Process method, 83
  IExtFormSupport interface, 85
  IExtFormSupport.GetLabel method, 91
  IExtFormSupport.GetValue method, 86
  IExtFormSupport.IsMandatory method, 88
  IExtFormSupport.IsReadOnly method, 90
  IExtFormSupport.IsVisible method, 89
  IExtFormSupport.SetValue method, 87
  integrating in a called program, 82

F

FirstName, 121, 126
FormField Collection class, 52, 54, 55, 56
FormFieldObj class, 56
FormObj class, 56
FrequencyCodeFieldObj class, 60
FrequencyCodeValueObj class, 73

G

GenderCode, 130
GetApplicationMode(), 114
GetApplicationName(), 109
GetBuild(), 107
GetClientVersion(), 112
GetColorPreference(), 109
GetConnectionString(), 104, 118
GetCurrentDatabase(), 113
GetCurrentUser(), 104
GetCurrentUserName(), 105
GetDatabaseVersion(), 113
GetDateDisplayFormat(), 111
GetDateEntryFormat(), 111
GetEnvironmentProfileInt(), 106
GetEnvironmentProfileString(), 105
GetEnvWorkstationProfileInt(), 107
GetEnvWorkstationProfileString(), 106
GetInstance(), 115
GetItem(), 116
GetLabel method, 91
GetOperatingSystem(), 114
GetSessionID(), 105
GetTimeDisplayFormat(), 111
GetValue method, 86
GetVersion(), 112
GetWorkstationGUID(), 107
GetWorkstationIDCode(), 108
GUID, 119

H

Handle, 114
HVCDateObj Class, 118
  HVCDateObj.ServerToday(), 119
HVCLogonObj class, 103
  HVCLogonObj.CheckRightsByID(), 108
  HVCLogonObj.GetApplicationName(), 109
  HVCLogonObj.GetBuild(), 107
  HVCLogonObj.GetColorPreference(), 109
  HVCLogonObj.GetConnectionString(), 104
  HVCLogonObj.GetDateDisplayFormat(), 111
  HVCLogonObj.GetDateEntryFormat(), 111
  HVCLogonObj.GetEnvironmentProfileInt(), 106
  HVCLogonObj.GetEnvironmentProfileString(), 105
  HVCLogonObj.GetEnvWorkstationProfileInt(), 107
  HVCLogonObj.GetEnvWorkstationProfileString(), 106
  HVCLogonObj.GetSessionID(), 105
  HVCLogonObj.GetTimeDisplayFormat(), 111
  HVCLogonObj.GetWorkstationGUID(), 107
  HVCLogonObj.GetWorkstationIDCode(), 108
  HVCLogonObjGetCurrentUser(), 104
  HVCLogonObjGetCurrentUserName(), 105

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

I

IDCode, 124, 131
IExternalFormProcessing Interface, 83
IExtFormSupport interface, 85
Integration
  notification server, 93
Interface details
  notification server, 94
Ismandatory method, 88
IsNoTaskFieldObj class, 61
IsPRNFieldObj class, 61
IsReadOnly method, 90
IsTemporary, 131
IsVIP, 133
IsVisible method, 89
IVAdditiveCalcFieldObj class, 61
IVAdditiveCalcListObj class, 73
IVAdditiveCalcValueObj class, 73
IVAdditiveFieldObj class, 61
IVAdditiveListObj class, 73
IVAdditiveValueObj class, 74
IVRateValueObj class, 75

K

Key differences in MLMs, 162

L

LanguageCode, 129
LastName, 121, 126

M

MainFrameObj Class
  MainFrameObj.GetApplicationMode(), 114
  MainFrameObj.GetCurrentDatabase(), 113
  MainFrameObj.GetOperatingSystem(), 114
  MainFrameObj.Handle, 114
MainFrameObj class, 112
  MainFrameObj.GetClientVersion(), 112
  MainFrameObj.GetDatabaseVersion(), 113
  MainFrameObj.GetVersion(), 112
MaritalStatusCode, 130
MiddleName, 122, 127
MLM Examples, 163
MultiOrderCheckboxFieldObj class, 61
MultiOrderGridFieldObj class, 62
MultiOrderInlineFieldObj class, 62
MultiOrderItemCollection class, 75
MultiOrderItemValueObj class, 76

N

Notification server
  Integration, 93
  Interface details, 94
  overview, 93
  writing transports, 93

O

ObjectsPlus/XA framework, 139
  best practices, 160
  creating .NET custom applications, 141
  common setup, 141
  creating a C# custom tab project, 143
  creating a C# standalone project, 156
  creating a VB custom tab project, 149
  creating a VB standalone project, 158
  MLM examples, 163
  MLMs, 161
  best practices, 167
  Key differences, 162
  overview, 139
  prerequisites, 140
  standalone applications, 156
  test MLMs, 167
OccupationCode, 124
OrderIVRateFieldObj class, 62
OrderRoleType, 125
Overview
  notification server, 93

P

PObj Class, 119
  PObj.Dispose, 120
  PObj.GUID, 119
PrivacyStatusCode, 134
Process method, 83

R

RaceCode, 130
ReligionCode, 129

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

RemoveItem(), 116
RequestedTimeFieldObj class, 62
rotating pages, 3

S

SecurityLevel, 134
ServerToday, 119
SetItem(), 115
SetValue method, 87
SpecialInstructionsFieldObj class, 62
StopAfterFieldObj class, 63
StopAfterValueObj class, 80
StopTimeFiledObj class, 63

T

Test MLMs, 167
Title, 122, 127
TypeCode, 136

U

User PObj Class, 120
  User PObj.BirthDayNum, 123
  User PObj.BirthMonthNum, 123
  User PObj.BirthYearNum, 123
  User PObj.DisplayName, 121
  User PObj.FirstName, 121
  User PObj.IDCode, 124
  User PObj.LastName, 121
  User PObj.MiddleName, 122
  User PObj.OccupationCode, 124
  User PObj.OrderRoleType, 125
  User PObj.Title, 122
User PObj Constructor, 120
UserGUID, 118
Using ObjectsPlus/XA framework
  MLM examples, 163
  MLMs, 161
  best practices, 167
  test MLMs, 167
using ObjectsPlus/XA framework in standalone applications, 156

V

VariableDoseFieldObj class, 63
VariableDoseListObj class, 80
VariableDoseValueObj class, 80
VB custom tab project, 149
VB standalone project, 158
VisitGUID, 117
VisitIDCode, 136

W

Writing transports, 93

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.

This page contains Eclipsys proprietary information and is not to be duplicated or disclosed to unauthorized persons.