



RESOURCE AND PATIENT MANAGEMENT SYSTEM

# **Electronic Health Record (RPMS-EHR)**

## **Technical Manual**

Version 1.1  
October 2007

Office of Information Technology (OIT)  
Division of Information Resource Management  
Albuquerque, New Mexico

## PREFACE

This guide provides information regarding technical aspects of the Indian Health Service RPMS Electronic Health Record (RPMS-EHR) v1.1 software. Its target audience is local and regional information technology support personnel who can be called upon to configure or troubleshoot the application.

The VueCentric<sup>®</sup> Framework is a proprietary product of Medsphere Systems Corporation (<http://www.medsphere.com>) used under license by Indian Health Service. Use and/or distribution outside the terms of this license is strictly prohibited.

## TABLE OF CONTENTS

<b>1.0</b>	<b>Introduction .....</b>	<b>1</b>
<b>2.0</b>	<b>VueCentric® Framework .....</b>	<b>2</b>
2.1	Introduction .....	2
2.2	Architecture .....	2
2.3	Implementation and Maintenance .....	4
2.3.1	VueCentric System Management (vcManager) Utility .....	4
2.3.1.1	Introduction.....	4
2.3.1.2	Logon Screen.....	5
2.3.1.3	Application Menu.....	6
2.3.1.4	Object Registry Tab.....	6
2.3.1.5	Template Registry Tab .....	20
2.3.1.6	Site Parameters Tab .....	24
2.3.1.7	Shutdown Tab .....	26
2.3.1.8	Monitor Tab .....	28
2.3.2	Ini Configuration (vcIniConfig) Utility.....	29
2.4	Routine Descriptions .....	31
2.5	File List.....	31
2.5.1	VueCentric Object Registry File (#19930.2) .....	31
2.5.2	VueCentric Object Category File (#19930.21) .....	33
2.5.3	VueCentric Template Registry File (#19930.3).....	34
2.6	Cross References .....	34
2.7	Callable Routines .....	34
2.7.1	\$\$HASKEY^CIAVCXUS .....	34
2.7.2	RPC: CIAVCXUS HASKEYS.....	34
2.7.3	RPC: CIAVCXUS VALIDPSW .....	35
2.7.4	RPC: CIAVMRPC INIT .....	35
2.7.5	RPC: CIAVMRPC DISV .....	35
2.7.6	RPC: CIAVMRPC PKG.....	35
2.7.7	RPC: CIAVMRPC PATCH .....	36
2.7.8	RPC: CIAVMRPC GETPAR .....	36
2.7.9	RPC: CIAVMRPC GETPARLI .....	36
2.7.10	RPC: CIAVMRPC GETPARWP .....	37
2.7.11	\$\$ENT^CIAVMRPC.....	37
2.7.12	RPC: CIAVMRPC SETPAR .....	38
2.7.13	RPC: CIAVMRPC GETVAR .....	38
2.7.14	RPC: CIAVMRPC SETVAR .....	38
2.7.15	RPC: CIAVMRPC GETIDX.....	39
2.7.16	RPC: CIAVMRPC STRTODAT .....	39
2.7.17	\$\$VERCMP^CIAVMRPC.....	39
2.7.18	\$\$TMPGBL^CIAVMRPC .....	40
2.7.19	RPC: CIAVUTIL SDINIT .....	40
2.7.20	RPC: CIAVUTIL SDABORT.....	40
2.7.21	RPC: CIAVUTIL MSGLOGIN.....	41

2.7.22	RPC: CIAVUTPR GETTPL .....	41
2.8	External Relations .....	41
2.9	Internal Relations .....	42
2.10	Exported Options .....	42
2.11	Exported Security Keys.....	42
2.12	Exported Protocols.....	43
2.13	Exported Parameters .....	43
2.14	Exported Mail Groups .....	44
2.15	Archiving and Purging.....	44
2.16	Components.....	44
2.16.1	Visual Interface Manager.....	44
2.16.1.1	Command Line Parameters .....	45
2.16.1.2	VIM Automation Object .....	48
2.16.2	Component Support Services.....	50
2.16.2.1	Server Automation Object .....	51
2.16.2.2	Session Automation Object .....	51
2.16.2.3	CSS_SessionEvents .....	63
2.16.2.4	Context Change Events .....	66
2.16.3	Component Management Service .....	67
2.16.3.1	Registry Object (CIA_CMS.CMS_Registry) .....	67
2.16.3.2	Component Object (CIA_CMS.CMS_Component) .....	69
2.16.4	Object Registry .....	71
2.16.5	Template Registry.....	71
2.16.5.1	Internal Representation .....	71
2.16.5.2	TObjectContainer .....	73
2.16.5.3	TPanelEx.....	73
2.16.5.4	TScrollBoxEx.....	73
2.16.5.5	TLabelEx .....	74
2.16.5.6	TToolBarEx.....	74
2.16.5.7	TPageControlEx .....	74
2.16.5.8	TTabSheetEx .....	74
2.16.5.9	TSplitterPaneEx .....	75
2.16.5.10	TPaneEx.....	75
2.16.5.11	TTreeViewEx.....	75
2.16.5.12	TTreePaneEx .....	75
2.16.5.13	TGroupBarEx .....	76
2.16.5.14	TGroupPaneEx.....	77
2.16.5.15	TMenuItemEx.....	77
2.16.5.16	XML Representation.....	77
2.16.6	Object Repository .....	78
<b>3.0</b>	<b>Remote Monitoring Service.....</b>	<b>80</b>
3.1	Introduction .....	80
3.2	Implementation and Maintenance .....	80
3.3	Routine Descriptions .....	81
3.4	File List.....	81

3.5	Cross References .....	81
3.6	Exported Options .....	81
3.7	Exported Security Keys.....	82
3.8	Exported Protocols.....	82
3.9	Exported Parameters .....	82
3.10	Exported Mail Groups .....	82
3.11	Callable Routines .....	82
3.12	External Relations .....	82
3.13	Internal Relations .....	82
3.14	Archiving and Purging.....	82
3.15	Components.....	82
3.15.1	Properties .....	82
3.15.2	GetData .....	83
<b>4.0</b>	<b>Date Service.....</b>	<b>84</b>
4.1	Introduction .....	84
4.2	Implementation and Maintenance .....	84
4.3	Routine Descriptions .....	84
4.4	File List.....	84
4.5	Cross References .....	84
4.6	Exported Options .....	84
4.7	Exported Security Keys.....	85
4.8	Exported Protocols.....	85
4.9	Exported Parameters .....	85
4.10	Exported Mail Groups .....	85
4.11	Callable Routines .....	85
4.12	External Relations .....	85
4.13	Internal Relations .....	85
4.14	Archiving and Purging.....	85
4.15	Components.....	85
4.15.1	DateRange.....	85
4.15.2	DateSelect .....	86
4.15.3	DateToFMDate .....	86
4.15.4	DateToFMDateStr.....	86
4.15.5	DefaultDateFormat .....	87
4.15.6	FMDateStrToDate.....	87
4.15.7	FMDateStrToFMDate .....	87
4.15.8	FMDateToDate .....	87
4.15.9	FMDateToFMDateStr .....	87
4.15.10	FormatAge .....	88
4.15.11	HL7DateToDate.....	88
4.15.12	HODateToDate .....	88
<b>5.0</b>	<b>Print Service .....</b>	<b>89</b>
5.1	Introduction .....	89
5.2	Implementation and Maintenance .....	89

5.3	Routine Descriptions .....	89
5.4	File List.....	89
5.5	Cross References .....	89
5.6	Exported Options .....	89
5.7	Exported Security Keys.....	90
5.8	Exported Protocols.....	90
5.9	Exported Parameters .....	90
5.10	Exported Mail Groups .....	90
5.11	Callable Routines .....	90
5.11.1	OUTPUT^CIAVUTIO .....	90
5.11.2	RPC: CIAVUTIO PRINT .....	91
5.11.3	RPC: CIAVUTIO PRTGETDF .....	91
5.11.4	RPC: CIAVUTIO PRTSETDF .....	91
5.11.5	RPC: CIAVUTIO DEVICE.....	92
5.12	External Relations .....	92
5.13	Internal Relations .....	92
5.14	Archiving and Purging.....	92
5.15	Components.....	92
5.15.1	Properties .....	93
5.15.2	ClosePreview.....	93
5.15.3	FindPreview .....	93
5.15.4	Format .....	93
5.15.5	Preview .....	94
5.15.6	Preview2 .....	95
5.15.7	Print .....	96
5.15.8	Print2 .....	96
5.15.9	Reset .....	97
5.15.10	SelectPrinter .....	97
5.15.11	UpdatePreview .....	97
<b>6.0</b>	<b>Remote Procedure Call (RPC) Broker .....</b>	<b>98</b>
6.1	Introduction .....	98
6.2	Architecture .....	98
6.3	Implementation and Maintenance .....	99
6.4	Routine Descriptions .....	100
6.5	File List.....	100
6.5.1	CIA AUTHENTICATION File (#19941.2) .....	100
6.5.2	CIA EVENT LOG File (#19941.23) .....	100
6.5.3	CIA EVENT TYPE File (#19941.21) .....	101
6.5.4	CIA LISTENER File (#19941.22) .....	102
6.6	Cross References .....	102
6.7	Exported Options .....	102
6.8	Exported Security Keys.....	103
6.9	Exported Protocols.....	103
6.10	Exported Parameters .....	103
6.11	Exported Mail Groups .....	104

6.12	Callable Routines .....	104
6.12.1	Server Management .....	104
6.12.1.1	DEBUG^CIANBLIS.....	104
6.12.1.2	MSERVER^CIANBLIS.....	104
6.12.1.3	START^CIANBLIS.....	104
6.12.1.4	STOPALL^CIANBLIS .....	105
6.12.2	Session Management .....	105
6.12.2.1	RPC: CIANBRPC GETSESS .....	105
6.12.2.2	\$\$SESSION^CIANBUTL .....	105
6.12.2.3	\$\$SHOWSESS^CIANBUTL / SHOWSESS^CIANBUTL .....	106
6.12.2.4	\$\$GETUID^CIANBUTL .....	106
6.12.2.5	\$\$NXTUID^CIANBUTL / NXTUID^CIANBUTL.....	106
6.12.2.6	\$\$CLRVAR^CIANBUTL / CLRVAR^CIANBUTL .....	106
6.12.2.7	\$\$GETVAR^CIANBUTL .....	107
6.12.2.8	\$\$SETVAR^CIANBUTL / SETVAR^CIANBUTL.....	107
6.12.2.9	RPC: CIANBRPC GETVAR .....	107
6.12.2.10	RPC: CIANBRPC SETVAR.....	108
6.12.3	Event Management.....	108
6.12.3.1	\$\$BRDCAST^CIANBEVT / BRDCAST^CIANBEVT / RPC: CIANBEVT BCAST.....	108
6.12.3.2	DOPURGE^CIANBEVT.....	108
6.12.3.3	RPC: CIANBEVT GETSUBSC .....	109
6.12.3.4	QUEUE^CIANBEVT .....	109
6.12.3.5	\$\$RELATES^CIANBEVT .....	109
6.12.3.6	SUBSCR^CIANBEVT / \$\$SUBSCR^CIANBEVT .....	109
6.12.3.7	TASKPRG^CIANBEVT.....	110
6.12.3.8	UNSUBALL^CIANBEVT .....	110
6.12.4	Miscellaneous .....	110
6.12.4.1	CLEANUP^CIANBUTL .....	110
6.12.4.2	REBLDCTX^CIANBUTL.....	110
6.12.4.3	RPC: CIANBRPC CANRUN.....	110
6.12.4.4	\$\$GETDLG^CIANBUTL / GETDLG^CIANBUTL .....	110
6.12.4.5	RPC: CIANBRPC DIALOG.....	111
6.12.5	External Relations.....	111
6.13	Internal Relations .....	111
6.14	Archiving and Purging .....	111
6.15	Components.....	111
6.15.1	Delphi Component.....	111
6.15.1.1	Properties .....	112
6.15.1.2	Call .....	113
6.15.1.3	CallAsync .....	114
6.15.1.4	CallRPCAsync.....	114
6.15.1.5	CallList.....	114
6.15.1.6	CallRPCStr .....	114
6.15.1.7	CallStr.....	114

6.15.1.8	Connect.....	114
6.15.1.9	Disconnect.....	115
6.15.1.10	EventSubscribe .....	115
6.15.1.11	GetServerInfo .....	115
6.15.1.12	Lock.....	115
6.15.1.13	LockGlobal .....	115
6.15.1.14	RestoreState .....	116
6.15.1.15	SaveState.....	116
6.15.1.16	Unlock .....	116
6.15.2	RPC Parameters.....	116
6.15.2.1	Assign.....	116
6.15.2.2	Clear .....	116
6.15.2.3	Delete .....	117
6.15.2.4	Get.....	117
6.15.2.5	Get.....	117
6.15.2.6	Put .....	117
6.15.2.7	Put.....	117
6.15.2.8	SubscriptAt.....	117
6.15.2.9	ValueAt.....	118
<b>7.0</b>	<b>Site Context Object.....</b>	<b>119</b>
<b>8.0</b>	<b>User Context Object.....</b>	<b>122</b>
8.1	Introduction .....	122
8.2	Implementation and Maintenance .....	122
8.3	Routine Descriptions .....	122
8.4	File List.....	122
8.5	Cross References .....	122
8.6	Exported Options .....	123
8.7	Exported Security Keys.....	123
8.8	Exported Protocols.....	123
8.9	Exported Parameters .....	123
8.10	Exported Mail Groups .....	123
8.11	Callable Routines .....	123
8.12	RPC: BEHOUSCX USERINFO.....	123
8.12.1	\$\$ORDROLE^BEHOUSCX .....	124
8.12.2	\$\$ISPROV^BEHOUSCX .....	124
8.12.3	\$\$HASKEY^BEHOUSCX .....	124
8.12.4	RPC: BEHOUSCX HASKEYS.....	124
8.12.5	RPC: BEHOUSCX NEWPERS.....	125
8.12.6	\$\$ACTIVE^BEHOUSCX.....	125
8.12.7	RPC: BEHOUSCX VALIDSIG .....	125
8.12.8	RPC: BEHOUSCX VALIDPSW .....	126
8.12.9	RPC: BEHOUSCX HASFMCD .....	126
8.13	External Relations .....	126
8.14	Internal Relations .....	126
8.15	Archiving and Purging.....	126



8.16	Components.....	126
8.16.1	Properties .....	127
8.16.2	ESigValidate .....	127
8.16.3	HasKey .....	128
8.16.4	HasKeys .....	128
<b>9.0</b>	<b>Patient Context Object.....</b>	<b>129</b>
9.1	Introduction .....	129
9.2	Implementation and Maintenance .....	129
9.3	Routine Descriptions .....	129
9.4	File List.....	130
9.4.1	BEH PATIENT LIST (#90460.03) .....	130
9.5	Cross References .....	130
9.6	Exported Options .....	130
9.7	Exported Security Keys.....	131
9.8	Exported Protocols.....	131
9.9	Exported Parameters .....	131
9.10	Exported Mail Groups .....	132
9.11	Callable Routines.....	133
9.11.1	RPC: BEHOPTCX CHKDUP .....	133
9.11.2	\$\$HRN^BEHOPTCX.....	133
9.11.3	\$\$ICN^BEHOPTCX .....	133
9.11.4	RPC: BEHOPTCX ICN2DFN.....	133
9.11.5	RPC: BEHOPTCX INPLOC .....	134
9.11.6	\$\$ISACTIVE^BEHOPTCX .....	134
9.11.7	\$\$ISSENS^BEHOPTCX .....	134
9.11.8	RPC: BEHOPTCX LEGACY .....	135
9.11.9	RPC: BEHOPTCX PCDETAIL.....	135
9.11.10	RPC: BEHOPTCX PTINFO .....	135
9.11.11	RPC: BEHOPTCX PTINQ .....	136
9.11.12	\$\$SETCTX^BEHOPTCX .....	136
9.11.13	RPC: BEHOPTPC DETAIL.....	137
9.11.14	\$\$OUTPTPR^BEHOPTPC .....	137
9.11.15	\$\$OUTPTTM^BEHOPTPC.....	137
9.11.16	TEAM^BEHOPTPC .....	137
9.11.17	RPC: BEHOPTPL CLINRNG.....	137
9.11.18	RPC: BEHOPTPL DOBLKP .....	138
9.11.19	RPC: BEHOPTPL GETDFLT .....	138
9.11.20	RPC: BEHOPTPL HRNLKP .....	138
9.11.21	RPC: BEHOPTPL IENLKP .....	139
9.11.22	RPC: BEHOPTPL LISTALL .....	139
9.11.23	RPC: BEHOPTPL LISTINFO.....	139
9.11.24	RPC: BEHOPTPL LISTPTS .....	140
9.11.25	RPC: BEHOPTPL LISTSEL.....	140
9.11.26	RPC: BEHOPTPL LOOKUP .....	141
9.11.27	RPC: BEHOPTPL MANAGE .....	141

9.11.28	RPC: BEHOPTPL SAVEDFLT .....	141
9.11.29	\$\$\$SSN^BEHOPTPL.....	142
9.11.30	External Relations.....	142
9.12	Internal Relations .....	142
9.13	Archiving and Purging.....	142
9.14	Components.....	142
9.14.1	Properties .....	142
9.14.2	Clear .....	144
9.14.3	Detail.....	144
9.14.4	Select.....	144
<b>10.0</b>	<b>Encounter Context Object.....</b>	<b>145</b>
10.1	Introduction .....	145
10.2	Implementation and Maintenance .....	145
10.3	Routine Descriptions .....	145
10.4	File List.....	146
10.5	Cross References .....	146
10.6	Exported Options .....	146
10.7	Exported Security Keys.....	146
10.8	Exported Protocols.....	146
10.9	Exported Parameters .....	147
10.10	Exported Mail Groups .....	147
10.11	Callable Routines.....	148
10.11.1	\$\$ACTLOC^BEHOENCX .....	148
10.11.2	\$\$ADDP RV^BEHOENCX.....	148
10.11.3	RPC: BEHOENCX ADMITCUR .....	148
10.11.4	\$\$ADMITINF^BEHOENCX.....	149
10.11.5	RPC: BEHOENCX ADMITLST .....	149
10.11.6	RPC: BEHOENCX APPTLST .....	149
10.11.7	RPC: BEHOENCX CLINLOC .....	149
10.11.8	RPC: BEHOENCX INPLOC.....	150
10.11.9	RPC: BEHOENCX FETCH .....	150
10.11.10	\$\$FNDVIS^BEHOENCX.....	151
10.11.11	RPC: BEHOENCX GETPRV .....	151
10.11.12	RPC: BEHOENCX GETPRV2 .....	152
10.11.13	RPC: BEHOENCX GETVISIT.....	152
10.11.14	\$\$ISLOCKED^BEHOENCX.....	153
10.11.15	RPC: BEHOENCX LOCIEN.....	153
10.11.16	RPC: BEHOENCX LOCINFO .....	153
10.11.17	\$\$SC2LOC^ BEHOENCX.....	153
10.11.18	\$\$SETCTX^BEHOENCX.....	154
10.11.19	RPC: BEHOENCX VID2IEN .....	154
10.11.20	RPC: BEHOENCX VISITLST .....	154
10.11.21	\$\$VIS2VSTR^BEHOENCX.....	155
10.11.22	\$\$VSTR2VIS^BEHOENCX.....	155
10.12	External Relations .....	155

10.13	Internal Relations .....	155
10.14	Archiving and Purging .....	155
10.15	Components .....	155
10.15.1	Properties .....	156
10.15.2	EnsureHandle .....	156
10.15.3	Prepare .....	157
10.15.4	SelectLocation .....	157
<b>11.0</b>	<b>Patient Identification Header.....</b>	<b>158</b>
11.1	Introduction .....	158
11.2	Implementation and Maintenance .....	158
11.3	Routine Descriptions .....	158
11.4	File List.....	158
11.5	Cross References .....	159
11.6	Exported Options .....	159
11.7	Exported Security Keys.....	159
11.8	Exported Protocols.....	159
11.9	Exported Parameters .....	159
11.10	Exported Mail Groups .....	159
11.11	Callable Routines .....	159
11.12	External Relations .....	159
11.13	Internal Relations .....	159
11.14	Archiving and Purging .....	159
11.15	Components.....	159
11.15.1	Properties .....	160
<b>12.0</b>	<b>Encounter Information Header .....</b>	<b>161</b>
12.1	Introduction .....	161
12.2	Implementation and Maintenance .....	161
12.3	Routine Descriptions .....	161
12.4	File List.....	161
12.5	Cross References .....	162
12.6	Exported Options .....	162
12.7	Exported Security Keys.....	162
12.8	Exported Protocols.....	162
12.9	Exported Parameters .....	162
12.10	Exported Mail Groups .....	162
12.11	Callable Routines .....	162
12.12	External Relations .....	162
12.13	Internal Relations .....	162
12.14	Archiving and Purging .....	162
12.15	Components.....	162
12.15.1	Properties .....	163
<b>13.0</b>	<b>Vital Measurement Entry .....</b>	<b>164</b>
13.1	Introduction .....	164

13.2	Implementation and Maintenance .....	164
13.2.1	Vital Measurement Entry (service).....	164
13.2.2	Vital Measurement Entry (visual component) .....	165
13.3	Routine Descriptions .....	165
13.4	File List.....	165
13.4.1	BEH MEASUREMENT CONTROL (#90460.01) .....	166
13.5	Cross References .....	166
13.6	Exported Options .....	166
13.7	Exported Security Keys.....	166
13.8	Exported Protocols.....	166
13.9	Exported Parameters .....	167
13.10	Exported Mail Groups .....	167
13.11	Callable Routines .....	167
13.11.1	RPC: BEHOVM DETAIL .....	167
13.11.2	RPC: BEHOVM GRID.....	168
13.11.3	RPC: BEHOVM HELP .....	168
13.11.4	RPC: BEHOVM LASTVIT .....	168
13.11.5	RPC: BEHOVM LIST .....	169
13.11.6	RPC: BEHOVM PCTILE .....	169
13.11.7	RPC: BEHOVM SAVE .....	169
13.11.8	RPC: BEHOVM TEMPLATE.....	170
13.11.9	RPC: BEHOVM VALIDATE .....	170
13.12	External Relations .....	170
13.13	Internal Relations .....	170
13.14	Archiving and Purging .....	170
13.15	Components.....	171
13.15.1	Service.....	171
13.15.1.1	Properties .....	171
13.15.1.2	Execute .....	171
13.15.2	Visual Component .....	171
13.15.2.1	Properties .....	171
<b>14.0</b>	<b>Vital Measurement Display.....</b>	<b>174</b>
14.1	Introduction .....	174
14.2	Implementation and Maintenance .....	174
14.3	Routine Descriptions .....	175
14.4	File List.....	175
14.5	Cross References .....	175
14.6	Exported Options .....	175
14.7	Exported Security Keys.....	175
14.8	Exported Protocols.....	175
14.9	Exported Parameters .....	175
14.10	Exported Mail Groups .....	175
14.11	Callable Routines .....	175
14.12	External Relations .....	175
14.13	Internal Relations .....	176

14.14	Archiving and Purging .....	176
14.15	Components .....	176
<b>15.0</b>	<b>Activity Time.....</b>	<b>178</b>
15.1	Introduction .....	178
15.2	Implementation and Maintenance .....	178
15.3	Routine Descriptions .....	178
15.4	File List.....	179
15.5	Cross References .....	179
15.6	Exported Options .....	179
15.7	Exported Security Keys.....	179
15.8	Exported Protocols.....	179
15.9	Exported Parameters .....	179
15.10	Exported Mail Groups .....	179
15.11	Callable Routines .....	179
15.12	RPC: BGOVTM DEL .....	179
15.12.1	RPC: BGOVTM GET .....	180
15.12.2	RPC: BGOVTM SET.....	180
15.13	External Relations .....	180
15.14	Internal Relations .....	180
15.15	Archiving and Purging .....	180
15.16	Components.....	180
15.16.1	Properties .....	181
<b>16.0</b>	<b>Chief Complaint.....</b>	<b>182</b>
16.1	Introduction .....	182
16.2	Implementation and Maintenance .....	182
16.3	Routine Descriptions .....	183
16.4	File List.....	183
16.4.1	BGO CHIEF COMPLAINT PICK LIST (#90362.2).....	183
16.5	Cross References .....	183
16.6	Exported Options .....	183
16.7	Exported Security Keys.....	183
16.8	Exported Protocols.....	183
16.9	Exported Parameters .....	184
16.10	Exported Mail Groups .....	184
16.11	Callable Routines .....	184
16.11.1	RPC: BGOCC DEL .....	184
16.11.2	RPC: BGOCC DELPL.....	184
16.11.3	RPC: BGOCC GET .....	185
16.11.4	RPC: BGOCC GETPL .....	185
16.11.5	RPC: BGOCC SET .....	185
16.11.6	RPC: BGOCC SETPL.....	185
16.12	External Relations .....	186
16.13	Internal Relations .....	186
16.14	Archiving and Purging .....	186

16.15	Components.....	186
16.15.1	Properties .....	186
<b>17.0</b>	<b>Evaluation and Management Coding .....</b>	<b>187</b>
17.1	Introduction .....	187
17.2	Implementation and Maintenance .....	187
17.3	Routine Descriptions .....	188
17.4	File List.....	188
17.5	Cross References .....	188
17.6	Exported Options .....	188
17.7	Exported Security Keys.....	188
17.8	Exported Protocols.....	188
17.9	Exported Parameters .....	189
17.10	Exported Mail Groups .....	189
17.11	Callable Routines .....	189
17.12	External Relations .....	189
17.13	Internal Relations .....	189
17.14	Archiving and Purging .....	189
17.15	Components.....	189
17.15.1	Properties .....	190
<b>18.0</b>	<b>Exams.....</b>	<b>191</b>
18.1	Introduction .....	191
18.2	Implementation and Maintenance .....	191
18.3	Routine Descriptions .....	191
18.4	File List.....	192
18.5	Cross References .....	192
18.6	Exported Options .....	192
18.7	Exported Security Keys.....	192
18.8	Exported Protocols.....	192
18.9	Exported Parameters .....	192
18.10	Exported Mail Groups .....	192
18.11	Callable Routines .....	192
18.11.1	RPC: BGOVEXAM DEL.....	192
18.11.2	RPC: BGOVEXAM GET .....	193
18.11.3	RPC: BGOVEXAM GETTYPES .....	193
18.11.4	RPC: BGOVEXAM PRIPRV .....	193
18.11.5	RPC: BGOVEXAM SET.....	193
18.12	External Relations .....	194
18.13	Internal Relations .....	194
18.14	Archiving and Purging .....	194
18.15	Components.....	194
18.15.1	Properties .....	194
<b>19.0</b>	<b>Health Factors .....</b>	<b>196</b>
19.1	Introduction .....	196

19.2	Implementation and Maintenance .....	196
19.3	Routine Descriptions .....	196
19.4	File List.....	197
19.5	Cross References .....	197
19.6	Exported Options .....	197
19.7	Exported Security Keys.....	197
19.8	Exported Protocols.....	197
19.9	Exported Parameters .....	197
19.10	Exported Mail Groups .....	197
19.11	Callable Routines .....	197
19.11.1	RPC: BGOVHF DEL .....	197
19.11.2	RPC: BGOVHF GET.....	198
19.11.3	RPC: BGOVHF GETTYPES.....	198
19.11.4	RPC: BGOVHF REFLIST .....	198
19.11.5	RPC: BGOVHF SET .....	199
19.11.6	RPC: BGOVHF SETREF .....	199
19.12	External Relations .....	199
19.13	Internal Relations .....	199
19.14	Archiving and Purging.....	199
19.15	Components.....	199
19.15.1	Properties .....	200
<b>20.0</b>	<b>ICD Pick List .....</b>	<b>201</b>
20.1	Introduction .....	201
20.2	Implementation and Maintenance .....	201
20.3	Routine Descriptions .....	201
20.4	File List.....	202
20.4.1	BGO ICD PREFERENCES (#90362.35) .....	202
20.4.1.1	ICD DIAGNOSIS subfile (#90362.351) .....	202
20.4.1.2	MANAGERS subfile (#90362.352) .....	202
20.2	Cross References .....	202
20.3	Exported Options .....	203
20.4	Exported Security Keys.....	203
20.5	Exported Protocols.....	203
20.6	Exported Parameters .....	203
20.7	Exported Mail Groups .....	203
20.8	Callable Routines .....	203
20.8.1	RPC: BGOICDLK ICDLKUP .....	203
20.8.2	RPC: BGOICDPR CLONE.....	203
20.8.3	RPC: BGOICD CLONEOTH .....	204
20.8.4	RPC: BGOICDPR GETCATS .....	204
20.8.5	RPC: BGOICDPR GETITEMS.....	204
20.8.6	RPC: BGOICDPR GETLNAME .....	204
20.8.7	RPC: BGOICDPR GETMGRS.....	205
20.8.8	RPC: BGOICDPR OTHCATS.....	205
20.8.9	RPC: BGOICDPR QUERY .....	205

20.8.10	RPC: BGOICDPR SETCAT .....	205
20.8.11	RPC: BGOICDPR SETFREQ .....	206
20.8.12	RPC: BGOICDPR SETITEM .....	206
20.8.13	RPC: BGOICDPR SETMGR.....	206
20.8.14	RPC: BGOICDPR SETNAME.....	206
20.9	External Relations .....	207
20.10	Internal Relations .....	207
20.11	Archiving and Purging .....	207
20.12	Components.....	207
20.12.1	Properties .....	207
<b>21.0</b>	<b>Immunizations .....</b>	<b>208</b>
21.1	Introduction .....	208
21.2	Implementation and Maintenance .....	208
21.3	Routine Descriptions .....	208
21.4	File List.....	209
21.5	Cross References .....	209
21.6	Exported Options .....	209
21.7	Exported Security Keys.....	209
21.8	Exported Protocols.....	209
21.9	Exported Parameters .....	209
21.10	Exported Mail Groups .....	209
21.11	Callable Routines .....	210
21.11.1	RPC: BGOVIMM DEL.....	210
21.11.2	RPC: BGOVIMM DELCONT.....	210
21.11.3	RPC: BGOVIMM GET .....	210
21.11.4	RPC: BGOVIMM GETCASE.....	211
21.11.5	RPC: BGOVIMM GETCONT .....	211
21.11.6	RPC: BGOVIMM LOADIMM .....	212
21.11.7	RPC: BGOVIMM LOT .....	212
21.11.8	RPC: BGOVIMM PRINT .....	212
21.11.9	RPC: BGOVIMM PRIPRV .....	212
21.11.10	RPC: BGOVIMM PROFILE .....	213
21.11.11	RPC: BGOVIMM SET .....	213
21.11.12	RPC: BGOVIMM SETCONT.....	213
21.11.13	RPC: BGOVIMM SETREG .....	213
21.12	External Relations .....	214
21.13	Internal Relations .....	214
21.14	Archiving and Purging .....	214
21.15	Components.....	214
21.15.1	Properties .....	214
<b>22.0</b>	<b>Superbill.....</b>	<b>216</b>
22.1	Introduction .....	216
22.2	Implementation and Maintenance .....	216
22.3	Routine Descriptions .....	216



22.4	File List.....	217
22.4.1	BGO CPT PREFERENCES (#90362.31) .....	217
22.4.1.1	CPT subfile (#90362.312) .....	217
22.4.1.2	ASSOCIATIONS subfile (#90362.3121).....	218
22.4.1.3	MANAGERS subfile (#90362.313) .....	218
22.2	Cross References .....	218
22.3	Exported Options .....	218
22.4	Exported Security Keys.....	218
22.5	Exported Protocols.....	218
22.6	Exported Parameters .....	219
22.7	Exported Mail Groups .....	219
22.8	Callable Routines.....	219
22.8.1	RPC: BGOCTPR CLONE.....	219
22.8.2	RPC: BGOCTPR CLONEOTH.....	219
22.8.3	RPC: BGOCTPR DELASSOC .....	219
22.8.4	RPC: BGOCTPR GETASSOC .....	220
22.8.1.1	RPC: BGOCTPR GETCATS.....	220
22.1.2	RPC: BGOCTPR GETITEMS.....	220
22.1.3	RPC: BGOCTPR GETLNAME .....	220
22.1.4	RPC: BGOCTPR GETMGRS.....	221
22.1.5	RPC: BGOCTPR OTHCATS.....	221
22.1.6	RPC: BGOCTPR QUERY .....	221
22.1.7	RPC: BGOCTPR SETACHK.....	221
22.1.8	RPC: BGOCTPR SETASSOC .....	222
22.1.9	RPC: BGOCTPR SETCAT .....	222
22.1.10	RPC: BGOCTPR SETFREQ.....	222
22.1.11	RPC: BGOCTPR SETITEM .....	222
22.1.12	RPC: BGOCTPR SETMGR.....	223
22.1.13	RPC: BGOCTPR SETNAME.....	223
22.1.14	RPC: BGOCTPR VSTASSOC .....	223
22.2	External Relations .....	223
22.3	Internal Relations .....	224
22.4	Archiving and Purging.....	224
22.5	Components.....	224
22.5.1	Properties .....	224
<b>23.0</b>	<b>Patient Education.....</b>	<b>226</b>
23.1	Introduction .....	226
23.2	Implementation and Maintenance .....	226
23.3	Routine Descriptions .....	226
23.4	File List.....	227
23.4.1	BGO ED TOPIC PREFERENCES (#90362.36) .....	227
23.4.1.1	ICD DIAGNOSIS subfile (#90362.361) .....	227
23.4.1.2	MANAGERS subfile (#90362.362) .....	227
23.5	Cross References .....	228
23.6	Exported Options .....	228

23.7	Exported Security Keys.....	228
23.8	Exported Protocols.....	228
23.9	Exported Parameters.....	228
23.10	Exported Mail Groups.....	228
23.11	Callable Routines.....	228
23.11.1	RPC: BGOEDTPR CLONE.....	228
23.11.2	RPC: BGOEDTPR GETCATS.....	228
23.11.3	RPC: BGOEDTPR GETITEMS.....	229
23.11.4	RPC: BGOEDTPR GETLNAME.....	229
23.11.5	RPC: BGOEDTPR GETMGRS.....	229
23.11.6	RPC: BGOEDTPR QUERY.....	229
23.11.7	RPC: BGOEDTPR SETCAT.....	230
23.11.8	RPC: BGOEDTPR SETFREQ.....	230
23.11.9	RPC: BGOEDTPR SETITEM.....	230
23.11.10	RPC: BGOEDTPR SETMGR.....	231
23.11.11	RPC: BGOEDTPR SETNAME.....	231
23.11.12	RPC: BGOVPED DEL.....	231
23.11.13	RPC: BGOVPED GET.....	231
23.11.14	RPC: BGOVPED GETNAME.....	232
23.11.15	RPC: BGOVPED GETOS.....	232
23.11.16	RPC: BGOVPED GETTOPIC.....	232
23.11.17	RPC: BGOVPED GETTYPES.....	232
23.11.18	RPC: BGOVPED PRIPRV.....	233
23.11.19	RPC: BGOVPED SET.....	233
23.11.20	RPC: BGOVPED SETDXTOP.....	233
23.11.21	RPC: BGOVPED SETPXTOP.....	233
23.12	External Relations.....	234
23.13	Internal Relations.....	234
23.14	Archiving and Purging.....	234
23.15	Components.....	234
23.15.1	Properties.....	234
<b>24.0</b>	<b>POV History.....</b>	<b>236</b>
24.1	Introduction.....	236
24.2	Implementation and Maintenance.....	236
24.3	Routine Descriptions.....	236
24.4	File List.....	236
24.5	Cross References.....	237
24.6	Exported Options.....	237
24.7	Exported Security Keys.....	237
24.8	Exported Protocols.....	237
24.9	Exported Parameters.....	237
24.10	Exported Mail Groups.....	237
24.11	Callable Routines.....	237
24.12	External Relations.....	237
24.13	Internal Relations.....	237

24.14	Archiving and Purging .....	237
24.15	Components .....	238
24.15.1	Properties .....	238
<b>25.0</b>	<b>Procedure Viewer .....</b>	<b>239</b>
25.1	Introduction .....	239
25.2	Implementation and Maintenance .....	239
25.3	Routine Descriptions .....	239
25.4	File List .....	239
25.5	Cross References .....	240
25.6	Exported Options .....	240
25.7	Exported Security Keys .....	240
25.8	Exported Protocols .....	240
25.9	Exported Parameters .....	240
25.10	Exported Mail Groups .....	240
25.11	Callable Routines .....	240
25.12	External Relations .....	240
25.13	Internal Relations .....	240
25.14	Archiving and Purging .....	240
25.15	Components .....	241
25.15.1	Properties .....	241
<b>26.0</b>	<b>Personal Health History .....</b>	<b>242</b>
26.1	Introduction .....	242
26.2	Implementation and Maintenance .....	242
26.3	Routine Descriptions .....	242
26.4	File List .....	243
26.5	Cross References .....	243
26.6	Exported Options .....	243
26.7	Exported Security Keys .....	243
26.8	Exported Protocols .....	243
26.9	Exported Parameters .....	244
26.10	Exported Mail Groups .....	244
26.11	Callable Routines .....	244
26.11.1	RPC: BGOBMSR DEL .....	244
26.11.2	RPC: BGOBMSR GET .....	244
26.11.3	RPC: BGOBMSR SET .....	244
26.11.4	RPC: BGOREF DEL .....	245
26.11.5	RPC: BGOREF GET .....	245
26.11.6	RPC: BGOREF REFLIST .....	245
26.11.7	RPC: BGOREF SET .....	245
26.11.8	RPC: BGOREP DEL .....	246
26.11.9	RPC: BGOREP GET .....	246
26.11.10	RPC: BGOREP SET .....	246
26.11.11	RPC: BGOVAST DEL .....	246
26.11.12	RPC: BGOVAST GET .....	247

26.11.13	RPC: BGOVAST GETNOTE .....	247
26.11.14	RPC: BGOVAST GETREG .....	247
26.11.15	RPC: BGOVAST SET .....	248
26.11.16	RPC: BGOVAST SETREG .....	248
26.11.17	RPC: BGOVELD DEL .....	248
26.11.18	RPC: BGOVELD GET .....	248
26.11.19	RPC: BGOVELD SET .....	249
26.11.20	RPC: BGOVIF DEL .....	249
26.11.21	RPC: BGOVIF GET .....	249
26.11.22	RPC: BGOVIF SET .....	250
26.11.23	RPC: BGOVIXC DEL .....	250
26.11.24	RPC: BGOVIXC GET .....	250
26.11.25	RPC: BGOVIXC SET .....	250
26.12	External Relations .....	251
26.13	Internal Relations .....	251
26.14	Archiving and Purging .....	251
26.15	Components .....	251
26.15.1	Properties .....	251
<b>27.0</b>	<b>Skin Tests .....</b>	<b>253</b>
27.1	Introduction .....	253
27.2	Implementation and Maintenance .....	253
27.3	Routine Descriptions .....	253
27.4	File List .....	253
27.5	Cross References .....	254
27.6	Exported Options .....	254
27.7	Exported Security Keys .....	254
27.8	Exported Protocols .....	254
27.9	Exported Parameters .....	254
27.10	Exported Mail Groups .....	254
27.11	Callable Routines .....	254
27.11.1	BGOVSK DEL .....	254
27.11.2	BGOVSK GET .....	254
27.11.3	BGOVSK SET .....	255
27.12	External Relations .....	255
27.13	Internal Relations .....	255
27.14	Archiving and Purging .....	255
27.15	Components .....	255
27.15.1	Properties .....	256
<b>28.0</b>	<b>VCPT .....</b>	<b>257</b>
28.1	Introduction .....	257
28.2	Implementation and Maintenance .....	257
28.3	Routine Descriptions .....	257
28.4	File List .....	258
28.5	Cross References .....	258

28.6	Exported Options .....	258
28.7	Exported Security Keys.....	258
28.8	Exported Protocols.....	258
28.9	Exported Parameters .....	258
28.10	Exported Mail Groups .....	258
28.11	Callable Routines.....	258
28.11.1	RPC: BGOVCPT CPTLKUP .....	259
28.11.2	RPC: BGOVCPT DEL.....	259
28.11.3	RPC: BGOVCPT GET .....	259
28.11.4	RPC: BGOVCPT GETIEN .....	259
28.11.5	RPC: BGOVCPT IMMCK.....	260
28.11.6	RPC: BGOVCPT MODLKUP.....	260
28.11.7	RPC: BGOVCPT SET.....	260
28.11.8	RPC: BGOVCPT SETDX.....	260
28.11.9	RPC: BGOVCPT SETQTY .....	261
28.12	External Relations .....	261
28.13	Internal Relations .....	261
28.14	Archiving and Purging.....	261
28.15	Components.....	261
28.15.1	Properties .....	262
<b>29.0</b>	<b>Visit Diagnosis (VPOV).....</b>	<b>263</b>
29.1	Introduction .....	263
29.2	Implementation and Maintenance .....	263
29.3	Routine Descriptions .....	263
29.4	File List.....	263
29.5	Cross References .....	264
29.6	Exported Options .....	264
29.7	Exported Security Keys.....	264
29.8	Exported Protocols.....	264
29.9	Exported Parameters .....	264
29.10	Exported Mail Groups .....	264
29.11	Callable Routines .....	264
29.11.1	RPC: BGOVPOV CHECK.....	264
29.11.2	RPC: BGOVPOV CHRTREVIEW .....	264
29.11.3	RPC: BGOVPOV CKSIGNBY.....	265
29.11.4	RPC: BGOVPOV DEL .....	265
29.11.5	RPC: BGOVPOV GET.....	265
29.11.6	RPC: BGOVPOV GETCODE .....	266
29.11.7	RPC: BGOVPOV GETICD.....	266
29.11.8	RPC: BGOVPOV RECENT .....	266
29.11.9	RPC: BGOVPOV SET .....	267
29.11.10	RPC: BGOVPOV SETPRI .....	267
29.11.11	RPC: BGOVPOV TELEPHON.....	267
29.12	External Relations .....	267
29.13	Internal Relations .....	267

29.14	Archiving and Purging .....	267
29.15	Components .....	268
29.15.1	Properties .....	268
<b>30.0</b>	<b>Problem Management .....</b>	<b>269</b>
30.1	Introduction .....	269
30.2	Implementation and Maintenance .....	269
30.3	Routine Descriptions .....	269
30.4	File List.....	269
30.4.1	BGO PROBLEM PRIORITY (#90362.22).....	270
30.5	Cross References .....	270
30.6	Exported Options .....	270
30.7	Exported Security Keys.....	270
30.8	Exported Protocols.....	270
30.9	Exported Parameters .....	270
30.10	Exported Mail Groups .....	270
30.11	Callable Routines .....	271
30.11.1	BGOPRBN DEL.....	271
30.11.2	BGOPRBN GET .....	271
30.11.3	BGOPRBN SET .....	271
30.11.4	BGOPROB CKID .....	271
30.11.5	BGOPROB DEL.....	272
30.11.6	BGOPROB GET .....	272
30.11.7	BGOPROB NEXTID .....	272
30.11.8	BGOPROB SET.....	272
30.11.9	BGOPROB SETPRI.....	273
30.12	External Relations .....	273
30.13	Internal Relations .....	273
30.14	Archiving and Purging .....	273
30.15	Components.....	273
30.15.1	Properties .....	273
<b>31.0</b>	<b>Problems (CPRS) .....</b>	<b>275</b>
31.1	Introduction .....	275
31.2	Implementation and Maintenance .....	275
31.3	Routine Descriptions .....	276
31.4	File List.....	276
31.5	Cross References .....	276
31.6	Exported Options .....	276
31.7	Exported Security Keys.....	276
31.8	Exported Protocols.....	276
31.9	Exported Parameters .....	276
31.10	Exported Mail Groups .....	276
31.11	Callable Routines .....	276
31.12	External Relations .....	276
31.13	Internal Relations .....	277

31.14	Archiving and Purging .....	277
31.15	Components .....	277
31.15.1	Properties .....	277
<b>32.0</b>	<b>Problem List .....</b>	<b>279</b>
32.1	Introduction .....	279
32.2	Implementation and Maintenance .....	279
32.3	Routine Descriptions .....	280
32.4	Cross References .....	280
32.5	Exported Options .....	280
32.6	Exported Security Keys .....	280
32.7	Exported Protocols .....	280
32.8	Exported Parameters .....	280
32.9	Exported Mail Groups .....	280
32.10	Callable Routines .....	280
32.10.1	RPC: BEHOPLCV DETAIL .....	280
32.10.2	RPC: BEHOPLCV LIST .....	281
32.11	External Relations .....	281
32.12	Internal Relations .....	281
32.13	Archiving and Purging .....	281
32.14	Components .....	281
32.14.1	Properties .....	281
<b>33.0</b>	<b>Consults (CPRS) .....</b>	<b>284</b>
33.1	Introduction .....	284
33.2	Implementation and Maintenance .....	284
33.3	Routine Descriptions .....	285
33.4	File List .....	285
33.5	Cross References .....	285
33.6	Exported Options .....	285
33.7	Exported Security Keys .....	289
33.8	Exported Protocols .....	289
33.9	Exported Parameters .....	289
33.10	Exported Mail Groups .....	289
33.11	Callable Routines .....	289
33.12	External Relations .....	290
33.13	Internal Relations .....	290
33.14	Archiving and Purging .....	290
33.15	Components .....	290
33.15.1	Properties .....	290
<b>34.0</b>	<b>Discharge Summary (CPRS) .....</b>	<b>292</b>
34.1	Introduction .....	292
34.2	Implementation and Maintenance .....	292
34.3	Routine Descriptions .....	293
34.4	File List .....	293

34.5	Cross References .....	293
34.6	Exported Options .....	293
34.6.1	Exported Security Keys .....	293
34.7	Exported Protocols.....	293
34.8	Exported Parameters .....	293
34.9	Exported Mail Groups .....	293
34.10	Callable Routines .....	293
34.11	External Relations .....	294
34.12	Internal Relations .....	294
34.13	Archiving and Purging.....	294
34.13.1	Components .....	294
34.13.2	Properties .....	294
<b>35.0</b>	<b>Progress Notes.....</b>	<b>296</b>
35.1	Introduction .....	296
35.2	Implementation and Maintenance .....	296
35.3	Routine Descriptions .....	297
35.4	File List.....	297
35.5	Cross References .....	297
35.6	Exported Options .....	297
35.7	Exported Security Keys.....	298
35.8	Exported Protocols.....	298
35.9	Exported Parameters .....	298
35.10	Exported Mail Groups .....	298
35.11	Callable Routines .....	298
35.12	External Relations .....	299
35.13	Internal Relations .....	299
35.14	Archiving and Purging.....	299
35.15	Components.....	299
35.15.1	Properties .....	299
<b>36.0</b>	<b>Dictate Notes .....</b>	<b>301</b>
36.1	Introduction .....	301
36.2	Implementation and Maintenance .....	301
36.3	Routine Descriptions .....	301
36.4	File List.....	302
36.5	Cross References .....	302
36.6	Exported Options .....	302
36.7	Exported Security Keys.....	302
36.8	Exported Protocols.....	302
36.9	Exported Parameters .....	302
36.10	Exported Mail Groups .....	303
36.11	Callable Routines .....	303
36.11.1	BATCH^BEHODC.....	303
36.11.2	GETFILE^BEHODC.....	303
36.11.3	EN^BEHODC6.....	303



36.11.4	PID^BEHODC6.....	303
36.11.5	KIL^BEHODC6 .....	303
36.11.6	PROCESS^BEHODC7 .....	304
36.11.7	OBX^BEHODC7 .....	304
36.11.8	\$\$AGTEXT^BEHODC7 .....	304
36.11.9	BOTH^BEHODC8.....	304
36.12	GENACK^BEHODC8.....	304
36.13	External Relations .....	304
36.14	Internal Relations .....	304
36.15	Archiving and Purging .....	305
36.16	Components.....	305
36.16.1	Properties .....	305
<b>37.0</b>	<b>Medication Management .....</b>	<b>307</b>
37.1	Introduction .....	307
37.2	Implementation and Maintenance .....	307
37.3	Routine Descriptions .....	308
37.4	File List.....	308
37.5	Cross References .....	308
37.6	Exported Options .....	308
37.7	Exported Security Keys.....	308
37.8	Exported Protocols.....	308
37.9	Exported Parameters .....	308
37.10	Exported Mail Groups .....	309
37.11	Callable Routines .....	309
37.11.1	\$\$GETCMF1^BEHORXFN .....	309
37.11.2	RPC: BEHORXFN GETRXS .....	309
37.11.3	RPC: BEHORXFN SETCMF .....	310
37.12	External Relations .....	310
37.13	Internal Relations .....	310
37.14	Archiving and Purging .....	310
37.15	Components.....	310
37.15.1	Properties .....	310
<b>38.0</b>	<b>Orders .....</b>	<b>312</b>
38.1	Introduction .....	312
38.2	Implementation and Maintenance .....	312
38.3	Routine Descriptions .....	312
38.4	File List.....	313
38.5	Cross References .....	313
38.6	Exported Options .....	313
38.7	Exported Security Keys.....	319
38.8	Exported Protocols.....	319
38.9	Exported Parameters .....	319
38.10	Exported Mail Groups .....	319
38.11	Callable Routines .....	319

38.12	External Relations .....	319
38.13	Internal Relations .....	319
38.14	Archiving and Purging .....	319
38.15	Components .....	320
38.15.1	Properties .....	320
<b>39.0</b>	<b>Quick Order Wizard.....</b>	<b>322</b>
39.1	Introduction .....	322
39.2	Implementation and Maintenance .....	322
39.3	Routine Descriptions .....	323
39.4	File List.....	323
39.5	Cross References .....	323
39.6	Exported Options .....	323
39.7	Exported Security Keys.....	323
39.8	Exported Protocols.....	323
39.9	Exported Parameters .....	323
39.10	Exported Mail Groups .....	323
39.11	Callable Routines .....	323
39.11.1	BEHOQOW CANDEL .....	323
39.11.2	BEHOQOW CLONE .....	324
39.11.3	BEHOQOW DEFDISGP .....	324
39.11.4	BEHOQOW DELETEQO .....	324
39.11.5	BEHOQOW DISGRP .....	324
39.11.6	BEHOQOW GETDISAB .....	325
39.11.7	BEHOQOW GETPKG.....	325
39.11.8	BEHOQOW GRPDEFWD.....	325
39.11.9	BEHOQOW PROPERTY .....	325
39.11.10	BEHOQOW QOFVAL .....	326
39.11.11	BEHOQOW QOITEMS .....	326
39.11.12	BEHOQOW SETDISAB.....	326
39.11.13	BEHOQOW UPDRSP.....	326
39.12	External Relations .....	327
39.13	Internal Relations .....	327
39.14	Archiving and Purging .....	327
39.15	Components.....	327
39.15.1	Properties .....	327
<b>40.0</b>	<b>Consult Order History.....</b>	<b>329</b>
40.1	Introduction .....	329
40.2	Implementation and Maintenance .....	329
40.3	Routine Descriptions .....	329
40.4	File List.....	330
40.5	Cross References .....	330
40.6	Exported Options .....	330
40.7	Exported Security Keys.....	330
40.8	Exported Protocols.....	330

40.9	Exported Parameters .....	330
40.10	Exported Mail Groups .....	330
40.11	Callable Routines .....	330
40.11.1	RPC: BEHOCNCV DETAIL .....	330
40.11.2	RPC: BEHOCNCV LIST .....	331
40.12	External Relations .....	331
40.13	Internal Relations .....	331
40.14	Archiving and Purging .....	331
40.15	Components .....	331
40.15.1	Properties .....	331
<b>41.0</b>	<b>Lab Orders .....</b>	<b>333</b>
41.1	Introduction .....	333
41.2	Implementation and Maintenance .....	333
41.3	Routine Descriptions .....	333
41.4	File List .....	334
41.5	Cross References .....	334
41.6	Exported Options .....	334
41.7	Exported Security Keys .....	334
41.8	Exported Protocols .....	334
41.9	Exported Parameters .....	334
41.10	Exported Mail Groups .....	334
41.11	Callable Routines .....	334
41.11.1	RPC: BEHOLRCV DETAIL .....	334
41.11.2	RPC: BEHOLRCV LIST .....	335
41.12	External Relations .....	335
41.13	Internal Relations .....	335
41.14	Archiving and Purging .....	335
41.15	Components .....	335
41.15.1	Properties .....	335
<b>42.0</b>	<b>Medications .....</b>	<b>338</b>
42.1	Introduction .....	338
42.2	Implementation and Maintenance .....	338
42.3	Routine Descriptions .....	339
42.4	File List .....	339
42.5	Cross References .....	339
42.6	Exported Options .....	339
42.7	Exported Security Keys .....	339
42.8	Exported Protocols .....	339
42.9	Exported Parameters .....	339
42.10	Exported Mail Groups .....	339
42.11	Callable Routines .....	339
42.11.1	RPC: BEHORXCV DETAIL .....	339
42.11.2	RPC: BEHORXCV LIST .....	340
42.12	External Relations .....	340

42.13	Internal Relations .....	340
42.14	Archiving and Purging .....	340
42.15	Components.....	340
42.15.1	Properties .....	340
<b>43.0</b>	<b>Health Summary Report .....</b>	<b>343</b>
43.1	Introduction .....	343
43.2	Implementation and Maintenance .....	343
43.3	Routine Descriptions .....	343
43.4	File List.....	343
43.5	Cross References .....	344
43.6	Exported Options .....	344
43.7	Exported Security Keys.....	344
43.8	Exported Protocols.....	344
43.9	Exported Parameters .....	344
43.10	Exported Mail Groups .....	344
43.11	Callable Routines .....	344
43.12	External Relations .....	344
43.13	Internal Relations .....	344
43.14	Archiving and Purging .....	344
43.15	Components.....	344
43.15.1	Properties .....	345
<b>44.0</b>	<b>Lab Results.....</b>	<b>347</b>
44.1	Introduction .....	347
44.2	Implementation and Maintenance .....	347
44.3	Routine Descriptions .....	347
44.4	File List.....	348
44.5	Cross References .....	348
44.6	Exported Options .....	348
44.7	Exported Security Keys.....	348
44.8	Exported Protocols.....	348
44.9	Exported Parameters .....	348
44.10	Exported Mail Groups .....	348
44.11	Callable Routines .....	348
44.12	External Relations .....	348
44.13	Internal Relations .....	348
44.14	Archiving and Purging .....	348
44.15	Components.....	349
44.15.1	Properties .....	349
<b>45.0</b>	<b>Remote Data (CPRS).....</b>	<b>351</b>
45.1	Introduction .....	351
45.2	Implementation and Maintenance .....	351
45.3	Routine Descriptions .....	351
45.4	File List.....	351

45.5	Cross References .....	351
45.6	Exported Options .....	352
45.7	Exported Security Keys.....	352
45.8	Exported Protocols.....	352
45.9	Exported Parameters .....	352
45.10	Exported Mail Groups .....	352
45.11	Callable Routines .....	352
45.12	External Relations .....	352
45.13	Internal Relations .....	352
45.14	Archiving and Purging.....	352
45.15	Components.....	352
45.15.1	Properties .....	352
<b>46.0</b>	<b>Remote Sites Service.....</b>	<b>354</b>
46.1	Introduction .....	354
46.2	Implementation and Maintenance .....	354
46.3	Routine Descriptions .....	354
46.4	File List.....	354
46.5	Cross References .....	354
46.6	Exported Options .....	354
46.7	Exported Security Keys.....	355
46.8	Exported Protocols.....	355
46.9	Exported Parameters .....	355
46.10	Exported Mail Groups .....	355
46.11	Callable Routines .....	355
46.11.1	RPC: BEHORDV DIRECT .....	355
46.12	External Relations .....	355
46.13	Internal Relations .....	355
46.14	Archiving and Purging.....	355
46.15	Components.....	356
46.15.1	IRemoteSites .....	356
46.15.1.1	Properties .....	356
46.15.1.2	CallRemote.....	356
46.15.1.3	GetSiteByID.....	356
46.15.1.4	Reset .....	356
46.15.1.5	Select .....	356
46.15.2	IRemoteSite .....	357
46.15.2.1	Properties .....	357
46.15.2.2	CallRemote.....	357
46.15.2.3	GetReportBySignature .....	357
46.15.2.4	Reset .....	357
46.15.3	IRemoteReport .....	357
46.15.3.1	Properties .....	357
46.15.4	IRemoteReport2 .....	358
46.15.4.1	CallRemote.....	358
46.15.5	IRemoteCallback .....	358

46.15.5.1	RemoteResults .....	358
<b>47.0</b>	<b>Reports (CPRS) .....</b>	<b>359</b>
47.1	Introduction .....	359
47.2	Implementation and Maintenance .....	359
47.3	Routine Descriptions .....	359
47.4	File List.....	360
47.5	Cross References .....	360
47.6	Exported Options .....	360
47.7	Exported Security Keys.....	368
47.8	Exported Protocols.....	368
47.9	Exported Parameters .....	368
47.10	Exported Mail Groups .....	368
47.11	Callable Routines .....	368
47.12	External Relations .....	369
47.13	Internal Relations .....	369
47.14	Archiving and Purging.....	369
47.15	Components.....	369
47.15.1	Properties .....	369
<b>48.0</b>	<b>Triage Summary .....</b>	<b>371</b>
48.1	Introduction .....	371
48.2	Implementation and Maintenance .....	371
48.3	Routine Descriptions .....	371
48.4	File List.....	371
48.5	Cross References .....	372
48.6	Exported Options .....	372
48.7	Exported Security Keys.....	372
48.8	Exported Protocols.....	372
48.9	Exported Parameters .....	372
48.10	Exported Mail Groups .....	372
48.11	Callable Routines.....	372
48.11.1	RPC: BGOTRG GETSUM .....	372
48.12	External Relations .....	372
48.13	Internal Relations .....	373
48.14	Archiving and Purging.....	373
48.15	Components.....	373
<b>49.0</b>	<b>Patient Detail View .....</b>	<b>375</b>
49.1	Introduction .....	375
49.2	Implementation and Maintenance .....	375
49.3	Routine Descriptions .....	375
49.4	File List.....	375
49.5	Cross References .....	375
49.6	Exported Options .....	376
49.7	Exported Security Keys.....	376

49.8	Exported Protocols .....	376
49.9	Exported Parameters .....	376
49.10	Exported Mail Groups .....	376
49.11	Callable Routines .....	376
49.12	External Relations .....	376
49.13	Internal Relations .....	376
49.14	Archiving and Purging .....	376
49.15	Components .....	376
49.15.1	Properties .....	376
<b>50.0</b>	<b>Notifications .....</b>	<b>378</b>
50.1	Introduction .....	378
50.2	Implementation and Maintenance .....	378
50.3	Routine Descriptions .....	379
50.4	File List.....	379
50.4.1	BEH ALERT CONTROL (#90460.021).....	379
50.4.2	BEH ALERT SCHEDULING (#90460.022).....	379
50.4.2.1	RECIPIENT (#90460.0221).....	379
50.5	Cross References .....	380
50.6	Exported Options .....	380
50.7	Exported Security Keys.....	383
50.8	Exported Protocols.....	384
50.9	Exported Parameters .....	384
50.10	Exported Mail Groups .....	384
50.11	Callable Routines .....	384
50.11.1	RPC: BEHOXQ ALRLIST .....	384
50.11.2	RPC: BEHOXQ ALRMSG.....	385
50.11.3	RPC: BEHOXQ ALRPP .....	385
50.11.4	RPC: BEHOXQ FORWARD .....	385
50.11.5	RPC: BEHOXQ SCHALR .....	385
50.11.6	RPC: BEHOXQ SCHDEL .....	385
50.11.7	RPC: BEHOXQ SCHLIST.....	386
50.11.8	RPC: BEHOXQ SCHMSG .....	386
50.11.9	RPC: BEHOXQ SCHRECIP .....	386
50.11.10	RPC: BEHOXQPC NOEMC .....	386
50.11.11	RPC: BEHOXQPC NOPOV .....	386
50.12	External Relations .....	387
50.13	Internal Relations .....	387
50.14	Archiving and Purging .....	387
50.15	Components.....	387
50.15.1	Properties .....	387
50.15.2	Schedule.....	388
<b>51.0</b>	<b>Alerts.....</b>	<b>389</b>
51.1	Introduction .....	389
51.2	Implementation and Maintenance .....	389

51.3	Routine Descriptions .....	390
51.4	File List.....	390
51.5	Cross References .....	390
51.6	Exported Options .....	390
51.7	Exported Security Keys.....	390
51.8	Exported Protocols.....	390
51.9	Exported Parameters .....	390
51.10	Exported Mail Groups .....	390
51.11	Callable Routines .....	390
51.11.1	RPC: BEHOXQCV DETAIL .....	390
51.11.2	RPC: BEHOXQCV LIST .....	391
51.12	External Relations .....	391
51.13	Internal Relations .....	391
51.14	Archiving and Purging .....	391
51.15	Components.....	391
51.15.1	Properties .....	391
<b>52.0</b>	<b>Crisis Alerts .....</b>	<b>394</b>
52.1	Introduction .....	394
52.2	Implementation and Maintenance .....	394
52.3	Routine Descriptions .....	394
52.4	File List.....	395
52.5	Cross References .....	395
52.6	Exported Options .....	395
52.7	Exported Security Keys.....	395
52.8	Exported Protocols.....	395
52.9	Exported Parameters .....	395
52.10	Exported Mail Groups .....	395
52.11	Callable Routines .....	395
52.11.1	RPC: EHOACV CWAD .....	395
52.11.2	RPC: EHOACV DETAIL.....	396
52.11.3	RPC: BEHOACV LIST .....	396
52.12	External Relations .....	396
52.13	Internal Relations .....	396
52.14	Archiving and Purging .....	396
52.15	Components.....	396
52.15.1	Properties .....	396
<b>53.0</b>	<b>Crises/Warnings/Alerts/Directives (CWAD).....</b>	<b>399</b>
53.1	Introduction .....	399
53.2	Implementation and Maintenance .....	399
53.3	Routine Descriptions .....	399
53.4	File List.....	399
53.5	Cross References .....	399
53.6	Exported Options .....	399
53.7	Exported Security Keys.....	400



53.8	Exported Protocols .....	400
53.9	Exported Parameters .....	400
53.10	Exported Mail Groups .....	400
53.11	Callable Routines .....	400
53.12	External Relations .....	400
53.13	Internal Relations .....	400
53.14	Archiving and Purging .....	400
53.15	Components .....	400
53.15.1	Properties .....	400
<b>54.0</b>	<b>Reminders (PCC).....</b>	<b>402</b>
54.1	Introduction .....	402
54.2	Implementation and Maintenance .....	402
54.3	Routine Descriptions .....	402
54.4	File List.....	403
54.5	Cross References .....	403
54.6	Exported Options .....	403
54.7	Exported Security Keys.....	403
54.8	Exported Protocols.....	403
54.9	Exported Parameters .....	403
54.10	Exported Mail Groups .....	403
54.11	Callable Routines .....	403
54.11.1	BEHORMCV DETAIL .....	403
54.11.2	BEHORMCV LIST .....	403
54.12	External Relations .....	404
54.13	Internal Relations .....	404
54.14	Archiving and Purging .....	404
54.15	Components.....	404
54.15.1	Properties .....	404
<b>55.0</b>	<b>View Reminders (CPRS).....</b>	<b>406</b>
55.1	Introduction .....	406
55.2	Implementation and Maintenance .....	406
55.3	Routine Descriptions .....	406
55.4	File List.....	406
55.5	Cross References .....	406
55.6	Exported Options .....	407
55.7	Exported Security Keys.....	410
55.8	Exported Protocols.....	410
55.9	Exported Parameters .....	410
55.10	Exported Mail Groups .....	410
55.11	Callable Routines .....	410
55.12	External Relations .....	411
55.13	Internal Relations .....	411
55.14	Archiving and Purging .....	411
55.15	Components.....	411

55.15.1	Properties .....	411
<b>56.0</b>	<b>Integrated Signature Tool.....</b>	<b>413</b>
56.1	Introduction .....	413
56.2	Implementation and Maintenance .....	413
56.3	Routine Descriptions .....	413
56.4	File List.....	413
56.5	Cross References .....	413
56.6	Exported Options .....	414
56.7	Exported Security Keys.....	414
56.8	Exported Protocols.....	414
56.9	Exported Parameters .....	414
56.10	Exported Mail Groups .....	414
56.11	Callable Routines .....	414
56.12	External Relations .....	414
56.13	Internal Relations .....	414
56.14	Archiving and Purging .....	414
56.15	Components.....	414
56.15.1	Properties .....	414
<b>57.0</b>	<b>Electronic Signature Service .....</b>	<b>416</b>
57.1	Introduction .....	416
57.2	Implementation and Maintenance .....	416
57.3	Routine Descriptions .....	416
57.4	File List.....	416
57.4.1	Cross References .....	416
57.4.2	Exported Options .....	416
57.4.3	Exported Security Keys .....	417
57.4.4	Exported Protocols .....	417
57.4.5	Exported Parameters.....	417
57.5	Exported Mail Groups .....	417
57.6	Callable Routines .....	417
57.7	External Relations .....	417
57.8	Internal Relations .....	417
57.9	Archiving and Purging .....	417
57.10	Components.....	417
57.10.1	Properties .....	417
57.10.2	Add .....	418
57.10.3	Clear .....	418
57.10.4	Exist .....	418
57.10.5	ExistForOrder .....	418
57.10.6	RegisterType .....	418
57.10.7	Remove .....	419
57.10.8	RenameGroup .....	419
57.10.9	ReplaceGroup.....	419
57.10.10	ReplaceID .....	419

57.10.11	ReplaceSignState .....	419
57.10.12	ReplaceText.....	420
57.10.13	Review .....	420
57.10.14	ReviewItems .....	420
<b>58.0</b>	<b>Allergies .....</b>	<b>421</b>
58.1	Introduction .....	421
58.2	Implementation and Maintenance .....	421
58.3	Routine Descriptions .....	421
58.4	File List.....	422
58.5	Cross References .....	422
58.6	Exported Options .....	422
58.7	Exported Security Keys.....	422
58.8	Exported Protocols.....	422
58.9	Exported Parameters .....	422
58.10	Exported Mail Groups .....	422
58.11	Callable Routines .....	422
58.11.1	RPC: BEHOARCV DETAIL .....	422
58.11.2	RPC: BEHOARCV LIST .....	423
58.12	External Relations .....	423
58.13	Internal Relations .....	423
58.14	Archiving and Purging.....	423
58.15	Components.....	423
58.15.1	Properties .....	423
<b>59.0</b>	<b>VueCentric to CPRS Context Adapter.....</b>	<b>425</b>
59.1	Introduction .....	425
59.2	Implementation and Maintenance .....	425
59.3	Routine Descriptions .....	425
59.4	File List.....	425
59.5	Cross References .....	425
59.6	Exported Options .....	425
59.7	Exported Security Keys.....	426
59.8	Exported Protocols.....	426
59.9	Exported Parameters .....	426
59.10	Exported Mail Groups .....	426
59.11	Callable Routines .....	426
59.12	External Relations .....	426
59.13	Internal Relations .....	426
59.14	Archiving and Purging.....	426
59.15	Components.....	426
59.15.1	NotifyOtherApps .....	426
<b>60.0</b>	<b>CPRS Options.....</b>	<b>427</b>
60.1	Introduction .....	427
60.2	Implementation and Maintenance .....	427

60.3	Routine Descriptions .....	428
60.4	File List.....	428
60.5	Cross References .....	428
60.6	Exported Options .....	428
60.7	Exported Security Keys.....	428
60.8	Exported Protocols.....	428
60.9	Exported Parameters .....	428
60.10	Exported Mail Groups .....	428
60.11	Callable Routines .....	428
60.12	External Relations .....	429
60.13	Internal Relations .....	429
60.14	Archiving and Purging .....	429
60.15	Components.....	429
60.15.1	Execute.....	429
<b>61.0</b>	<b>Medication Counseling.....</b>	<b>430</b>
61.1	Introduction .....	430
61.2	Implementation and Maintenance .....	430
61.3	Routine Descriptions .....	430
61.4	File List.....	431
61.5	Cross References .....	431
61.6	Exported Options .....	431
61.7	Exported Security Keys.....	431
61.8	Exported Protocols.....	431
61.9	Exported Parameters .....	432
61.10	Exported Mail Groups .....	432
61.11	Callable Routines .....	432
61.11.1	RPC: BEHORXED CANUSE .....	432
61.11.2	RPC: BEHORXED COMPLST.....	433
61.11.3	RPC: BEHORXED EDLST .....	433
61.11.4	RPC: BEHORXED POVLST.....	433
61.11.5	RPC: BEHORXED PRVNRPC .....	433
61.11.6	RPC: BEHORXED STORE.....	433
61.11.7	RPC: BEHORXED VSTLST .....	433
61.12	External Relations .....	434
61.13	Internal Relations .....	434
61.14	Archiving and Purging .....	434
61.15	Components.....	434
61.15.1	Properties .....	434
<b>62.0</b>	<b>Primary Care Information Header.....</b>	<b>436</b>
62.1	Introduction .....	436
62.2	Implementation and Maintenance .....	436
62.3	Routine Descriptions .....	436
62.4	File List.....	436
62.5	Cross References .....	436

62.6	Exported Options .....	437
62.7	Exported Security Keys.....	437
62.8	Exported Protocols.....	437
62.9	Exported Parameters .....	437
62.10	Exported Mail Groups .....	437
62.11	Callable Routines .....	437
62.12	External Relations .....	437
62.13	Internal Relations .....	437
62.14	Archiving and Purging .....	437
62.15	Components.....	437
62.15.1	Properties .....	438
<b>63.0</b>	<b>Spell Checking Service.....</b>	<b>439</b>
63.1	Introduction .....	439
63.2	Implementation and Maintenance .....	439
63.3	Routine Descriptions .....	439
63.4	File List.....	439
63.5	Cross References .....	439
63.6	Exported Options .....	440
63.7	Exported Security Keys.....	440
63.7.1	Exported Protocols .....	440
63.7.2	Exported Parameters.....	440
63.8	Exported Mail Groups .....	440
63.9	Callable Routines .....	440
63.9.1	\$\$SVCSCN^BEHOSPUT .....	440
63.10	External Relations .....	441
63.11	Internal Relations .....	441
63.12	Archiving and Purging .....	441
63.13	Components.....	441
63.13.1	Properties .....	441
63.13.2	AddDictionary .....	441
63.13.3	GrammarCheck .....	441
63.13.4	RemoveDictionary .....	442
63.13.5	Reset .....	442
63.13.6	ShowOptions .....	442
63.13.6.1	SpellCheck .....	442
<b>64.0</b>	<b>Appointments .....</b>	<b>443</b>
64.1	Introduction .....	443
64.2	Implementation and Maintenance .....	443
64.3	Routine Descriptions .....	444
64.4	File List.....	444
64.5	Cross References .....	444
64.6	Exported Options .....	444
64.7	Exported Security Keys.....	444
64.8	Exported Protocols.....	444

64.9	Exported Parameters .....	444
64.10	Exported Mail Groups .....	444
64.11	Callable Routines .....	444
64.11.1	RPC: BEHOENCV DETAIL .....	444
64.11.2	RPC: BEHOENCV LIST .....	445
64.12	External Relations .....	445
64.13	Internal Relations .....	445
64.14	Archiving and Purging .....	445
64.15	Components .....	445
64.15.1	Properties .....	445
<b>65.0</b>	<b>Message Broadcast .....</b>	<b>448</b>
65.1	Introduction .....	448
65.2	Implementation and Maintenance .....	448
65.3	Routine Descriptions .....	449
65.4	File List .....	449
65.5	Cross References .....	449
65.6	Exported Options .....	449
65.7	Exported Security Keys .....	449
65.8	Exported Protocols .....	449
65.9	Exported Parameters .....	449
65.10	Exported Mail Groups .....	449
65.11	Callable Routines .....	449
65.12	External Relations .....	449
65.13	Internal Relations .....	449
65.14	Archiving and Purging .....	449
65.15	Components .....	450
65.15.1	Properties .....	450
<b>66.0</b>	<b>Chat Service .....</b>	<b>452</b>
66.1	Introduction .....	452
66.2	Implementation and Maintenance .....	452
66.3	Routine Descriptions .....	452
66.4	File List .....	452
66.5	Cross References .....	453
66.6	Exported Options .....	453
66.7	Exported Security Keys .....	453
66.8	Exported Protocols .....	453
66.9	Exported Parameters .....	453
66.10	Exported Mail Groups .....	453
66.11	Callable Routines .....	453
66.12	External Relations .....	453
66.13	Internal Relations .....	453
66.14	Archiving and Purging .....	453
66.15	Components .....	453
66.15.1	IChatService .....	454

66.15.1.1	Properties .....	454
66.15.1.2	NewSession .....	454
66.15.2	IChatSession .....	454
66.15.2.1	Properties .....	454
<b>67.0</b>	<b>Internet Explorer .....</b>	<b>455</b>
67.1	Introduction .....	455
67.2	Implementation and Maintenance .....	455
67.3	Routine Descriptions .....	456
67.4	File List.....	456
67.5	Cross References .....	456
67.6	Exported Options .....	456
67.7	Exported Security Keys.....	456
67.8	Exported Protocols.....	457
67.9	Exported Parameters .....	457
67.10	Exported Mail Groups .....	457
67.11	Callable Routines .....	457
67.12	External Relations .....	457
67.13	Internal Relations .....	457
67.14	Archiving and Purging .....	457
67.15	Components.....	457
67.15.1	Properties .....	457
<b>68.0</b>	<b>Image.....</b>	<b>458</b>
68.1	Introduction .....	458
68.2	Implementation and Maintenance .....	458
68.3	Routine Descriptions .....	458
68.4	File List.....	458
68.5	Cross References .....	459
68.6	Exported Options .....	459
68.7	Exported Security Keys.....	459
68.8	Exported Protocols.....	459
68.9	Exported Parameters .....	459
68.10	Exported Mail Groups .....	459
68.11	Callable Routines .....	459
68.12	External Relations .....	459
68.13	Internal Relations .....	459
68.14	Archiving and Purging .....	459
68.15	Components.....	459
68.15.1	Properties .....	460
<b>69.0</b>	<b>Program Launcher .....</b>	<b>461</b>
69.1	Introduction .....	461
69.2	Implementation and Maintenance .....	461
69.3	Routine Descriptions .....	461
69.4	File List.....	461

69.5	Cross References .....	461
69.6	Exported Options .....	462
69.7	Exported Security Keys.....	462
69.8	Exported Protocols.....	462
69.9	Exported Parameters .....	462
69.10	Exported Mail Groups .....	462
69.11	Callable Routines .....	462
69.12	External Relations .....	462
69.13	Internal Relations .....	462
69.14	Archiving and Purging .....	462
69.15	Components.....	462
69.15.1	Properties .....	462
<b>70.0</b>	<b>Patient Photo .....</b>	<b>464</b>
70.1	Introduction .....	464
70.2	Implementation and Maintenance .....	464
70.3	Routine Descriptions .....	464
70.4	File List.....	464
70.5	Cross References .....	464
70.6	Exported Options .....	465
70.7	Exported Security Keys.....	465
70.8	Exported Protocols.....	465
70.9	Exported Parameters .....	465
70.10	Exported Mail Groups .....	465
70.11	Callable Routines .....	465
70.12	External Relations .....	465
70.13	Internal Relations .....	465
70.14	Archiving and Purging .....	465
70.15	Components.....	465
70.15.1	Properties .....	465
<b>71.0</b>	<b>Telnet.....</b>	<b>467</b>
71.1	Introduction .....	467
71.2	Implementation and Maintenance .....	467
71.3	Routine Descriptions .....	468
71.4	File List.....	468
71.5	Cross References .....	468
71.6	Exported Options .....	468
71.7	Exported Security Keys.....	468
71.8	Exported Protocols.....	468
71.9	Exported Parameters .....	468
71.10	Exported Mail Groups .....	468
71.11	Callable Routines .....	468
71.12	External Relations .....	469
71.13	Internal Relations .....	469
71.14	Archiving and Purging .....	469



71.15	Components.....	469
71.15.1	Properties .....	469
<b>72.0</b>	<b>Glossary.....</b>	<b>471</b>
<b>73.0</b>	<b>Appendix I – Developer Tutorial .....</b>	<b>472</b>
73.1	Introduction .....	472
73.2	Using Debug Mode .....	473
73.3	Using the Trace Log.....	473
73.4	About Component Support Services.....	475
73.5	About COM and ActiveX .....	476
73.6	Component Types.....	478
73.7	Component Registration .....	478
73.7.1	COM Registration .....	478
73.7.2	Framework Registration.....	478
73.7.3	Runtime Registration .....	479
73.8	Naming Conventions.....	479
73.9	Multiple vs. Single Instancing.....	480
73.10	Remote Procedure Calls .....	480
73.10.1	Create the M Routine.....	481
73.10.2	Create a Remote Procedure Definition.....	481
73.10.3	Register the Remote Procedure .....	482
73.10.4	Calling a Remote Procedure.....	482
73.10.5	Synchronous Calls.....	483
73.10.5.1	RPC Methods.....	483
73.10.5.2	Specifying the Remote Procedure.....	483
73.10.5.3	Specifying the Parameter List .....	484
73.10.5.4	Specifying Array Parameters.....	484
73.10.5.5	Handling Exceptions.....	485
73.10.6	Asynchronous Calls .....	485
73.10.6.1	Calling the Remote Procedure Asynchronously.....	485
73.10.6.2	Implementing the Callback Interface .....	486
73.10.6.3	Aborting a Pending Call.....	486
73.11	Context Management.....	487
73.11.1	Callbacks .....	487
73.11.2	Requesting a Context Change.....	488
73.12	Events .....	488
73.12.1	Defining an Event .....	488
73.12.2	Firing Events: Local vs. Remote .....	489
73.12.3	Receiving Events: Callbacks .....	489
73.13	Creating Visual Components with Delphi.....	490
73.13.1	Creating the Active Form Project.....	490
73.13.2	Designing the Form .....	493
73.13.3	Accessing the Session Object .....	493
73.13.4	Accessing the Patient Context Object .....	495
73.13.5	Calling a Remote Procedure in Synchronous Mode.....	496
73.13.6	Testing the Component .....	497

73.13.7	Subscribing to Patient Context Changes .....	498
73.13.8	Calling a Remote Procedure in Asynchronous Mode .....	503
73.13.9	Firing an Event.....	505
73.13.10	Subscribing and Responding to an Event.....	506
73.13.11	Summary .....	507
73.14	Creating Visual Components with Visual Basic .....	507
73.14.1	Creating the ActiveX Control Project .....	507
73.14.2	Designing the Form .....	509
73.14.3	Accessing the Session Object .....	510
73.14.4	Calling a Remote Procedure in Synchronous Mode.....	513
73.14.5	Testing the Component .....	513
73.14.6	Subscribing to Patient Context Changes .....	515
73.14.7	Calling a Remote Procedure in Asynchronous Mode .....	517
73.14.8	Firing an Event.....	519
73.14.9	Subscribing and Responding to an Event.....	520
73.14.10	Summary .....	521
73.15	Creating Visual Components with C# .....	521
73.15.1	Creating the Windows Control Project.....	521
73.15.2	Accessing the Session Object .....	526
73.15.3	Accessing the Patient Context Object .....	527
73.16	Calling a Remote Procedure in Synchronous Mode .....	529
73.16.1	Testing the Component .....	529
73.16.2	Subscribing to Patient Context Changes .....	532
73.16.3	Calling a Remote Procedure in Asynchronous Mode .....	534
73.16.4	Firing an Event.....	536
73.16.5	Subscribing and Responding to an Event.....	537
73.16.6	Summary .....	538
73.17	Creating Services.....	538
73.18	Creating Services with Delphi .....	538
73.18.1	Creating the Project.....	539
73.18.2	Creating the Service Object.....	539
73.18.3	Accessing the Session Object .....	541
73.18.4	Modifying the Interface .....	542
73.18.5	Providing the Implementation .....	543
73.18.6	Registering the Service.....	544
73.18.7	Accessing the Service .....	544
73.18.8	Summary .....	546
73.19	Creating Services with Visual Basic.....	546
73.19.1	Creating the Project.....	546
73.19.2	Accessing the Session Object .....	547
73.19.3	Modifying the Interface .....	549
73.19.4	Registering the Service.....	549
73.19.5	Accessing the Service .....	549
73.19.6	Summary .....	551
73.20	Deploying Components.....	551
73.21	Version Control .....	551

---

73.21.1	Version Numbers .....	552
73.21.2	Which Version?.....	552
73.21.3	Registering Version Information .....	552
73.21.4	What is Side-by-Side Versioning? .....	553
73.21.5	Imbedding Version Information.....	554
73.22	Handling Dependencies .....	555
73.23	Generating KIDS Builds .....	556
73.24	Pitfalls and Special Techniques .....	557
73.24.1	Component Initialization .....	557
73.24.2	Component Destruction .....	557
73.24.3	Other Containers .....	558
73.24.4	Focus Issues.....	558
73.24.5	Deferring Data Fetches.....	558
73.24.6	Intercomponent Communication.....	558
73.24.7	Creating Trace Log Entries.....	559
73.24.8	Embedding Licensed Controls.....	561
73.24.9	Forced Context Changes.....	561
73.24.10	Integrating Help Content.....	562
<b>74.0</b>	<b>Contact Information .....</b>	<b>564</b>

## 1.0 Introduction

The RPMS-EHR is comprised of multiple functional components built upon an open architecture framework known as VueCentric®. The unique construction of the application from over 70 discrete components dictates a slightly different structure for technical documentation. This manual is organized into multiple sections that correspond to functional groupings of the various components that make up the RPMS-EHR. These groupings are, in order:

- VueCentric® Framework
- RemoteProcedure Call Broker
- Context Objects and Related Components
- PCC-based Components
- Problem List
- TIU-based Components
- Order Entry
- General Reporting
- Notifications
- Reminders
- Electronic Signature
- Other Components

## 2.0 VueCentric® Framework

### 2.1 Introduction

VueCentric® is a multi-tiered, open-architecture, component-based framework that supports a wide range of clinical functions using standardized, plug-in objects. With the appropriate server-side RPMS components, the fully implemented version has objects that support patient lookup, clinical encounter documentation, on-line ordering, results retrieval, decision support, problem list management, consult tracking and adverse reaction tracking, to list a few. Using a Visual Interface Manager (VIM), power users can select from a palette of objects and construct a fully functional application from discrete components. An application can be designed to meet the needs of an individual, user class, site, or a specialized requirement. Once assembled, a configuration has the appearance of a cohesive whole, belying its component-based origins. Each component communicates with a middle tier Component Support Services (CSS) that coordinates the activities of the objects so that the result is a seamless application. The CSS provides event and context management and remote data access services to components within its application space. The CSS also communicates with any CCOW-compliant context manager to allow the application and its components to share context state with other CCOW-aware applications running on the same desktop. The Component Management Service (CMS) performs just-in-time updating of requested components, enforces access security, and controls many aspects of component run-time behavior. The Communication Service Layer (CSL) performs user authentication and mediates data exchange between the host system and the CSS.

### 2.2 Architecture

Before one can support the interoperability of plug-in components, one must explicitly define the rules for such interoperability. VueCentric® defines a multi-tiered architecture that insures the interoperability of components developed in accordance with the specification. The key constituents of the VueCentric® Framework are described below.

The *Visual Interface Manager* (VIM) represents the top tier of the VueCentric® architecture. It acts as the glue that holds the individual components together. It empowers the user to define the visual relationships among discrete components, provides the ability to compose complex interfaces from individual visual elements, supports the streaming of compositional entities to and from a central store, controls user-level access to components, and can interrogate components for the resources they require and automatically connect them to those resources.

The *Component Support Services* (CSS) comprise the middle tier and provide shared resources that all components can access and coordinate the activities of components. The CSS supports the concept of plug-in services that augment the functionality of the middle tier in a fully extensible manner. Available services include context objects that

reflect the current state of the application, such as the currently selected patient, the user who is logged in, or the clinical encounter that is being referenced. Other plug-in services include unified electronic signature, report generation, remote data views and clinical reminders. The CSS also provides support for performing remote procedure calls to allow objects to interact with the host system. The CSS is also a manager and producer of events that can notify components who choose to subscribe that, for example, the patient selection has changed. Finally, the CSS can also participate in context changes that originate outside the application. Because the CSS automatically detects the presence of a CCOW-compliant context manager and registers as a participant, the VueCentric® application can synchronize its context with other CCOW-compliant applications residing on the same desktop.

Critical to the interaction between the middle tier elements and the bottom tier host system is the *Communication Service Layer* (CSL). Its roles are to perform user authentication and to mediate both synchronous and asynchronous data exchange between the two tiers. The CSL is completely encapsulated by the CSS in order to facilitate the abstraction of the data access layer. This makes it possible to incorporate other data access components without adversely affecting existing consumers of the service.

One of the core features of the VueCentric® framework is the just-in-time deployment of components. The *Component Management Service* (CMS) performs this function. This service enforces version control and access security, deploys updates from a central repository, and controls other aspects of a component's behavior at runtime.

In addition to the architectural elements described above, the VueCentric® framework also includes external data stores in the form of an *Object Registry*, a *Template Registry*, and an Object Repository.

The *Object Registry* provides information about available objects and their default characteristics. The CMS provides a read-only, object-oriented view of these data.

The *Template Registry* provides a globally accessible location for the storage of state-information in a context-free format. It is used to store user interface configuration information.

The *Object Repository* is a store of components that are accessible to the VueCentric® application. The component repository allows an application to automatically update locally installed components from a reliable source. The component repository can be implemented by a Web server, an ftp server, or any globally accessible directory, or any combination of all three. The VueCentric® framework models the just-in-time component updating mechanism after that employed by web browsers. Under that paradigm, an HTML document requests a component by a unique identifier and version. If the requested component is already available locally, that version is used. If it is not, the HTML document includes a URL reference that defines a source from which the component can be downloaded and installed. The CMS uses a very similar

approach that permits it to automatically propagate updates to existing components as well as deploy new components to individual workstations as they are needed.

The diagram below summarizes the architecture of the VueCentric® Framework.

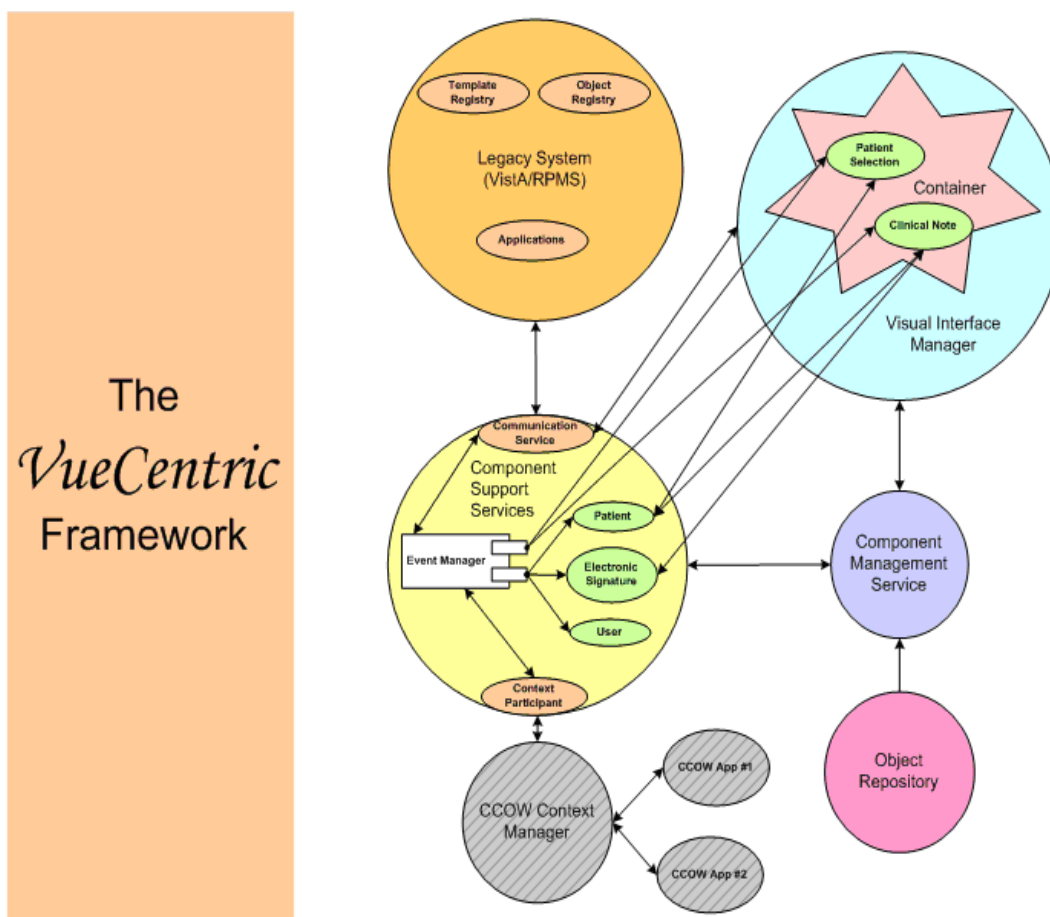


Figure 2-1: Architecture of the Framework

## 2.3 Implementation and Maintenance

The following sections describe tools available for the implementation and maintenance of the VueCentric® Framework.

### 2.3.1 VueCentric System Management (vcManager) Utility

#### 2.3.1.1 Introduction

The VueCentric System Management Utility permits the package administrator to control several aspects of the VueCentric® framework including:

- Object Registration
- Template Management
- Site Parameters
- System Shutdown

- Remote Troubleshooting

The utility can be found in the “utl” folder of the RPMS-EHR distribution as the file vcManager.exe.

### 2.3.1.2 Logon Screen

Before using the VueCentric® System Management Utility, the user must logon to the host system. The logon dialog can be preceded by a server selection dialog, depending on how the connection service is configured. If this dialog is presented, the user must first select the server with which to establish a connection.

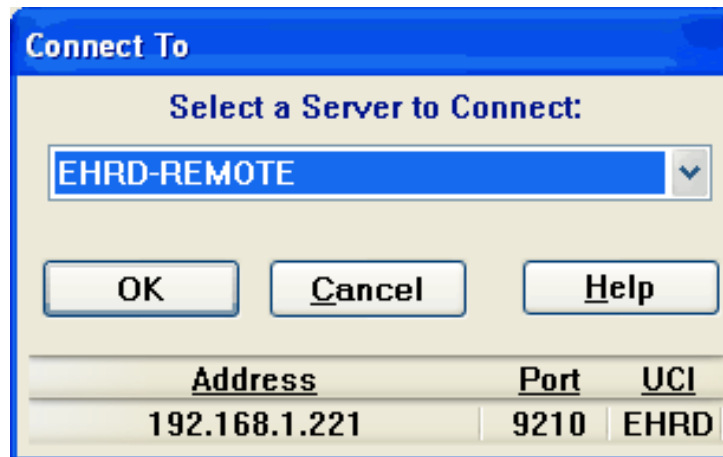


Figure 2-2: Server Selection Dialog

In either case, the user will then be presented with the logon dialog.

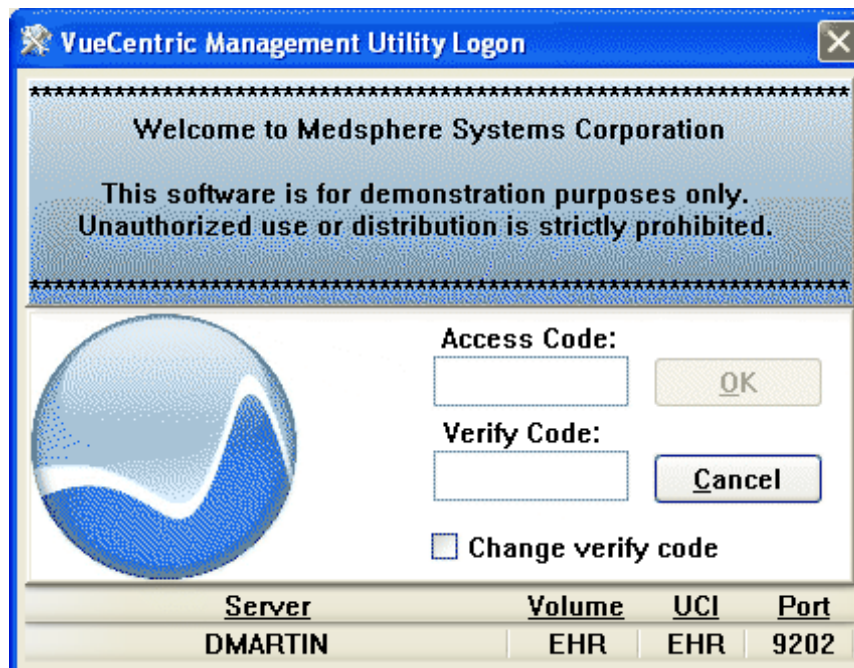


Figure 2-3: Login Dialog



Enter your access and verify code appropriate for the server to which you are connected.

**Note: You must have the CIAV SITE MANAGER security key to successfully run the manager utility.**

### 2.3.1.3 Application Menu

The following menu options are available regardless of the tab selected:

Menu	Menu Item	Effect
File	Default Source Path...	Sets the corresponding parameter (VueCentric Object Source) on the currently connected host. This parameter determines the default location of the VueCentric Object Repository. Unless otherwise specified in a component's registration information, this is the location from which a component is retrieved for updating.
	Default Target Path...	This sets the directory location on the workstation where retrieved components are to be placed. This setting only affects the current session. If the Retrieve button is clicked and a default target path has not already been specified, the user is prompted for this information.
	Logout...	Logs off the current host, but does not terminate the application.
	Exit	Logs off and terminates the application.
Tools	<i>User Configurable</i>	<p>User configurable by adding to the [tools] section of the vcManager.ini file.</p> <p>Use NotePad to create a text file called vcManager.ini in the same folder as the vcManager application and format according to the example below.</p> <p>The following example creates a single menu item labeled Registry Editor under the Tools menu, which invokes the Windows registry editor when clicked.</p> <pre>[tools] Registry Editor=regedit.exe</pre>
Help	Contents...	Displays the help file table of contents.
	About...	Displays the About tab.

### 2.3.1.4 Object Registry Tab

The *Object Registry* tab permits the registration of objects that can then be accessed by the VueCentric® framework. Objects can be visual components that are available in the *Add Object* dialog of the VueCentric® *Visual Interface Manager* or services that visual components can access. Object registration permits the package manager to control many aspects of an object's behavior including security access, visual appearance, object-specific properties, versioning, and installation. These settings are stored in the *VueCentric Object Registry* file on the target host system and are, therefore, specific to that system.

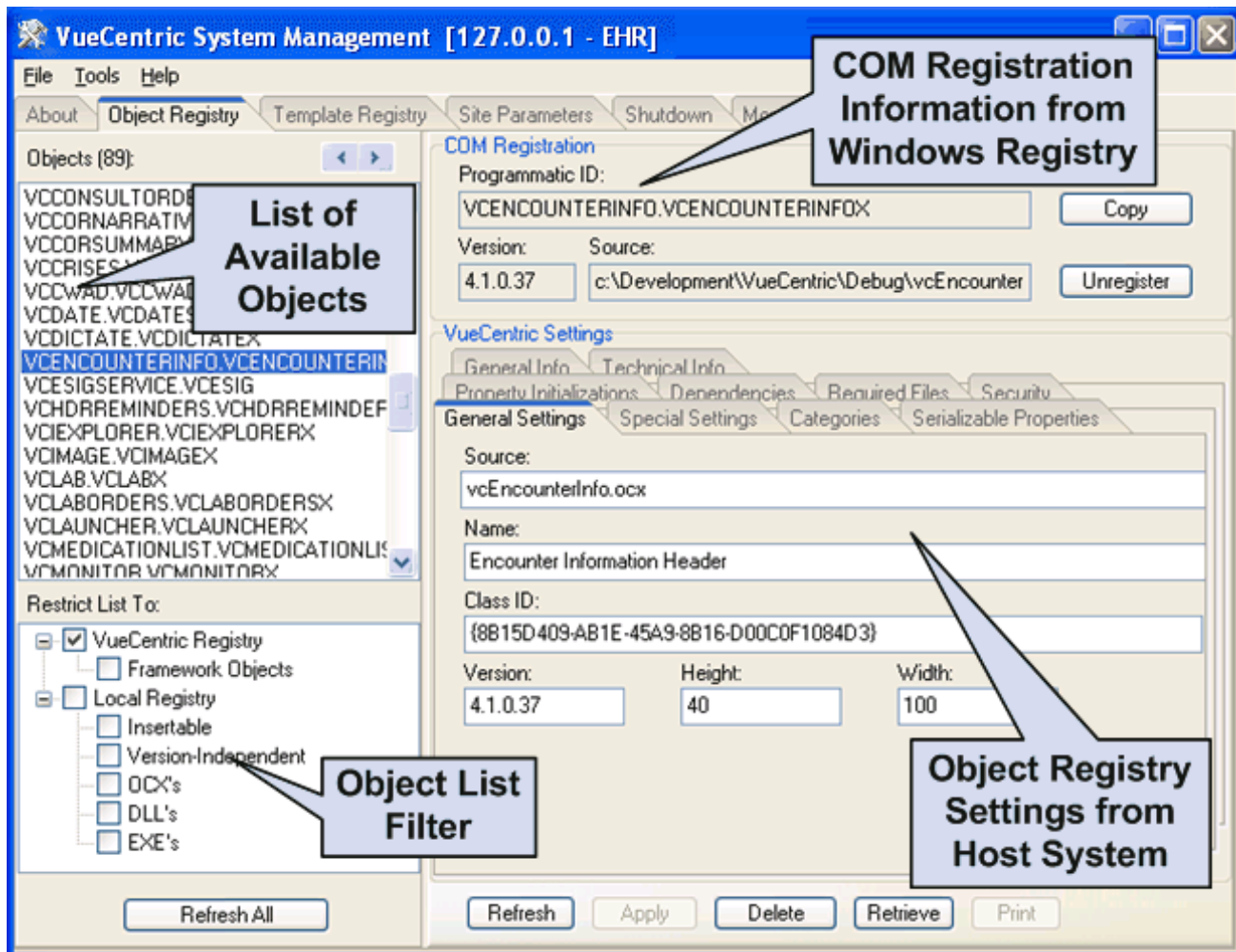


Figure 2-4: View Registry Tab

### 2.3.1.4.1 File Menu

The following menu options are available under the File menu only when the Object Registry tab is active.

Menu Item	Effect
New	Inserts a new entry in the object list. Prompts the user for a programmatic identifier. This is useful for manually registering an object that has no associated export file.
Import	Enables the importation of object registration information from an export file (.vor extension).
Export	Creates an export file (.vor extension) that contains registration information for the currently selected object.

### 2.3.1.4.2 Object List Pane

The **Object List** pane shows the programmatic identifiers of available objects. Selecting an entry in this list displays information about that entry in the *COM Registration* and *VueCentric Settings* panes on the right.

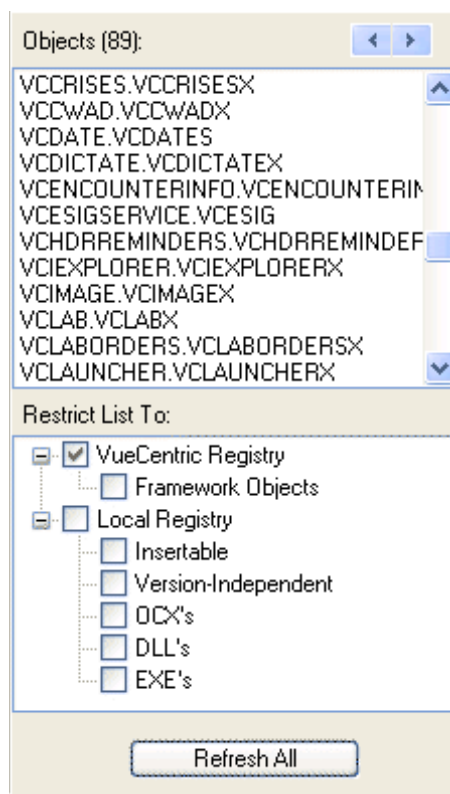


Figure 2-5: Object List

Clicking the **Refresh All** button causes a refresh of the object list using the current filter settings. Changing a filter setting also causes a refresh of the object list.

The **arrow buttons** at the top right move to the next/previous entry where the local version and the master version differ. This is useful to locate entries that require updating.

The **Restrict List To** pane controls the content of the object list. The filter has two top level entries. The **VueCentric Registry** entry is checked by default and restricts the list to those objects that are registered in the VueCentric Object Registry file on the host system. The **Local Registry** entry controls the inclusion of items from the COM registry on the local machine. Both entries have subordinate entries that can be used to further restrict which objects are included in or excluded from this list (below).

Filter Item	Effect
VueCentric Registry	Includes entries from the VueCentric Object Registry file located on the host system.

Filter Item	Effect
Framework Objects	Includes internal framework objects. These objects are essential to the proper operation of the framework. Their settings should never be modified.
Local Registry	Includes entries from the local machine's COM registry. This can be further restricted using one or more of the filter items below:
Insertable	Restricts list to local registry entries marked with the insertable attribute.
Version Independent	Includes version-independent programmatic identifiers. COM objects can register with both version-dependent (which typically includes a version number in the identifier) and version-independent (which do not include a version number) identifiers.
OCXs	Restricts list to objects whose executable has an OCX extension.
DLLs	Restricts list to objects whose executable has a DLL extension.
EXEs	Restricts list to objects whose executable has a EXE extension.

### 2.3.1.4.3 COM Registration Pane

The **COM Registration** pane displays information about the selected object from the local machine's Windows registry. If the object has not been previously registered on the local machine, these fields might be blank. These fields are display only and cannot be modified.

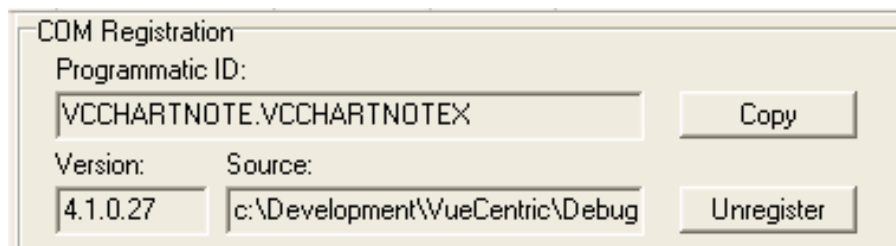


Figure 2-6: COM Registration

The **Copy** button copies the contents of the version and source fields as well as the object's class identifier (GUID) to the corresponding fields in the VueCentric Settings pane. This is a useful shortcut when updating version information or creating new entries.

The **Unregister** button executes the COM object's `DLLUnregisterServer` method to remove its registration information from the Windows registry.

**Note:** the version number is derived from the version number resource imbedded in the object's executable image rather than by reading it from the COM registry. Because the COM registry version only reflects the version of the type library of the last copy of the object that was registered, it is an unreliable indicator of the actual version of the object that is instantiated at runtime.

#### 2.3.1.4.4 VueCentric Settings Pane

The VueCentric Settings pane presents information about the currently selected object from the VueCentric Object Registry file on the host system. These settings can be modified by the user.

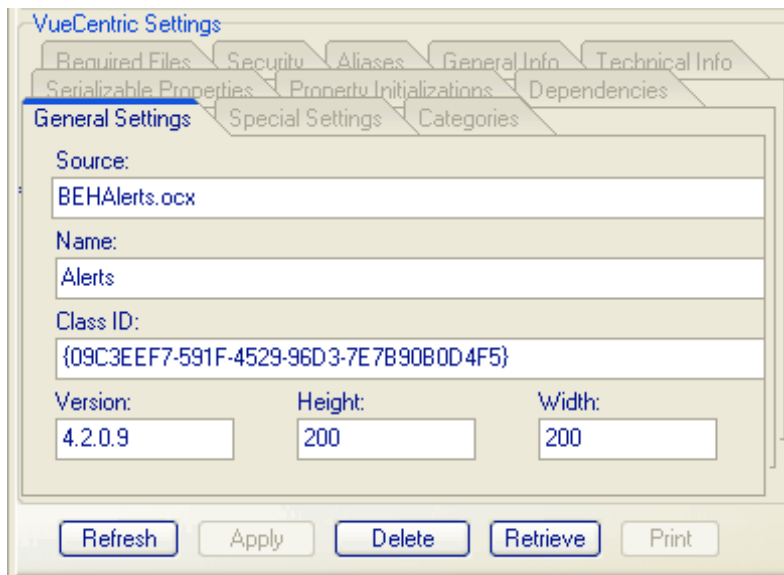


Figure 2-7: VueCentric Settings Pane

The pane consists of 10 tabs that organize fields into logical groupings. Each of these tabs is described in detail in the sections that follow.

The **Refresh** button synchronizes the displayed settings with those stored on the host system. Any pending changes are lost.

The **Apply** button is enabled when pending changes are present. Clicking this button commits changes to the database.

The **Delete** button removes the entry for the selected object from the VueCentric Object Registry file.

The **Retrieve** button retrieves the selected object from the source and stores it into the local application directory. Any dependent components are also retrieved. COM components retrieved in this manner are automatically registered.

The **Print** button is enabled when either information tab is selected, allowing the user to print the information contained therein.

#### 2.3.1.4.5 General Settings Tab

The General Settings tab displays the following fields:

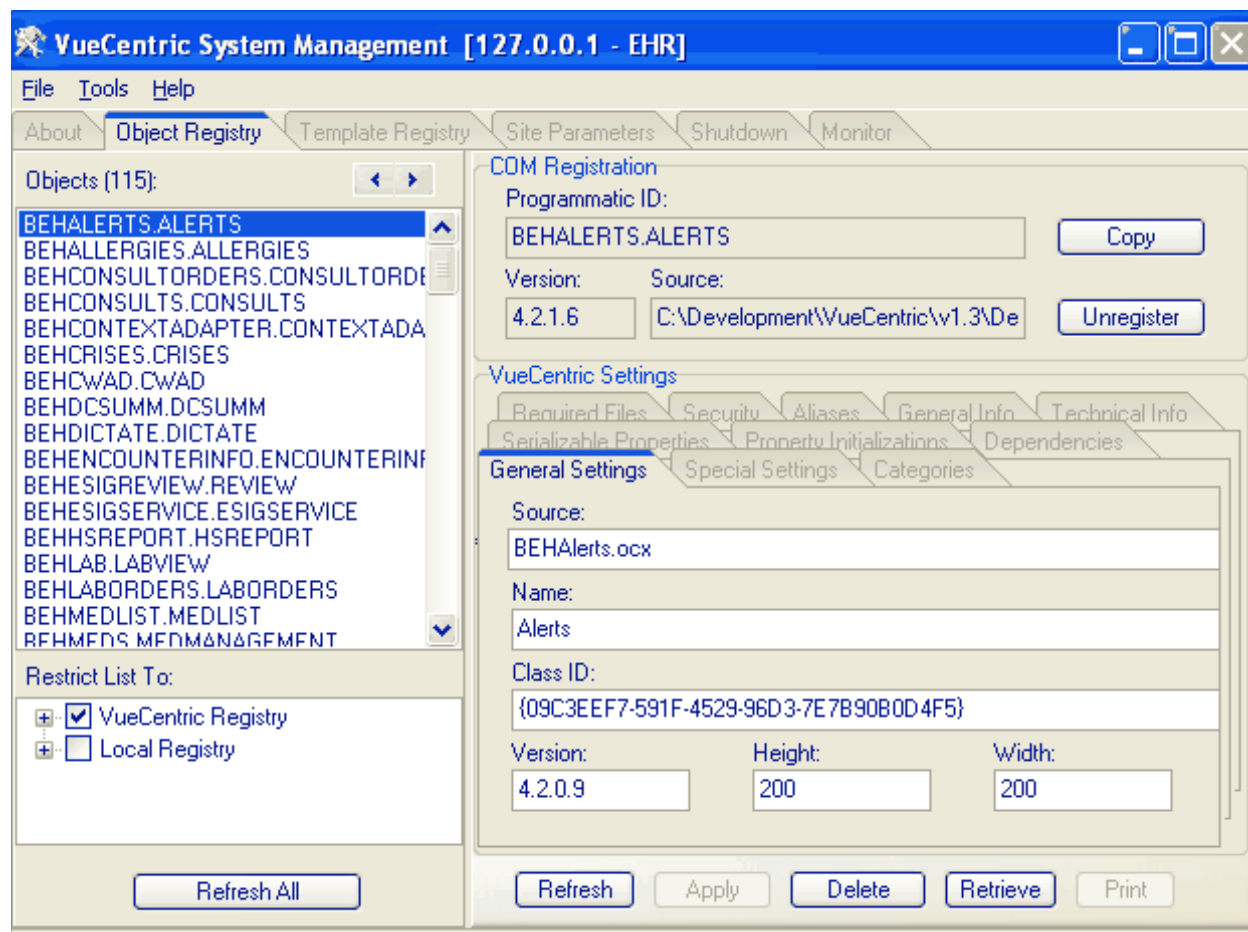


Figure 2-8: General Settings Tab

**Source:** This is the name of the file containing the object. If path information is not included, the default source path is assumed. This information is used to locate the master copy of an object when an update is required.

**Name:** This is the friendly name of the object that appears in the Add Object dialog of the Visual Interface Manager. It should be kept short, yet be sufficiently descriptive of the object's function.

**Class ID:** If the object is a COM object, this is the class identifier of the principal CoClass. This information can be automatically copied from the COM Registry by clicking the Copy button in the COM Registration pane. This entry is not required. However, if the class identifier is not specified, requests of this object's services using its class identifier will fail if the object has not be previously installed and registered. If the class identifier is specified, the Component Support Service can use this information to locate the appropriate entry and retrieve and install the necessary components.

**Version:** This is the version of the master copy of the object. This information is derived from the file's imbedded version resource if one exists, or is derived from the

file's modification timestamp if it does not. It is important that this setting accurately reflect the actual version of the master copy. If it does not, the Component Support Service will not be able to properly determine when an object needs to be updated.

**Height & Width:** These fields apply to visual components only and represent the default dimensions (in pixels) for the component when it is initially created in design mode.

#### 2.3.1.4.6 Special Settings Tab

The Special Settings tab permits the specification of attributes that affect how the Visual Interface Manager handles an object.

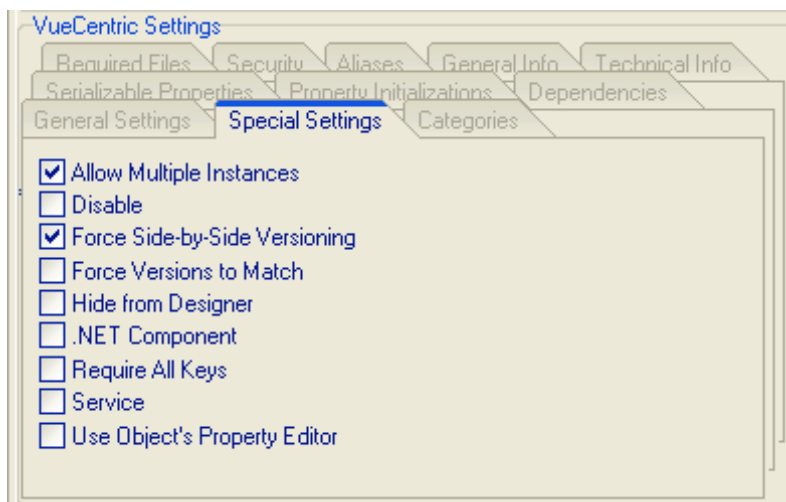


Figure 2-9: Special Settings Tab

The table below describes the function of each of these attributes.

Attribute	Effect
Allow Multiple Instances	If checked, multiple instances of the object can exist within the current configuration template. By default, only one instance of an object can be placed within a template. Subsequent attempts to place the same object in the template will be rejected.
Disable	If checked, the object is disabled. Any attempts to instantiate the object will be rejected. In the Add Object dialog within the Visual Interface Manager, the object's icon will be grayed and the object will not be selectable. If a configuration template is loaded that contains a disable object, the template will still be usable, but a placeholder will appear in place of the object indicating that it is disabled. This feature is useful if an object is problematic and needs to be temporarily taken out of service.

Attribute	Effect
Force Side-by-Side Versioning	<p>Side-by-side versioning refers to the ability to have multiple versions of the same component residing on a machine at the same time.</p> <p>The default behavior is to share a single copy of an object across all applications. This is often not desirable, especially in the situation where an object version is tied to a specific software version on the host system.</p> <p>When side-by-side versioning is enabled, no path information is included in the COM registration for the object. When a request is made to instantiate an object registered in this manner, Windows searches for the executable image first in the application directory, then in the operating system directories. By partitioning the application and its components into separate directories on the basis of the target host system, one can insure that the client versions are appropriate for the target host.</p> <p>While some objects implement side-by-side versioning internally, checking this attribute ensures this form of version control is imposed regardless of whether the object supports it natively.</p> <p>Unless an object is clearly independent of the target host system, it is recommended that it be registered with side-by-side versioning enabled.</p>
Hide from Designer	<p>Checking this attribute prevents the object from appearing in the Add Object and Required Services dialogs of the <i>Visual Interface Manager</i>. This setting is appropriate for any supporting files that are not ActiveX components.</p>
.NET Component	<p>If checked, this object represents a .NET component. This setting is necessary to ensure successful creation of a .NET component with the <i>Visual Interface Manager</i>.</p>
Require All Keys	<p>If checked, this attribute forces the requirement that the user must have all security keys associated with object in order to use it. If unchecked, having any one of the associated keys is sufficient. See the description of the Security Tab for more information on using security keys to control object access.</p>
Service	<p>If checked, the object represents a service that can be shared across multiple components. A service is essentially a plug-in that augments the capabilities of the framework in some way (e.g., all context objects are implemented as services). Services are accessed by other objects programmatically through the <i>Component Support Services</i> layer. Marking an object as a service ensures that it does not appear in the <i>Add Object</i> dialog, and that it is automatically loaded and started when needed.</p>
Use Object's Property Editor	<p>The Visual Interface Manager provides a generic property editor for all visual components. This is generally suitable for most needs. Since ActiveX objects can and often do implement their own property editors, checking this attribute forces the Visual Interface Manager to use the object's internal property editor instead of the generic one.</p> <p>For more information about the generic property editor, see the discussion of the Serializable Properties tab.</p>



### 2.3.1.4.7 Categories Tab

Categories determine the placement of an object in the tree view control within the Add Object dialog of the Visual Interface Manager. They can also help organize nonvisual components into functional categories as well.

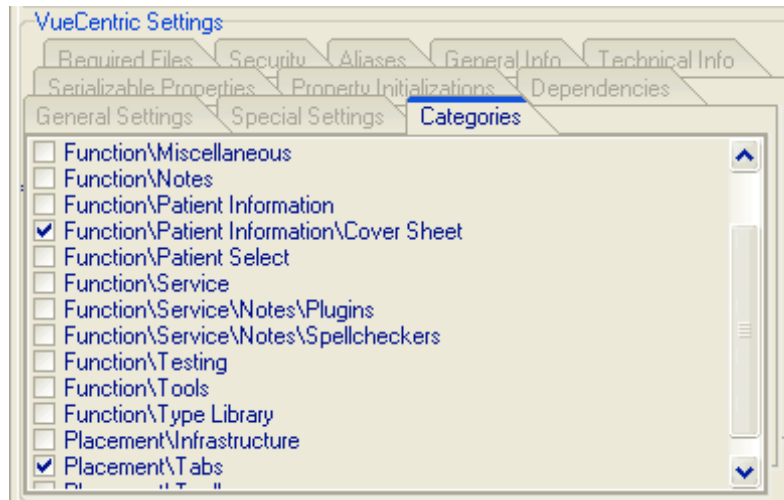


Figure 2-10: Categories Tab

The category list comes from the *VueCentric Object Category* file of the host system. Additional entries can be added using FileMan. Note the use of the backslash character to separate labels for each of the nodes in the tree view. While these can be nested as deeply as desired, generally a depth greater than two to three nodes is not advisable.

Note that an object can belong to multiple categories and doing so will cause the object to appear in more than one place in the tree view.

**Hint:** When this tab initially appears, only previously checked items are visible. To see all possible entries, right-click on the tab and select Show All Categories from the popup menu.

### 2.3.1.4.8 Serializable Properties Tab

The purpose of serializable properties is twofold. First, entries in this list control which properties appear in the generic property editor of the Visual Interface Manager and in what order. Second, the current value of every property in this list will be saved

when a configuration template containing the associated object is saved and restored when that template is loaded.

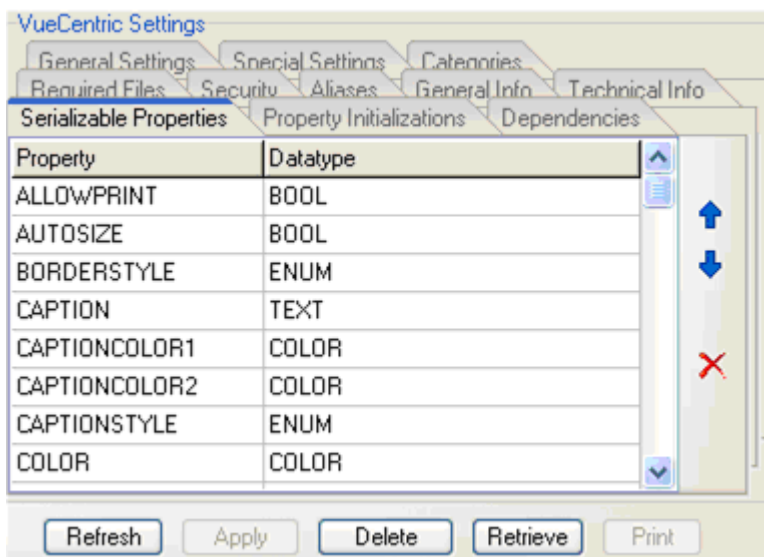


Figure 2-11: Serializable Properties Tab

The property name must match the name of an existing property within the associated object. Case is not significant. The datatype is used to determine which property page the generic property editor presents to the user for editing this property. The available datatypes are listed in the table below.

To delete an entry, click on the property name and then click the delete button. To add an entry, move beyond the last row using either the arrow keys or pressing enter when on the last row. To resequence entries, use the arrow keys to move the selected row up or down.

The following datatypes are recognized.

Datatype	Description	Property Page Appearance
BOOL	The property is true or false.	<input type="checkbox"/> False
COLOR	The property is color.	<input type="text" value="Yellow"/>
ENUM	The property is one of several fixed values.	No caption
FILE	The property is filename.	C:\Splash.jpg ...
FLAG	Similar to an enumeration, but selections are not mutually exclusive and might overlap (i.e., selecting one can affect the selection of others).	<input checked="" type="checkbox"/> Top; Left

Datatype	Description	Property Page Appearance
FONT	The property is font.	<input type="text" value="MS Sans Serif"/>
HIDDEN	The property is not	N/A
ICON IMAGE	The property is an icon or graphic image file.	<input type="text" value="C:\Splash.jpg"/>
INT	The property is an integer value.	<input type="text" value="600"/>
TEXT	The property is simple text.	<input type="text" value="VueCentric"/>

### 2.3.1.4.9 Property Initializations Tab

Property initializations permit setting property values to something other than their default values. These initializations are applied to every instance of the object and before any serialized property values are set. Therefore, property values set by a property editor and saved (see Serializable Properties Tab) as part of a configuration template override any settings established here.

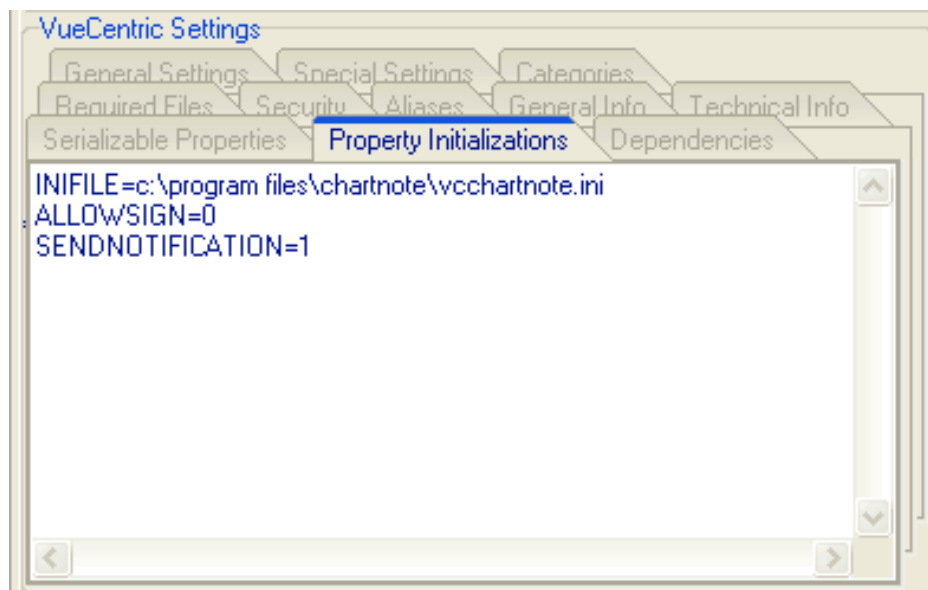


Figure 2-12: Property Initializations Tab

The format for entries in this list is

**<property name>=<value>**  
**or**  
**<property name>=@<parameter name>**

The property name must match that of an existing property. Case is not significant. Parameter names correspond to parameters in the Kernel Parameter Definition file. This format allows mapping a property to a parameter.

### 2.3.1.4.10 Dependencies Tab

Some objects depend on the presence of other components. This tab enables these dependencies to be explicitly declared. Whenever a request is made for an object, the Component Management Service uses this information to ensure that all required components are present and up-to-date. This process is recursive, so if any required component itself has required components, these too are updated if necessary. Cyclic dependencies are appropriately handled so infinite update loops are not possible.

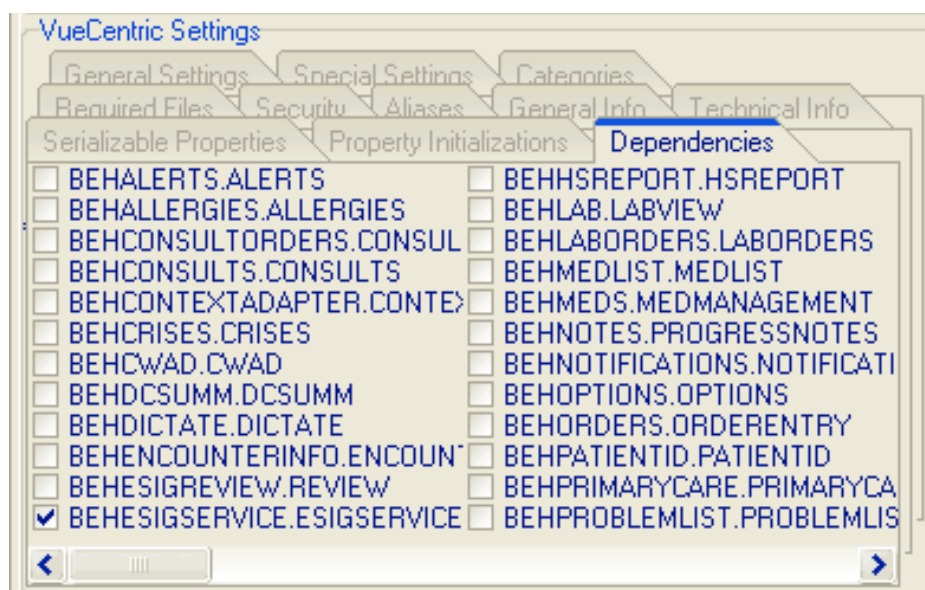


Figure 2-13: Dependencies Tab

If a required component is marked as a service (see discussion of special settings), the service will be started automatically if it is not already running. This eliminates the need for an object to explicitly start a service that it requires.

This list includes all entries from the *VueCentric Object Registry* file.

**Hint:** When this tab initially appears, only previously checked items are visible. To see all possible entries, right-click on the tab and select Show All Objects from the popup menu.

### 2.3.1.4.11 Required Files Tab

This field allows one to list any additional files that can be required for the operation of the associated object. If a filename is followed by a semicolon and version number, that file will be updated if that version is more recent than the installed version. In the absence of a version number, a file is updated only when the associated object is updated.

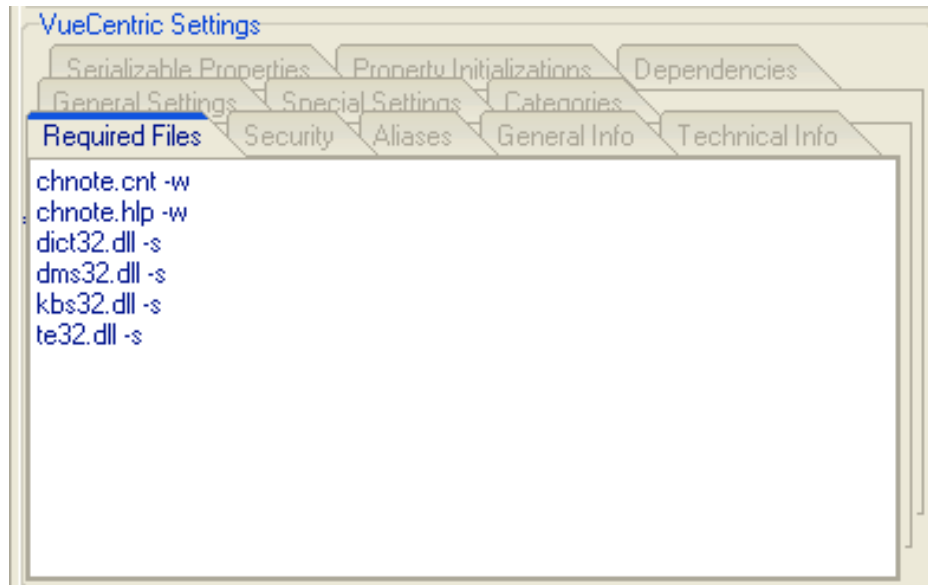


Figure 2-14: Required Files Tab

The format for each entry is:

**<filename> -<flag>**  
**or**  
**<filename>;<version #> -<flag>**

The inclusion of one or more flags after the filename is optional. The available flags are described in the table below.

Flag	Effect
-d	Delete image afterwards (use with -e)
-e	Execute image after copying
-n	Don't copy if file already exists
-p	Delete image only (do not copy)
-r	Register image after copying
-s	Copy to system directory
-t	Download as text (applies to FTP downloads only)
-u	Unregister image
-w	Copy to windows directory
-x	Copy to temp directory

**Hint:** To automatically update version numbers for required files, right-click anywhere on the file list and select **Update Version Numbers** from the popup menu

### 2.3.1.4.12 Security Tab

This tab permits associating one or more security keys with the selected object. This controls access to the object. If the Require All Keys setting on the Special Settings tab is checked, the user must possess all checked keys to access the object. Otherwise, the possession of at least one of the checked keys is sufficient.

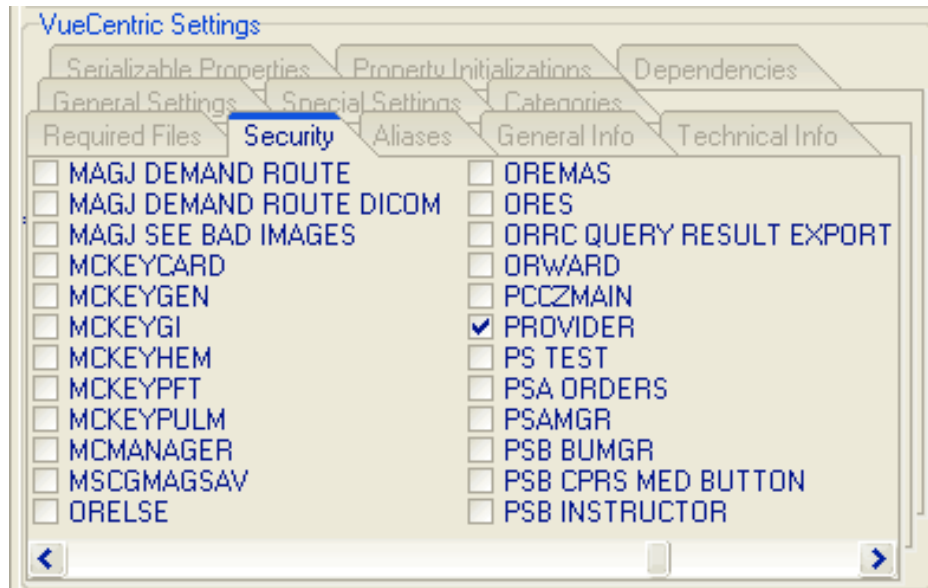


Figure 2-15: Security Tab

**Hint:** When this tab initially appears, only previously checked items are visible. To see all possible entries, right-click on the tab and select Show All Keys from the popup menu.

### 2.3.1.4.13 Aliases Tab

This field lists all programmatic identifiers by which the associated object has been known in the past. This information is used to redirect old object references in templates to the current identifier.

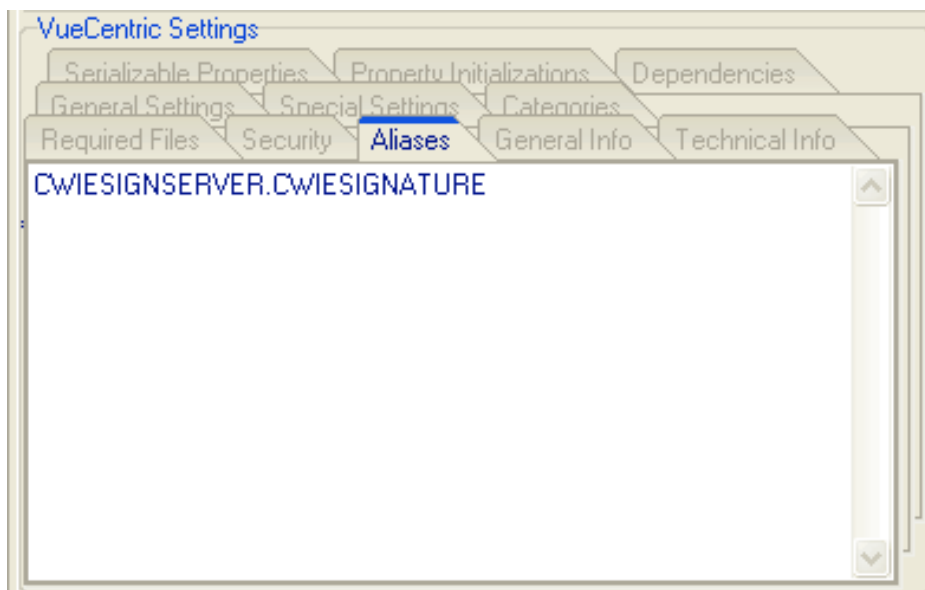


Figure 2-16: Aliases Tab

#### 2.3.1.4.14 General Info Tab

Displays general information about the object that describes its purpose and function.

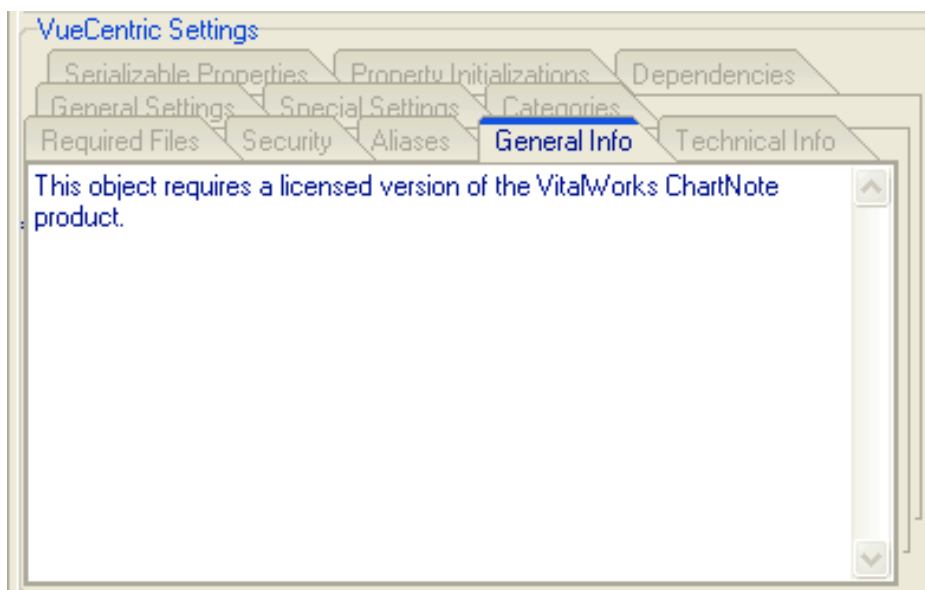


Figure 2-17: General Info Tab

#### 2.3.1.5 Template Registry Tab

The Template Registry tab supports management of configuration templates, which represent snapshots of various visual layouts. Templates are stored on the host system in the VueCentric Template Registry file as an XML representation of the visual layout. There are three types of templates:

- **Application Templates** begin with a % and represent snapshots of an entire application. These templates include application level elements such as font settings and menu items.
- **User Templates** begin with a \$ followed by the corresponding user's internal identifier. User templates are like application templates except they are private to a specific user and are created when a user with the appropriate privilege saves a configuration as their personal default.
- **Object Templates** lack a prefix character and do not have application level elements stored within them. They are used to assemble precomposed collections of individual objects and appear in the Add Object dialog of the Visual Interface Manager.

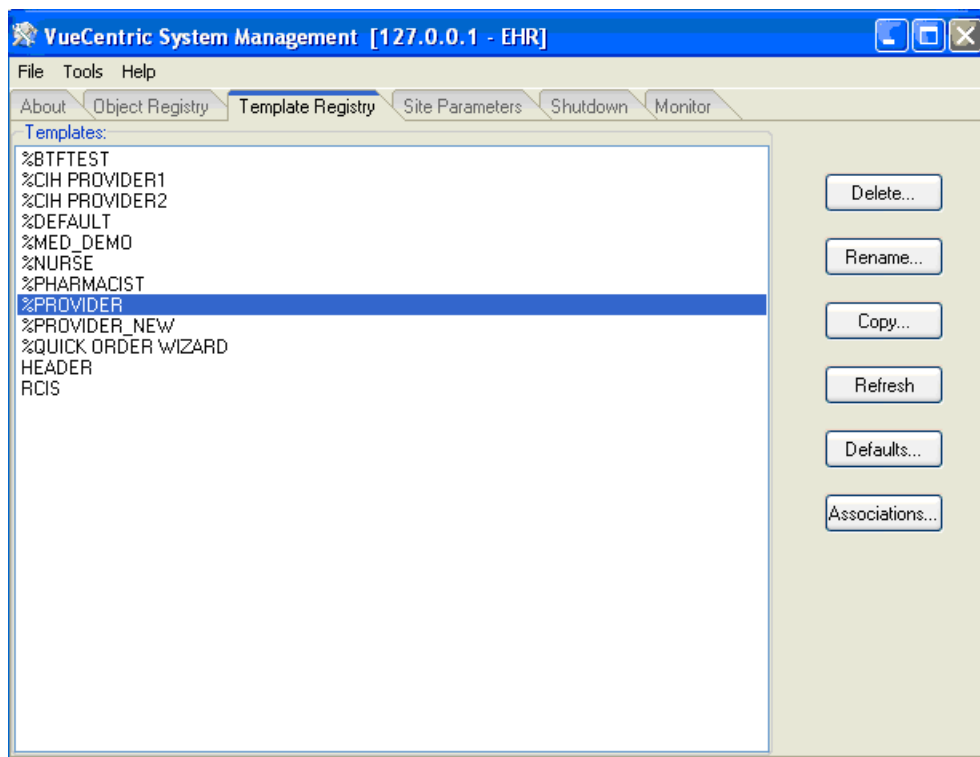


Figure 2-18: Template Register Tab

The **Delete** button deletes the selected template. Any associations with the deleted template are also deleted.

The **Rename** button prompts for a new template name and renames the selected template to that name.

The **Copy** button prompts for a new template name and creates an exact copy of the template under that name.

The *Refresh* button reloads the template list from the host system. This is usually not necessary since the list dynamically updates when additions or deletions are made to the list.



The **Defaults** button displays the Default Templates dialog. This dialog enables changing the default template for various entity types.

The **Associations** button displays the Template Associations dialog. This dialog displays in a tree view format the associations between templates and entities.

### 2.3.1.5.1 File Menu

The following menu options are available under the File menu only when the Template Registry tab is active:

Menu Item	Effect
Import	Enables the importation of a template from an export file (.vtr extension).
Export	Creates an export file (.vtr extension) for the currently selected template.

### 2.3.1.5.2 Template Associations Dialog

The Template Associations dialog displays template associations with the six different entity types.

The dialog can be sorted by template as shown below:

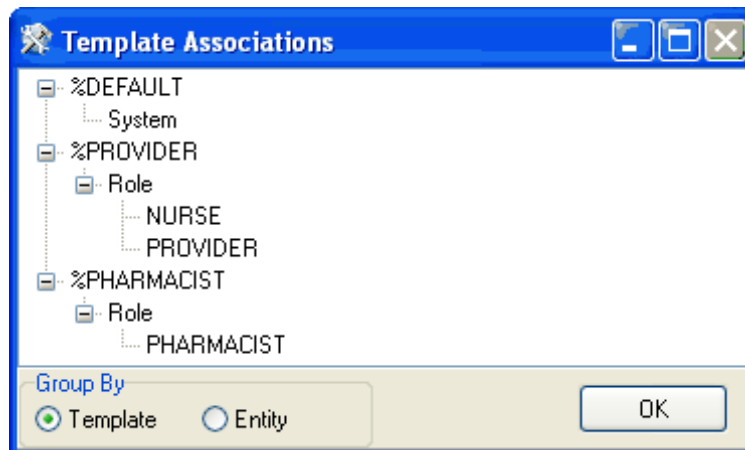


Figure 2-19: Template Associations Sorted by Template

or by entity as below:

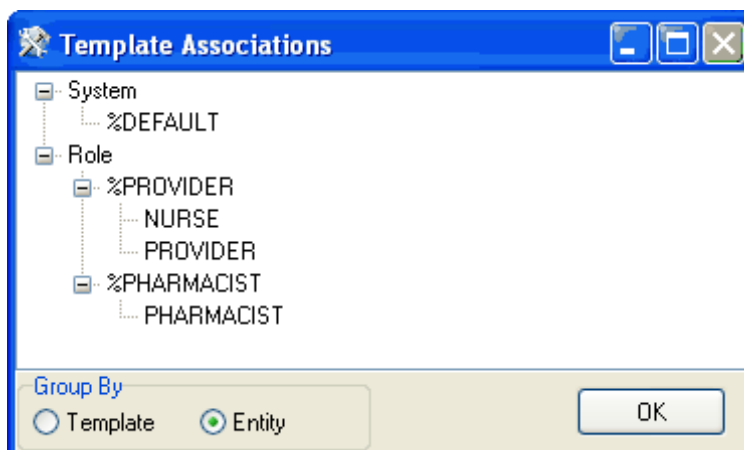


Figure 2-20: Template Associations Sorted by Entity

### 2.3.1.5.3 Template Defaults Dialog

The **Default Templates** dialog permits viewing and modifying associations between templates and entities. These associations determine which template is loaded when a user logs in. Entities are shown in the order of lowest to highest precedence. For example, a template association with a user entity always overrides all other associations.

For system-level associations, only two panes are visible as seen below:

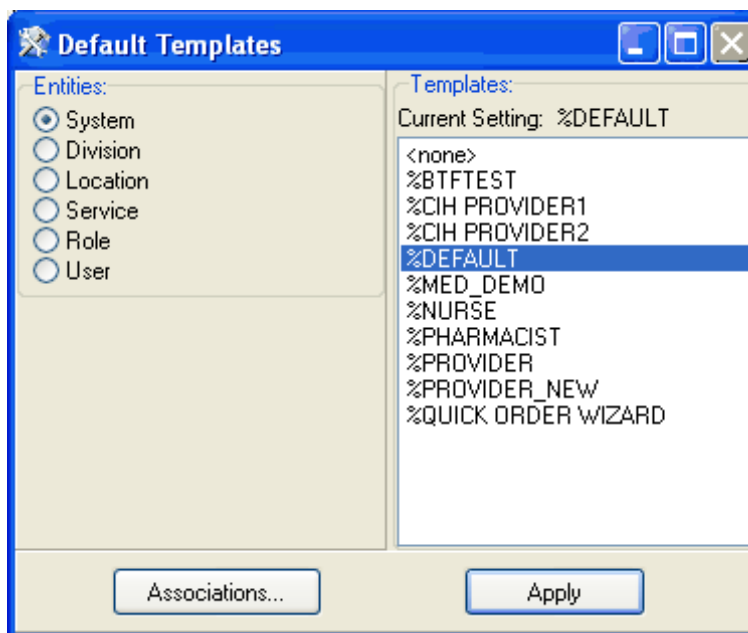


Figure 2-21: System-Level Associations

For all other entity types, a third pane appears listing the different entities available under the selected entity type. For example, if the Role entity is selected, a list of all defined user roles appears as seen below:

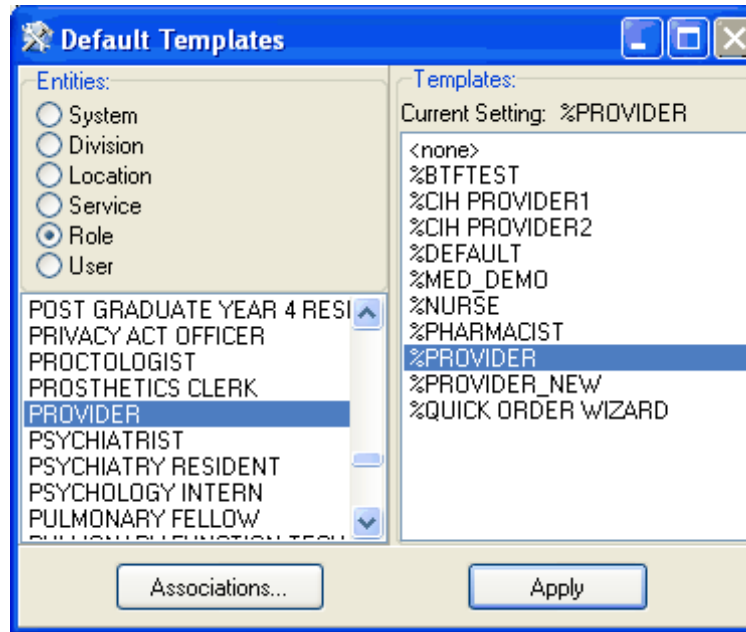


Figure 2-22: Associations for Role Entity

To modify an association, select the desired entity type and, except for the package entity, the desired entry in the entity list. The currently associated template appears in the upper right with that entry selected by default. To change the association, either double-click on the desired template or select a template and click **Apply**. To view current associations for all entity types and templates, click the **Associations** button.

### 2.3.1.6 Site Parameters Tab

The **Site Parameters** tab displays several application level parameters that effect the operation of the *VueCentric*® framework. Because the layout of this tab is determined by the *CIAVM SITE PARAMETERS* parameter template, its appearance can differ from that described here. The package manager can elect to add or remove additional parameter elements depending on local needs.

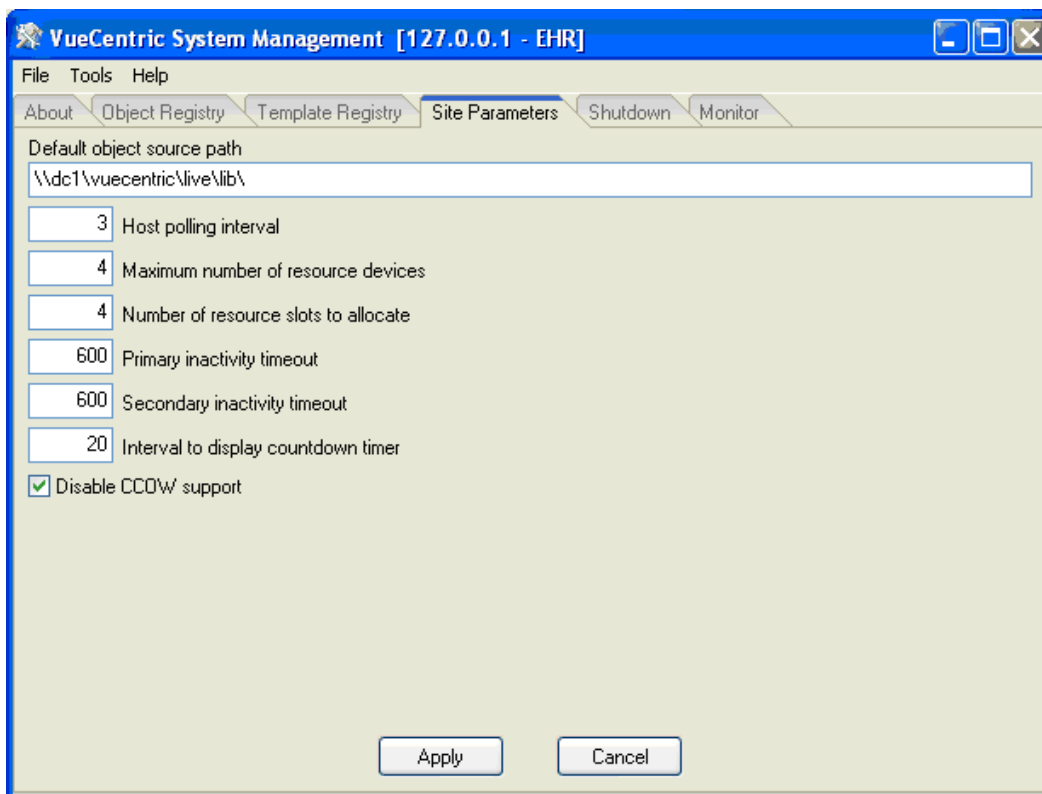


Figure 2-23: Site Parameters Tab

The table below describes each of the parameters and their function.

Parameter	Effect
Default object source path	This is the default path to the location where the master copies of object files are kept. This can be a shared network folder, or a web or ftp address. If the registration entry for an object does not specify a path, this path is used.
Host polling interval	This is the interval, in seconds, that the Communication Services layer polls the host system for completed asynchronous RPC calls and events. Setting this parameter to a larger value placed less load on the host system, but reduces responsiveness. Note that changing this value has an immediate effect on all running <i>VueCentric</i> ® applications.
Maximum number of resource devices	<i>VueCentric</i> ® uses resource devices to control concurrency for background tasks used to service asynchronous remote procedure calls. This parameter controls the pool of resource devices that are available for this purpose. When the Communication Services layer establishes a new connection, it allocates a resource device from this pool. Since several connections can share the same resource device, a load balancing algorithm is utilized to determine which device is allocated.

Parameter	Effect
Number of resource slots to allocate	This parameter determines the number of resource slots to be allocated for each resource device. This dictates how many concurrent processes can run for a given resource device. Additional processes must wait until a slot becomes free before executing. The selection of values for this and the preceding parameter are largely dictated by the number of concurrent users and the host system capacity. Larger values improve responsiveness of asynchronous remote procedure calls, but place a larger burden on the host system by allowing more concurrent background processes to run. This is the inactivity timeout interval, in seconds, after which the Visual Interface Manager will lock the application. If this value is 0, the primary timeout function is disabled.
Primary inactivity timeout	This is the inactivity timeout interval, in seconds, after which the <i>Visual Interface Manager</i> will lock the application. If this value is 0, the primary timeout function is disabled.
Secondary inactivity timeout	This is the inactivity timeout interval, in seconds, after which the Visual Interface Manager will terminate the application. If this value is 0, the secondary timeout function is disabled.
	Before either a primary or secondary timeout occurs, a warning timer appears. This interval, in seconds, determines how long this warning dialog displays before the timeout is finalized.
Disable CCOW support	The <i>VueCentric</i> ® Framework supports context management using a third party, CCOW-compliant context manager. If a CCOW-compliant context manager is not available or to disable its use by the <i>VueCentric</i> ® Framework, check this parameter.

### 2.3.1.7 Shutdown Tab

The Shutdown Tab permits orderly shutdown of all or selected *VueCentric*® sessions.

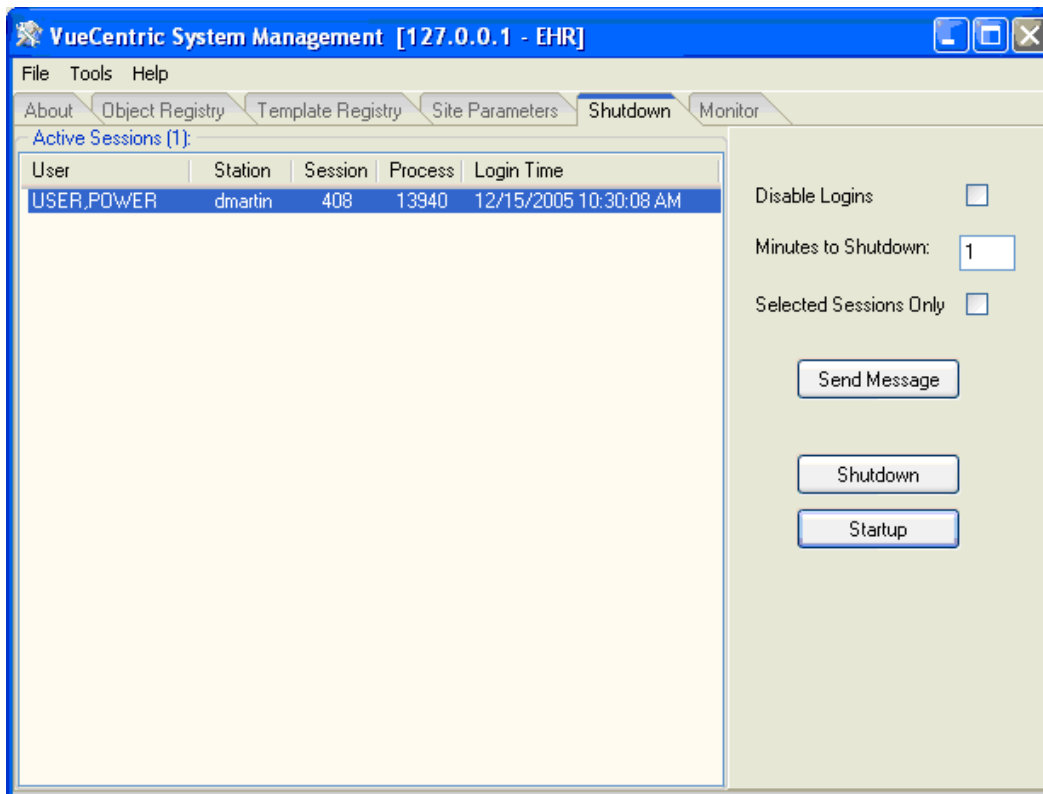


Figure 2-24: Shutdown Tab

The pane on the left displays all currently running VueCentric® sessions. It is updated dynamically as sessions come and go or can be updated manually by right clicking the list and selecting Refresh from the popup menu.

The **Disable Logins** checkbox will inhibit further logins if checked. It does not affect currently running sessions.

**Minutes to Shutdown** determines the delay before running sessions are forced to terminate. Sessions will be notified of the time remaining as shutdown progresses.

The **Selected Sessions Only** checkbox appears when one or more sessions are selected. If checked, the buttons described below affect only those sessions that are selected. Otherwise, all sessions are affected.

The **Send Message** button allows sending a message to selected or all sessions. Messages appear as a popup dialog on the target workstations.

The **Shutdown** button initiates a shutdown sequence for all running sessions, or only selected sessions (see preceding). If performing a system-wide shutdown, further logins are automatically disabled. In either case, a shutdown event is sent to all targeted sessions. The shutdown event initiates a countdown timer which will force the session to terminate when it expires.

The **Startup** button terminates the shutdown sequence by enabling logins and sending a shutdown abort event to all running sessions.

### 2.3.1.8 Monitor Tab

The **Monitor** tab permits the monitoring of system information for remote clients. It requires the deployment of the vcMonitor.dll service plug-in component to each remote workstation that is to be monitored. The component can be deployed like any other service component and should be registered as a dependency (see Dependencies Tab) for either the CIA\_VIM.VIM object or the CIA\_CSS.CSS\_SESSION object so that it is started automatically when a session starts. The service operates in the background and responds to queries by the VueCentric System Management utility.

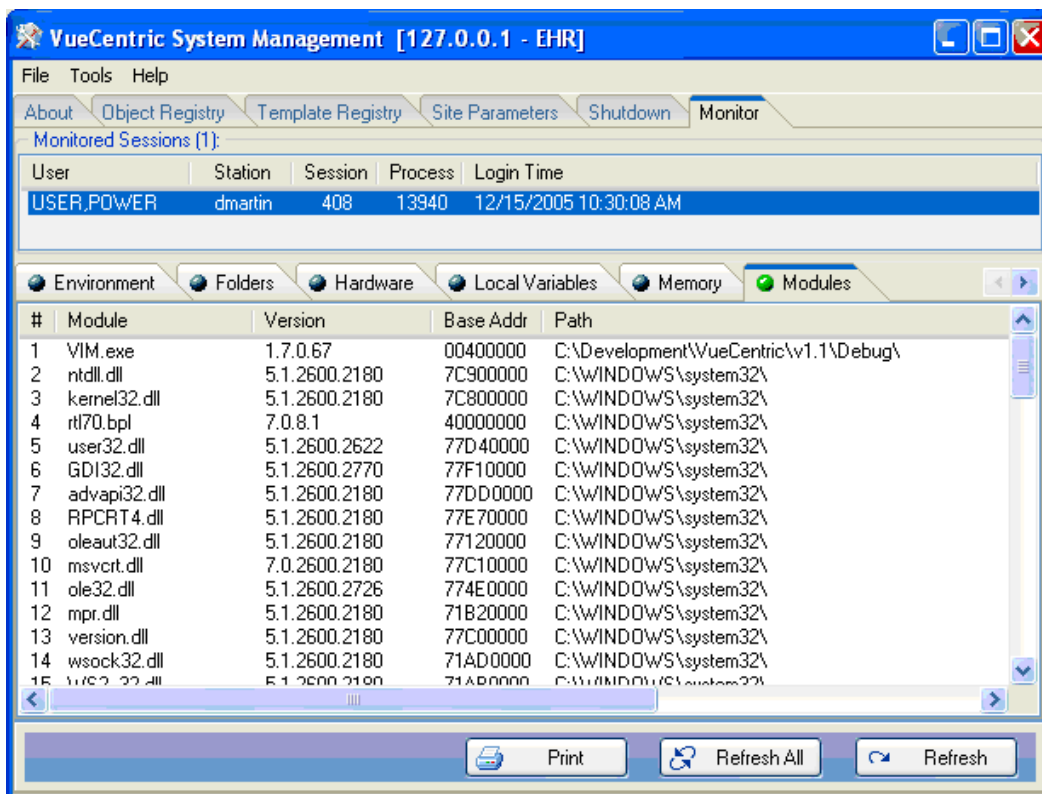


Figure 2-25: Monitor Tab

At the top is a list of active sessions. Only sessions running the vcMonitor service will be visible. This list will change dynamically as users log on and off. To monitor a session, select it from the list. The utility will issue an attach request to the vcMonitor service running under that session. Once the connection is established, a series of tabs will appear at the bottom of the display. Each tab represents a category of information that the vcMonitor service is capable of providing. This will vary depending on the version of the service that is running.

With the exception of the **Trace Log** tab (see below), each tab has three buttons at the bottom. The **Refresh** button issues a remote query for information relevant to the selected tab only. The **Refresh All** button issues a remote query for every tab. The **Print** button allows the printing of the contents of the selected tab.

The **Trace Log** tab behaves in an identical fashion to the Trace Log feature in the Visual Interface Manager. It allows the monitoring of a variety of events on the remote client such as remote procedure calls, host events, context changes, etc. Because enabling this feature might adversely affect performance of the remote client, it is initially turned off by default. Click the **Resume/Suspend** button once to activate the feature, once again to deactivate it. For detailed information on the operation of this feature, see the description of the trace log feature in the help file accompanying the Visual Interface Manager.

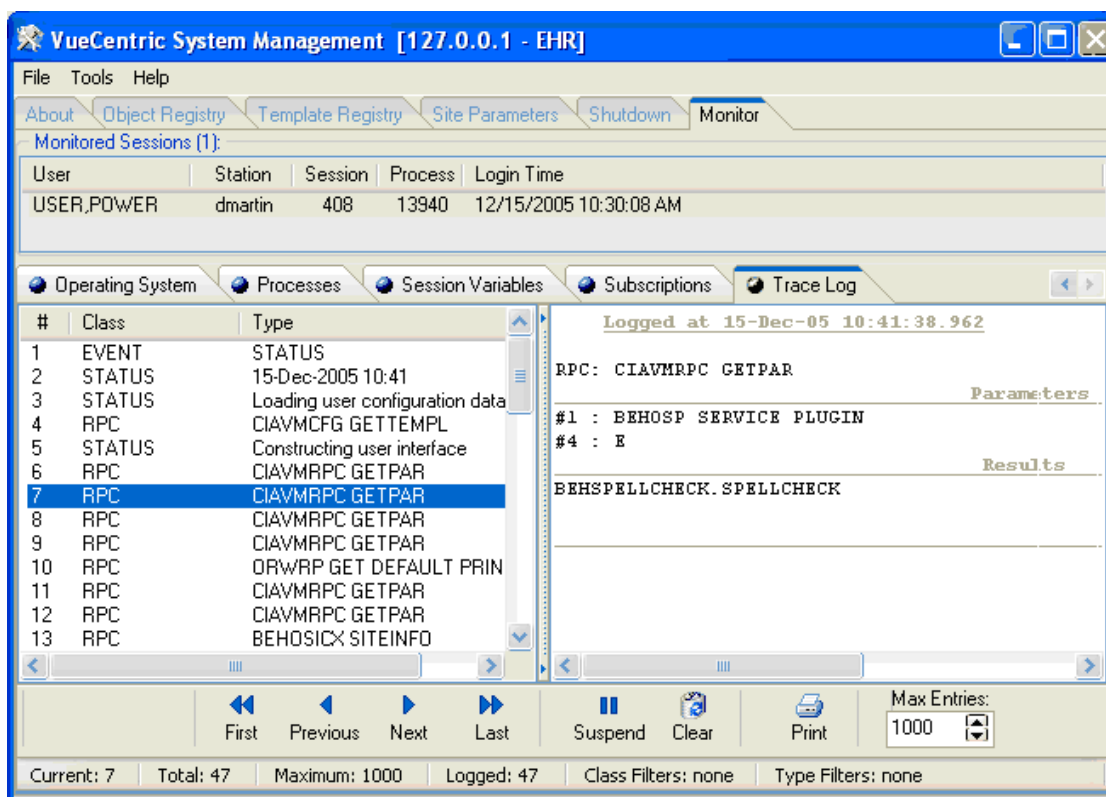


Figure 2-26: Trace Log

### 2.3.2 Ini Configuration (vcIniConfig) Utility

The Ini Configuration utility can be found in the "bin" folder of the RPMS-EHR distribution as the file `vcIniConfig.exe`. This utility performs various maintenance operations on the `VueCentric.ini` control file that controls the updating of core components of the RPMS-EHR application. This utility is usually automatically invoked at the completion of the installation of an RPMS-EHR distribution on the file server, but can also be invoked by the application administrator to update the `VueCentric.ini` file whenever a manual change has been made to a file within the "bin" folder (for example, the `vcBroker.ini` file has been edited or a new license file has been installed).

The utility serves the following functions:



- Synchronizes version information imbedded within the VueCentric.ini file with the respective files in the “bin” folder.
- Provides a simple means to manually edit key settings of the VueCentric.ini file.
- Merges changes into the VueCentric.ini file during the installation of an RPMS-EHR update.
- Provides a shortcut to edit the vcBroker.ini file.

When invoked, the vcIniConfig utility presents the following dialog:

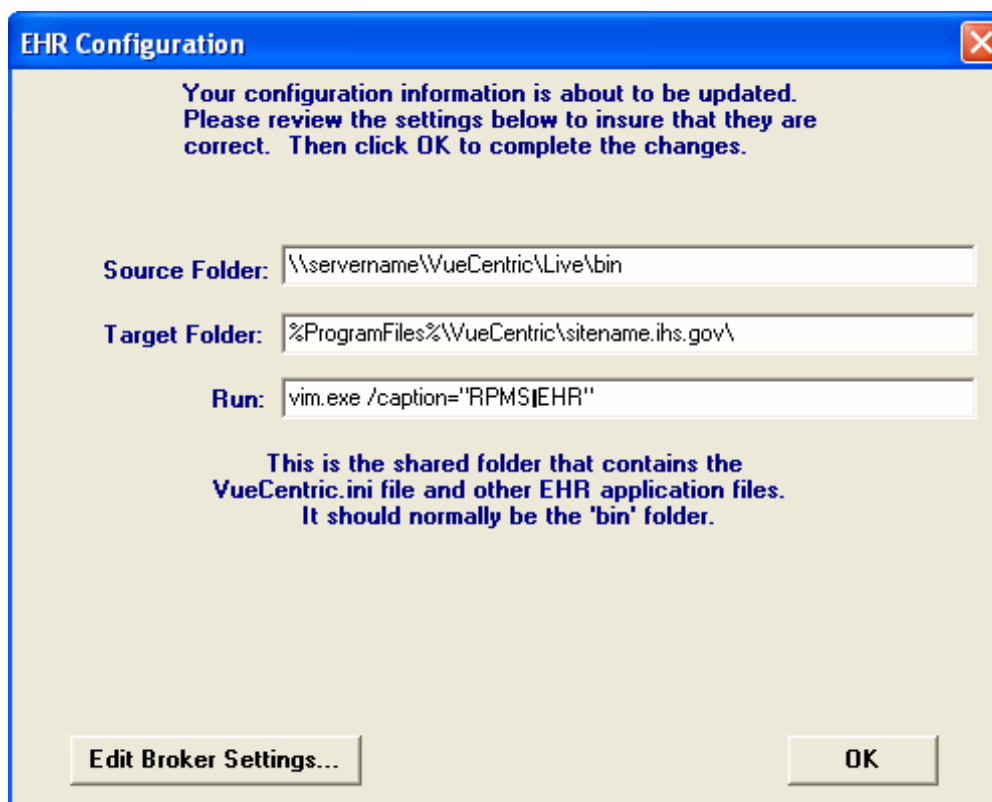


Figure 2-27: Dialog from vcIniConfig Utility

Three key settings from the VueCentric.ini file are displayed and can be edited if necessary. Descriptive text for each setting is displayed near the bottom of the dialog as the corresponding setting is selected for editing. A more detailed description of the function of these settings can be found in the RPMS-EHR Installation Guide.

Clicking the “Edit Broker Settings...” button will open the vcBroker.ini file for editing. Refer to the section on RPC Broker maintenance for more information.

Clicking "OK" will apply any changes made by the user and will resynchronize imbedded version information with the corresponding files in the “bin” folder.

**Note:** The vcIniConfig utility should be invoked whenever a manual change has been made to any file within the “bin” folder. It is never harmful to run this utility, so do not be hesitant to do so.

## 2.4 Routine Descriptions

The VueCentric® Framework has been assigned the namespace designation of "CIAV". The following routines are distributed.

Routine	Description
CIAVCXUS	User context support
CIAVINIT CIAVIN1 CIAVINX	KIDS installation support
CIAVMCFG	Object and template registry APIs
CIAVMEVT	Event management (deprecated)
CIAVMRPC	Shared remote procedure calls
CIAVUTIL	Miscellaneous utilities
CIAVUTPR	Parameter managemen

## 2.5 File List

The VueCentric® Framework has been assigned the file number range of 19930.1 through 19930.9. The following files are distributed:

### 2.5.1 VueCentric Object Registry File (#19930.2)

This file contains information about all components available within the VueCentric® Framework. It is maintained by the VueCentric System Management Utility.

Field Name	#	Datatype	Indexes	Description
PROGID	.01	Text	B-Standard	This is the programmatic identifier of the object and is the primary key for the file.
CLSID	.5	Text		The class identifier (GUID) for this object.
NAME	1	Text	C-Standard	This is a brief text description of the object. This is the object name that is displayed by the 'add object' dialog.
VERSION	2	Text		This is the version of the object that is available from the URL named in SOURCE.
SOURCE	3	Text		This is a URL which can be used to retrieve a copy of the object's executable image. If no explicit path information is provided, the default path defined by the host system is used.
HEIGHT	4	Integer		The default height, in pixels, when an object is created in design mode.

Field Name	#	Datatype	Indexes	Description
WIDTH	5	Integer		The default width, in pixels, when an object is created in design mode.
CATEGORY	6	Pointer(1993 0.21) (multiple)	B - Standard (subfile)	These are the categories under which the object is to be classified. This controls where the object appears in the 'Add Object' dialog of the VIM design editor.
SERIALIZABLE	8	Word Processing		These are properties whose values are to be saved when a snapshot of the visual interface is taken and that appear in the generic property editor of the VIM. These property values are restored when the snapshot is loaded.
INITIALIZATION	9	Word Processing		These are the property initializers for an object. When an object is created, the properties listed here are initialized to the specified values. The format is <name>=<value> or <name>=@<parameter>.. Separate multiple initializers with a carriage return character.
REQUIRED	10	Word Processing		This is a list of URLs of additional files an object needs to run. For example, if an object requires a DLL to be installed, place a URL pointing to the DLL here.
PROPEdit	11	Boolean		If true, the object's internal property editor is invoked by the VIM rather than the default, generic property editor.
MULTIPLE	12	Boolean		If true, multiple instances of the object are allowed to exist concurrently in the same application instance.
DISABLE	13	Boolean		If set to true, the object cannot be loaded by a configuration. Use this feature to take an object out of service.
ALLKEYS	14	Boolean		If true, the user must possess all keys listed in the KEYS multiple to access the object. Otherwise, possession of any one key is sufficient.
HIDDEN	15	Boolean		If true, the object does not appear in the Add Object dialog.
SIDEBYSIDE	16	Boolean		If true, objects are registered in such a manner as to support the co-existence of multiple versions of the object on the same workstation. This is useful in situation where multiple host systems are accessed, each with different client version requirements.

Field Name	#	Datatype	Indexes	Description
SERVICE	17	Boolean		If true, the object represents a service that can be registered with the middle tier. Objects flagged as such are not displayed in the 'add object' dialog. If an object lists a service in its USES multiple, that service is automatically started when the object is loaded.
REGRESS	18	Boolean		If true, the object is retrieved from the repository if the repository version differs from the local version, even if the latter is newer.
NORIGISTER	19	Boolean		If true, the VIM will not register the object with the CSS. This should be set to false unless the object does not require event notification and does not wish to be discovered by other objects or if the object performs its own registration.
KEYS	20	Pointer (19.1) (multiple)	B - Standard (subfile)	Security keys required to access this object. The ALLKEYS field determines whether all listed keys or any one key is required for access. If no keys are specified, access is unrestricted.
USES	21	Pointer (19930.2) (multiple)	B - Standard (subfile)	This is a list of other objects that are required by this object. The Framework ensures that all objects in this list are available and up-to-date.
DOTNET	22	Boolean		If set to true, indicates that the object is a .Net component and requires special handling.
ALIAS	23	Word Processing		This is a list of programmatic identifiers by which this object was previously known. This list is used to automatically update templates that might contain old object references.
TECHNICAL DESCRIPTION	98	Word Processing		This is used to provide technical information that would be of use to a developer utilizing this object.
DESCRIPTION	99	Word Processing		A description of the object's purpose and any special procedures required for its implementation.

## 2.5.2 VueCentric Object Category File (#19930.21)

This file contains category names that can be used to organize objects within the Add Object dialog of the VIM Designer. Use FileMan to create additional entries.

Field	#	Datatype	Indexes	Description
NAME	.01	Text	B - Standard	Name of the category. Category names are hierarchical. Separate hierarchy levels with the backslash character.

### 2.5.3 VueCentric Template Registry File (#19930.3)

Field	#	Datatype	Indexes	Description
NAME	.01	Text	B - Standard	Unique name for the template.
DATA	1	Word Processing		XML data for template

## 2.6 Cross References

Cross references are described in the preceding section.

## 2.7 Callable Routines

This section and those that follow describe the various components that comprise the VueCentric® Framework and the supported means for interacting with those components.

### 2.7.1 \$\$HASKEY^CIAVCXUS

Scope: private.

Parameter	Datatype	Description
Key Name	String	Security key or, if prefixed with the "@" character, a parameter name.
User IEN	Pointer (#200)	IEN of user to check. Defaults to current user.
<return value>	Boolean	Returns true if the specified user possesses the security key or the specified parameter has a value of true.

### 2.7.2 RPC: CIAVCXUS HASKEYS

Scope: private.

Parameter	Datatype	Description
Keys	String	^~delimited list of security keys or parameter names (when prefixed with an "@" character).
<return value>	String	^~delimited list of Boolean values corresponding to the KEYS input parameter.

Checks if the current user has the specified security keys or parameters. Makes a call to \$\$HASKEY^CIAVCXUS for each entry in the KEYS parameter.

### 2.7.3 RPC: CIAVCXUS VALIDPSW

Scope: private.

Parameter	Datatype	Description
Password	String	Encrypted verify code.
<return value>	Boolean	True if the specified verify code is valid.

### 2.7.4 RPC: CIAVMRPC INIT

Scope: private.

Parameter	Datatype	Description
Client Version	String	Version of the VIM client.
<return value>	String List	First list entry is status code (0 if success, -n^Error text if not). Subsequent list entries represent various initialization parameter settings.

Used by the VIM to verify compatibility between the client and server and to retrieve various initialization parameter settings.

### 2.7.5 RPC: CIAVMRPC DISV

Scope: private.

Parameter	Datatype	Description
File Number	Numeric	Number of the file.
Entry Number	Numeric	Internal entry number of the record if this is a set operation, or null or not specified if this is a get operation.
<return value>	Numeric	Internal entry number of the last selected record for the specified file.

Gets or sets the last record selected for the specified file. If an entry number is not specified as input, the stored entry number for the file (if any) is returned. Otherwise, the specified entry number is stored (and its value returned).

### 2.7.6 RPC: CIAVMRPC PKG

Scope: private.

Parameter	Datatype	Description
Package	String	Package name
<return value>	String	Version number of the specified package

Makes a call to \$\$VERSION^XPDUTL to determine the version number of the specified package.

### 2.7.7 RPC: CIAVMRPC PATCH

Scope: private.

Parameter	Datatype	Description
Patch	String	Patch specifier
<return value>	Boolean	True if the specified patch is present

Makes a call to \$\$PATCH^XPDUTL to determine if the specified patch has been installed.

### 2.7.8 RPC: CIAVMRPC GETPAR

Scope: private.

Parameter	Datatype	Description
Parameter	String	Parameter name.
Entities	String	List of ^-delimited entities to check for parameter settings. If not specified, all entities associated with the parameter are checked in order of precedence.
Instance	String	Optional instance specifier for the parameter.
Format	String	Optional return format specifier.
User	Pointer (#200)	Option IEN of user (defaults to current).
<return value>	String	Value of the parameter setting, or null if none found.

Makes a call to \$\$GET^XPAR to return the specified parameter setting.

### 2.7.9 RPC: CIAVMRPC GETPARLI

Scope: private.

Parameter	Datatype	Description
Parameter	String	Parameter name.
Entities	String	List of ^-delimited entities to check for parameter settings. If not specified, all entities associated with the parameter are checked in order of precedence.
Format	String	Optional return format specifier.
User	Pointer (#200)	Option IEN of user (defaults to current).

Parameter	Datatype	Description
<return value>	String	Value of the parameter setting, or null if none found.

Makes a call to GETLST^XPAR to return the values associated with a multi-valued parameter.

## 2.7.10 RPC: CIAVMRPC GETPARWP

Scope: private.

Parameter	Datatype	Description
Parameter	String	Parameter name.
Entities	String	List of ^-delimited entities to check for parameter settings. If not specified, all entities associated with the parameter are checked in order of precedence.
Instance	String	Optional instance specifier for the parameter.
User	Pointer (#200)	Option IEN of user (defaults to current).
<return value>	String	Value of the parameter setting, or null if none found.

Makes a call to GETWP^XPAR to return the value associated with a word processing parameter.

## 2.7.11 \$\$ENT^CIAVMRPC

Scope: private.

Parameter	Datatype	Description
Parameter	String	Parameter name.
Entities	String	Current value of entity list. If this is not null, this is the value returned by the function call. Otherwise, the list of entities associated with the parameter is returned..
User	Pointer (#200)	IEN of user. Defaults to current user.
<return value>	String	Value of the parameter setting, or null if none found.

Returns a ^-delimited list of entity values. If the Entities parameter is specified, this value is returned. Otherwise, a list of precedence-ordered entity values associated with the parameter is returned. This function is used to ensure that a default entity list can be supplied for a parameter when an explicit list is not specified.



## 2.7.12 RPC: CIAVMRPC SETPAR

Scope: private.

Parameter	Datatype	Description
Parameter	String	Parameter name.
Value	String	Parameter value.
Entity	String	Entity with which to associate parameter value. Defaults to package.
Instance	String	Instance value under which to store parameter value. Defaults to 1.
<return value>	String	Value of the parameter setting, or null if none found.

Makes a call to EN^XPAR to set the specified parameter value.

## 2.7.13 RPC: CIAVMRPC GETVAR

Scope: private.

Parameter	Datatype	Description
Variable Name(s)	String or String List	Parameter name or list of parameter names
Namespace	String	Namespace for variables
<return value>	String	List of variable values, one for each specified in the input list. Format for each returned value is: <name>=<value>

Makes a call to \$\$GETVAR^CIANBUTL to return one or more session variable values.

## 2.7.14 RPC: CIAVMRPC SETVAR

Scope: private.

Parameter	Datatype	Description
Variable Name(s)	String or String List	Single or list of parameter name/value pairs in the format: <name>=<value>
Namespace	String	Namespace for variables
Reset	Boolean	If true, all variables in the specified namespace are removed before storing the new values
<return value>	String	Count of variables in input list

Makes a call to \$\$SETVAR^CIANBUTL to set one or more session variable values.

### 2.7.15 RPC: CIAVMRPC GETIDX

Scope: private.

Parameter	Datatype	Description
File Number	Numeric	Number of the file from which to fetch entries
Format Flag	Boolean	If false, only .01 field values are returned. If true, values returned as: <ien>^<.01 field value>
<return value>	String List	List of values in format determined by format flag

Returns a list of all file entries from the specified file. This RPC should only be used for files with a small number of entries.

### 2.7.16 RPC: CIAVMRPC STRTODAT

Scope: private.

Parameter	Datatype	Description
Value	String	Text value to convert.
Format	String	Optional format specifier for %DT call. Defaults to "TS".
<return value>	FileMan Date/Time	VFileMan date/time value corresponding to input, or null if input was invalid.

### 2.7.17 \$\$VERCMP^CIAVMRPC

Scope: private.

Parameter	Datatype	Description
Version1	String	First version number
Version2	String	Second version number
Level	String	Version level to check (1-4). Defaults to 4 (all levels)
<return value>	Integer	One of: 1 = Version1>Version2 0 = Version1=Version2 -1 = Version1<Version2

Compares two version values and returns the result of the comparison. A version value is of the format *major.minor.release.build*. The level parameter determines the level to which the comparison is made (1 would mean check only the major version number; 3 would mean check the major, minor, and release version numbers, ignoring the build number).

## 2.7.18 \$\$TMPGBL^CIAVMRPC

Scope: private.

Parameter	Datatype	Description
Identifier	String	Optional unique identifier for temporary global.
<return value>	Boolean	A global reference that can be used for temporary storage of data.

Returns a global reference that can be used to store temporary data. This is typically used to obtain a global reference for returning data for a remote procedure call. The returned reference is guaranteed to contain no data. If the content of the reference is not used to return data for a remote procedure call, it should be deleted when no longer needed. Do not use this reference to persist data across remote procedure calls.

## 2.7.19 RPC: CIAVUTIL SDINIT

Scope: private.

Parameter	Datatype	Description
Delay	Integer	Number of seconds to before shutdown is enforced. Minimum value is 30 seconds. An optional second ^-delimited piece can be specified to override the default warning message that is sent to users.
Disable	Boolean	If true, further application logins are prohibited. This takes effect immediately, regardless of the delay setting.
LoginUsers	String List	Optional list of users or sessions that will be the target of the shutdown request. If not specified, all sessions are targeted.
<return value>	String	A confirmation message that can be displayed to the user.

Issues a remote shutdown request to specified users and/or sessions.

## 2.7.20 RPC: CIAVUTIL SDABORT

Scope: private.

Parameter	Datatype	Description
Enable Logins	Boolean	If true, application logins are enabled
Users	String List	Optional list of users or sessions that will be the target of the abort request. If not specified, all sessions are targeted
<return value>	String	A confirmation message that can be displayed to the user.

Issues a shutdown abort signal to specified users and/or sessions. If a shutdown request is in progress for the specified target(s), it is immediately aborted and the user is notified.

## 2.7.21 RPC: CIAVUTIL MSGLOGIN

Scope: private.

Parameter	Datatype	Description
Message	String	Message that will be displayed to users attempting to login to the application. If not specified, the current message is returned and no further action taken.
<return value>	String	If this is a get operation, returns the value of the current login inhibition message. If this is a set operation, no value is returned.

Gets or sets the current login inhibition message. If the Message parameter is not specified, the current setting is returned. A null return value in this case indicates that logins are not inhibited. If the Message parameter is specified, this value is stored as the current login inhibition message. A null value for this parameter would effectively enable logins for the application. Any other value would inhibit logins for the application with that message being displayed to users attempting to login.

## 2.7.22 RPC: CIAVUTPR GETTPL

Scope: private.

Parameter	Datatype	Description
Parameter Template	Pointer (#8989.52) or String	Name or IEN of an entry from the PARAMETER TEMPLATE file.
<return value>	String	List of parameters belonging to the specified template in the format: IEN^Name^Display Text^Data Type^Domain^Help

Returns a list of parameters that are members of the specified parameter template.

## 2.8 External Relations

The VueCentric<sup>®</sup> Framework has a number of external package dependencies as noted in the following table:

Package	Version	Dependency
CIA RPC Broker	1.1	Framework uses the CIA RPC broker for client-server communication
CIA Utilities	1.1	Numerous dependencies throughout
Kernel	8.0	Numerous dependencies throughout
PIMS	5.3	The patient and encounter context support code make a number of calls to PIMS routines
Toolkit	7.3	Numerous dependencies throughout

Package	Version	Dependency
VA FileMan	22	Numerous dependencies throughout

## 2.9 Internal Relations

There are no significant internal relations for this package.

## 2.10 Exported Options

Option	Type	Description
CIAV DEFAULT TEMPLATE	Action	Permits modifying the default template for the package
CIAV MANAGER	Menu	Menu of VueCentric® Framework management tasks
CIAV SHOW USERS	Action	Displays a list of currently active user sessions.
CIAV SHUTDOWN ABORT	Action	Aborts a shutdown sequence in progress and re-enables application logins
CIAV SHUTDOWN START	Action	Disables application logins and initiates a shutdown sequence for all running VueCentric® applications
CIAV SITE PARAMETERS	Action	Permits editing configuration parameters for the VueCentric® Framework
CIAV VUECENTRIC	Broker Context	Controls user access to the VueCentric® Framework. For a user to have access to the framework, one of the following conditions must be met: The user must have programmer privilege. The CIAV VUECENTRIC option must be on the user's secondary menu. The CIAV VUECENTRIC option must be a menu item under the XUCOMMAND menu (grants access to all users)

## 2.11 Exported Security Keys

Key	Description
CIAV COMPOSE	Grants compose mode privilege in the VIM designer. See the VIM online help documentation for a description of this capability.
CIAV DESIGN	Grants access to the VIM designer. See the VIM online help documentation for a description of this capability.
CIAV SHOW USERS	Grants access to the VueCentric System Management Utility.

Key	Description
CIAV SITE MANAGER	Aborts a shutdown sequence in progress and re-enables application logins

## 2.12 Exported Protocols

Protocol	Type	Description
CIAV USER TERMINATE	Action	Removes user configuration options upon termination of the user account. Is attached to the XU USER TERMINATE protocol during installation.

## 2.13 Exported Parameters

Parameter	Instance Type	Value Type	Precedence	Description
CIAMV COUNTDOWN INTERVAL		Integer	User, System	This value is the number of seconds to display the timeout warning dialog.
CIAMV DEFAULT SOURCE		String	System	This is the default path to the object repository. When retrieving an object that has no explicit path specified for its source, this path is used.
CIAMV DEFAULT TEMPLATE		Pointer (19930.3)	User, Class, Service, Location, Division, System	Specifies the template to be loaded if no specific template is requested
CIAMV DISABLE CCOW		Boolean	User, System	If yes, CCOW support is disabled in the client application even if a CCOW context manager has been installed
CIAMV OBJECT FAVORITES	Integer	String	User	Used to store personal favorites for the add object dialog in the Visual Interface Manager.
CIAMV PRIMARY TIMEOUT		Integer	User, System	This value sets the number of seconds that the application will remain idle before initiating an application lockout countdown.
CIAMV SECONDARY TIMEOUT		Integer	User, System	This value sets the number of seconds that the application will remain idle before initiating an application lockout countdown.

## 2.14 Exported Mail Groups

Mail Group	Description
VUECENTRIC TECH SUPPORT	Used by the VUECENTRIC TECH SUPPORT device to distribute technical support requests to appropriate recipients.

## 2.15 Archiving and Purging

There are no archiving or purging requirements within this software.

## 2.16 Components

### 2.16.1 Visual Interface Manager

The Visual Interface Manager (VIM) provides a number of services:

- Acts as an intelligent container for components
- Provides access to persistent state information
- Defines the visual relationship of components to one another
- Possesses a design feature that allows the tailoring of the environment under user control
- Initializes and prepares the Component Support Services for use by components
- Provides menu management

When the user logs in, the VIM accesses the Object and Template Registries residing on the host system to retrieve information about the requested configuration. Using this information, the VIM reconstructs the visual interface. The requested configuration can be the user's private configuration or one that is defined for a specific user role or function. Specific templates can be requested using the appropriate command line parameter when invoking the VIM. In the absence of such, the VIM retrieves the user's customized template if one exists or a default template that is determined by the host configuration and can be specific to the user, user class, department or institution.

Before each component is loaded into the visual interface, the VIM consults the Component Management Service (CMS) to determine if the object is available and if the user is permitted to use it. If the latest version of the object is not currently installed, the CMS retrieves it and any additional required components from the Object Repository and deploys them to the local machine. If all of these conditions are met, the VIM instantiates a copy of the component, performs any initializations specified by the Object Registry, and restores any saved state information (e.g., the color of the component as set by the user). The VIM then registers the component with the Component Support Services (CSS). The purpose of the registration process is to allow the CSS to determine if the component implements any of the event interfaces it

or its plug-in services publishes and to allow other components to discover the object at runtime. For each implemented event interface, the CSS automatically connects the interface to the corresponding event producer. Thus, all a component must do to subscribe to an event is to simply implement the corresponding interface. The registration process takes care of the rest.

### 2.16.1.1 Command Line Parameters

The VIM executable recognizes a variety of command line parameters. Some of these are provided to facilitate object development and testing. Command line parameters are case-insensitive can be preceded by a hyphen or forward slash, but neither is required. Some parameters accept a value, which should be separated from the parameter by an equal sign (without leading or trailing spaces). Parameter values with imbedded spaces must be surrounded by quotation marks. The VIM recognizes the following command line parameters:

Parameter	Value	Description
blank		Suppresses the loading of a template upon login. Instead, the VIM presents the user with a blank desktop. This is useful for testing objects and for designing templates where an initial configuration is not desired.
caption		Sets the caption for the application's main form. The caption can also be set in design mode as a desktop property and saved as a template
clrversion	<version>	Specifies the .Net framework version to load. Only use to force a specific version of the .Net framework. By default, the most current version installed is loaded
color	<info>, <popup>, <status>, <progress>	Sets default colors for balloon alerts, popup messages, status bar, and progress bar. Each color setting is in RGB format. Prefix each with a "\$" character to use hexadecimal notation.
debug		Enables debugging on the remote host
design		Activates design mode on application startup if the user has design mode privilege.
fixtabs		Modifies the handling of the tab character that can improve tabbing behavior in some ActiveX objects
help	<filename>	Permits overriding the default help file associated with the application
host		Same as the server parameter. See description of that parameter for details
icon	<filename>	Allows the specification of an icon file. When specified, the contained icon will be used in place of the application's default icon. This can also be set as a desktop property when in design mode and saved as part of a configuration



Parameter	Value	Description
image	<filename>	Allows the specification of an image file. When specified, the contained image (several image formats are supported) is displayed in place of the blank desktop when the VIM is in the logged out state. This can also be set as a desktop property when in design mode and saved as part of a configuration
log	<filename>	Logs unhandled exception to the Windows application event log. Server name is the remote machine where the event log resides. If the server name is not specific, the event log on the local machine is used. This option only works on Windows versions based on the NT kernel. It is ignored on other platforms
noccow		Disables CCOW interaction. When this parameter is specified, the VIM does not instruct the CSS to attempt to contact a CCOW context manager even if one is present. This is useful for debugging and when a second instance of the VIM is desired that does not share context with other applications
nocompose		Disables compose mode even if user possesses the necessary security key. This is provided primarily for debugging purposes
nodesign		Disables design mode even if user possesses the necessary security key. This is provided primarily for debugging purposes.
nojoin		Disables automatic joining of CCOW context. By default, the VIM instructs the CSS to connect to a CCOW context manager (if one is detected and the noccow command line parameter is not specified) and join the common context. If nojoin is specified, the CSS still connects to the context manager, but does not join the context. Unlike the noccow parameter, the user still retains the option of manually joining the common context by right-clicking the CCOW icon in the lower right corner of the application window and selecting the appropriate popup menu choice
notimeout		Disables auto timeout. By default, the VIM logs out after a site-specified period of keyboard and mouse inactivity. This option causes the VIM to ignore the timeout
noupdate		Disables the automatic updating of objects. When this parameter is specified, the VIM does not retrieve objects from the Object Repository. This parameter is provided primarily for debugging purposes. It can also have application in environments that use system administrator tools to push software updates to workstations

Parameter	Value	Description
server	<hostname>:<port>:<uci> or <host ip>:<port>:<uci>	Specifies information about the target host. If this parameter is not specified, the CSS will either select the default host or present the user with a choice of hosts taken from the vcBroker.ini file. Which action is taken depends upon how the RPC broker has been configured on the workstation. If the port number is omitted, the default port for the host will be used. If the UCI specification is omitted, the default UCI for the host will be used.
showflags		Shows active command line flags in the status panel. This is useful for debugging purposes to have a visual indicator of which command line flags are in effect.
template	<template name>	Loads the named template upon login. By default, the VIM loads the user's personal configuration template upon login or, in the absence of one, the user's default template as determined by site configuration. This parameter overrides this default behavior
timeout	<primary>, <secondary>, <countdown>	Sets the timeout intervals, in seconds, for the primary timeout (after which the application is locked), the secondary timeout (after which the application is logged out), and the countdown timer (which determines how long the timeout warning appears).
trace		Enables a special trace mode that causes the CSS to report internal events such as RPC calls and host events via the TRACE event. Any object subscribing to the TRACE event can view this data. The VIM intercepts the TRACE event and adds the information to a cumulative event log.
updateall		Overrides the default behavior of updating objects only if they are not present or newer versions are available in the Object Repository by forcing updates to occur every time an object is requested. This is provided primarily for debugging purposes
verbose		Displays additional status information for debugging
windowstate	<state>	Sets the initial window state. Possible values are min, max, nml for minimized, maximized, or normal, respectively

Typical usage of command line parameter for a production application might look something like the following:

```
vim.exe /server=PRODUCTION /autologin /caption="VueCentric IHS-EMR" /image="splash.jpg"
```

whereas, command line parameters for debugging purposes might be:

```
vim.exe /server=localhost /autologin /debug /noupdates /verbose /notimeout /showflags /template=TEST
```

### 2.16.1.2 VIM Automation Object

Components typically have minimal interaction with the VIM. Rather, they react passively to control by the container. The component's principal interaction is with the CSS or another component. The VIM does, however, register an automation interface with the CSS that is accessible by components. This interface is defined as follows:

Programmatic Id: CIA\_VIM.VIM

Class GUID: A45DCEDF-22F6-4F2B-BA12-DF5D5765ED68

Default Interface:IVIM

#### 2.16.1.2.1 Properties

Parameter	Datatype	Access	Description
Caption	String	RW	The caption of the application's main form.
Colors	String	RW	Initializes the custom color list used by the property editor for color property types. Up to 15 custom colors can be specified. Separate each color specification with a "^". Colors should be specified in RGB format using hexadecimal notation.
Font	IFontDisp	RW	The default font for the application. Changes to this property are automatically propagated to all objects that also publish a font property.
Height	Integer	RW	Height (in pixels) of the main form.
HelpFile	String	RW	Default help file for the application.
Icon	Integer	RW	Index of the custom icon (see Icons property) to use for the application title bar. A value of -1 forces the default icon to be used.
Icons	String	RW	The name of the file that contains the icons to be used by the application. These icons can be used by custom menu items and other application elements that require icons. If null, the application's default icons are used.
Image	String	RW	The name of the file that contains an image to be displayed while the application is in a logged out state. If null, no image is displayed.
InfoColor	Color	RW	The background color of the balloon alert.
PopupColor	Color	RW	The background color of the modeless message dialog.
PopupColor2	Color	RW	Secondary background color of the modeless message dialog for creating gradient effects.
ProgressColor	Color	RW	The background color of the progress bar that appears during forced application shutdown.
StatusColor	Color	RW	The background color of the status bar.

Parameter	Datatype	Access	Description
Version	String	R	The current version of the VIM.
Width	Integer	RW	Width (in pixels) of the main form.

#### 2.16.1.2.2 BringToFront

Parameter	Datatype	Description
ObjRef	IUnknown	IUnknown interface reference of the object to be brought to the foreground.
<return value>	Boolean	Returns true if the referenced object was found.

This function can be invoked to insure that the referenced object is visible within the visual interface. If other windows obscure the object, it is brought to the top of the Z-order. If the object is located on a tab or pane that is not currently active that tab or pane will be automatically activated.

#### 2.16.1.2.3 BringToFront2

Parameter	Datatype	Description
ProgID	String	Programmatic identifier of the object to be brought to the foreground.
<return value>	Boolean	Returns true if the referenced object was found.

This function can be invoked to insure that the referenced object is visible within the visual interface. If other windows obscure the object, it is brought to the top of the Z-order. If the object is located on a tab or pane that is not currently active that tab or pane will be automatically activated.

#### 2.16.1.2.4 Lock

This procedure causes the VIM to minimize all open windows and present a modal dialog prompting for the user's verify code. Only by entering a valid verify code can the application be restored.

#### 2.16.1.2.5 Logout

This procedure causes the VIM to forcibly logout. First all context objects first revert to a null state and fire their respective context change events to all subscribers. Next, all loaded visual components are unloaded. Next, all running services are shutdown. Finally, the VIM enters its logout state. If autologin is enabled, this causes the application to terminate. Otherwise, the application displays its initial logon screen.

### 2.16.1.2.6 Popup

Parameter	Datatype	Description
ObjRef	IUnknown	IUnknown interface reference of the object to be displayed modally.
Title	String	The caption to be displayed for the popup dialog.
<return value>	Boolean	Returns true if the referenced object was found.

This function causes the referenced object to appear in a modal window. The VIM accomplishes this by temporarily moving the object from its parent within the interface to a modal window. When the modal window is closed, the object is returned to its original position.

Only one object can exist in a popup window at a time. If an object is already displayed in a popup window, it is restored to its previous state before presenting the new popup. If ObjRef is null, any existing popup is closed, but no new popup is presented.

### 2.16.1.2.7 Popup2

Parameter	Datatype	Description
ProgID	String	Programmatic identifier of the object to be brought to the foreground.
Title	String	The caption to be displayed for the popup dialog
<return value>	Boolean	Returns true if the referenced object was found.

This function causes the referenced object to appear in a modal window. The VIM accomplishes this by temporarily moving the object from its parent within the interface to a modal window. When the modal window is closed, the object is returned to its original position.

Only one object can exist in a popup window at a time. If an object is already displayed in a popup window, it is restored to its previous state before presenting the new popup.

## 2.16.2 Component Support Services

The Component Support Services (CSS) provide a suite of services that allow objects to access context information (current user, current patient, current encounter, etc.), host-based data (through RPC calls) and receive event notifications (change in selected patient, for example). Unlike the VIM, which is a standalone executable, the CSS is implemented as an in-process COM server that executes in the background and is shared by all objects within an application instance. Unlike previous versions that

defined a single automation server with over a dozen interfaces, the CSS implements two automation servers and defines four interfaces. Many of the interfaces previously declared by the CSS that provided access to context information are implemented as separate automation objects that are registered with the CSS at runtime.

The two automation servers defined by the CSS are the Server (CIA\_CSS.CSS\_Server) and the Session (CIA\_CSS.CSS\_Session) objects. The sole purpose of the Server object is to instantiate and maintain a shared instance of the Session object. A component cannot directly create an instance of the Session object. Rather, it must first create an instance of the Server object and request a reference to the Session object. Having obtained such a reference, the component has no further need of the Server object and can release it.

### 2.16.2.1 Server Automation Object

The server automation object has the following definition:

Programmatic Id: CIA\_CSS.CSS\_Server  
 Class GUID: 8C061A95-8FCE-41A7-A806-66B02E5CE6EF  
 Default Interface:ICSS\_Server

#### 2.16.2.1.1 Properties

Parameter	Datatype	Access	Description
Session	ICSS_Session	R	Reference to the session automation object.

Components desiring access to middle tier services should do so by first creating an instance of the server automation object and then obtaining a reference to the session automation object by reading the value of the Session property. Once this is done, the reference to the server object can be released.

### 2.16.2.2 Session Automation Object

The session automation object provides access to middle tier services. While the session object has a programmatic identifier and class GUID, it cannot be instantiated directly, but must be requested from the server automation object (see above). The session automation object has the following definition:

Programmatic Id: CIA\_CSS.CSS\_Session  
 Class GUID:8C061A95-8FCE-41A7-A806-66B02E5CE6EF  
 Default Interface:ICSS\_Session  
 Event Interface:ICSS\_SessionEvents

### 2.16.2.2.1 Properties

Parameter	Datatype	Access	Description
CCOWState	Enum	R	The current CCOW state. Possible values are: ccowBroken = Not participating ccowChanging = Context change in progress ccowJoined = Participating ccowNone = No context manager ccowDisabled = Disabled by host
DebugMod	Boolean	RW	Indicates whether or not the CSS is in debug mode.
DomainName	String	R	The domain name of the currently connected host. If there is no active connection, returns null.
HostAddress	String	R	The IP address of the host system that is currently connected. If no connection is active, returns null.
HostDateTime	DateTime	R	The current date and time as returned by the host system. If there is no active connection, returns a NULL_DATE value.
HostName	String	R	The name of the host system that is currently connected. If no connection is active, returns null.
HostPort	Integer	R	The port number of the active connection. If no connection is active, returns 0.
Param(Name)	OleVariant	RW	Allows an object to register an arbitrary named parameter and value that can be accessed by other objects and by the ReplaceParams method. The Name parameter can consist only of alphanumeric characters and the underscore
SessionID	Integer	R	The unique identifier for the current session.
SiteName	String	R	The site name of the currently connected host.
TraceMode	Boolean	RW	Enables or disables trace mode.
Version	String	R	The version of the CSS.

### 2.16.2.2.2 CallRPCAbort

Parameter	Datatype	Description
Handle	Integer	Handle of the asynchronous call to abort.

This procedure causes the asynchronously executing remote procedure identified by *Handle* to be aborted.

### 2.16.2.2.3 CallRPCAsync

Parameter	Datatype	Description
RPCName	String	Name of the remote procedure to be invoked. If an execution context other than the default is desired, precede the RPC name with a context name and the '^' delimiter. To specify a version number, prefix the RPC name with the version number enclosed in vertical bars.
Parameters	OleVariant	Parameters to be passed to the remote procedure
Callback	ICSS_Session Events	Callback interface to be invoked on completion of the remote procedure
PlainText	Boolean	If true, data returned to the callback interface is in plain text format. Otherwise, format is in comma text format.
<return value>	Integer	Handle that uniquely identifies this asynchronous call.

This function invokes the remote procedure named in *RPCName* in asynchronous mode, passing it the parameters listed in *Parameters* (see description of *CallRPCList* for details on formatting of parameters). A negative return value indicates that the remote procedure failed. Otherwise, the return value is a unique handle that identifies the call. Upon completion of the remote procedure, the callback interface identified by the *Callback* parameter is invoked. See a description of the *ICSS\_SessionEvents* interface for details.

### 2.16.2.2.4 CallRPCBool

Parameter	Datatype	Description
RPCName	String	Name of the remote procedure to be invoked. If an execution context other than the default is desired, precede the RPC name with a context name and the '^' delimiter. To specify a version number, prefix the RPC name with the version number enclosed in vertical bars.
Parameters	OleVariant	Parameters to be passed to the remote procedure
<return value>	Integer	Output of remote procedure call as a Boolean value..

This function invokes the remote procedure named in *RPCName*, passing it the parameters listed in *Parameters* (see description of *CallRPCList* for details on formatting of parameters). The return value is a Boolean result.



**2.16.2.2.5 CallRPCDate**

Parameter	Datatype	Description
RPCName	String	Name of the remote procedure to be invoked. If an execution context other than the default is desired, precede the RPC name with a context name and the '^' delimiter. To specify a version number, prefix the RPC name with the version number enclosed in vertical bars.
Parameters	OleVariant	Parameters to be passed to the remote procedure
<return value>	DateTime	Output of remote procedure call as a DateTime value.

This function invokes the remote procedure named in *RPCName*, passing it the parameters listed in *Parameters* (see description of *CallRPCList* for details on formatting of parameters). The return value is a DateTime datatype.

**2.16.2.2.6 CallRPCInt**

Parameter	Datatype	Description
RPCName	String	Name of the remote procedure to be invoked. If an execution context other than the default is desired, precede the RPC name with a context name and the '^' delimiter. To specify a version number, prefix the RPC name with the version number enclosed in vertical bars.
Parameters	OleVariant	Parameters to be passed to the remote procedure
<return value>	Integer	Output of remote procedure call as a 32-bit value.

This function invokes the remote procedure named in *RPCName*, passing it the parameters listed in *Parameters* (see description of *CallRPCList* for details on formatting of parameters). The return value is a 32-bit integer value.

**2.16.2.2.7 CallRPCList**

Parameter	Datatype	Description
RPCName	String	Name of the remote procedure to be invoked. If an execution context other than the default is desired, precede the RPC name with a context name and the '^' delimiter. To specify a version number, prefix the RPC name with the version number enclosed in vertical bars.
Parameters	OleVariant	Parameters to be passed to the remote procedure
<return value>	String	Output of remote procedure call in CommaText format. (see discussion)

This function invokes the remote procedure named in *RPCName*, passing it the parameters listed in *Parameters*. The return value is a string in CommaText format. This format can be converted to a TStrings descendant by setting it into the CommaText property.

The *Parameters* argument can be any OleVariant datatype, including a variant array. If it is scalar (non-array) datatype, it is passed as a single argument to the remote procedure call. If it is an array, each element of the array is passed as an argument. If an array element is itself an array, it is passed as a list argument to the remote procedure call.

```
var
  lstXYZ: TStringList;
begin
  lstXYZ:=TStringList.Create;
  lstXYZ.CommaText:=vcSession.CallRPCList('FETCH',[Name,SSN]);
  ...
end;
```

#### 2.16.2.2.8 CallRPCStr

Parameter	Datatype	Description
RPCName	String	Name of the remote procedure to be invoked. If an execution context other than the default is desired, precede the RPC name with a context name and the '^' delimiter. To specify a version number, prefix the RPC name with the version number enclosed in vertical bars.
Parameters	OleVariant	Parameters to be passed to the remote procedure
<return value>	String	Output of remote procedure call as a string

This function invokes the remote procedure named in *RPCName*, passing it the parameters listed in *Parameters* (see description of *CallRPCList* for details on formatting of parameters). The return value is a string.

#### 2.16.2.2.9 CallRPCTex

Parameter	Datatype	Description
RPCName	String	Name of the remote procedure to be invoked. If an execution context other than the default is desired, precede the RPC name with a context name and the '^' delimiter. To specify a version number, prefix the RPC name with the version number enclosed in vertical bars.

Parameter	Datatype	Description
Parameters	OleVariant	Parameters to be passed to the remote procedure.
<return value>	String	Output of remote procedure call in plain text format.(see discussion)

This function invokes the remote procedure named in *RPCName*, passing it the parameters listed in *Parameters*. The return value is a string in plain text format. This format can be converted to a TStrings descendant by setting it into the Text property.

The *Parameters* argument can be any OleVariant datatype, including a variant array. If it is scalar (non-array) datatype, it is passed as a single argument to the remote procedure call. If it is an array, each element of the array is passed as an argument. If an array element is itself an array, it is passed as a list argument to the remote procedure call.

#### 2.16.2.2.10 CanDisconnect

Parameter	Datatype	Description
Survey	Boolean	If true, all internal context participants for all contexts are surveyed. If any participant declines, the return value is false.
<return value>	Boolean	True if disconnect is permissible. This would only be false if the Survey parameter was true and a context participant declined.

This function is used to prepare the environment for termination of the host connection. If the Survey parameter is true, context participants are polled and no action is taken if any participant declines. Otherwise, all context objects are reset to their baseline state. This function allows context participants to prepare for a disconnect action and possibly abort the action if necessary.

#### 2.16.2.2.11 CCOWJoin

Parameter	Datatype	Description
<return value>	Boolean	True if the CSS was successful in joining the common context..

This function instructs the CSS to contact a CCOW-compliant context manager and register itself as a context participant. If no context manager is present, CCOW support has been disabled, or the attempt to join the common context failed, the function returns false.

This function is reserved for use by the VIM.

**2.16.2.2.12 CCOWLeave**

This parameterless procedure causes the CSS to suspend its participation in the CCOW context. If no context manager is present, or the CSS is not an active participant, no action is taken.

This function is reserved for use by the VIM.

**2.16.2.2.13 Connect**

Parameter	Datatype	Description
Server	String	Name of the remote system to be connected. The format is: <username>:<password>@<hostname>:<port> Any portion can be omitted. If the hostname is omitted, either the default host is selected or a list of available hosts is presented, depending on the workstation configuration. If the port is omitted, the host's default RPC port is used. If username and password are omitted, the host requests authentication..
<return value>	Boolean	True if the connection request was successful.

This function connects to the remote server named in *Server*. The return value indicates the success of the request. Note that currently the CSS implements a single, shared instance of the RPC broker. This means that only a single host connection can be active at a given time. This restriction can be relaxed in future versions to allow concurrent connections to multiple hosts and, possibly, multithreaded connections to the same host.

**2.16.2.2.14 ContextChangeBegin**

Context objects use this parameterless procedure to initiate a context change sequence. Consecutive calls to this procedure are nested so that the context change sequence does not actually begin until an equal number of *ContextChangeEnd* procedure calls have been invoked.

**2.16.2.2.15 ContextChangeEnd**

This parameterless procedure decrements the context change reference count and invokes a context change sequence when the reference count reaches zero.

**2.16.2.2.16 Disconnect**

This procedure terminates the connection to the remote server. If no connection is active, the call has no effect.

**2.16.2.2.17 EventFireLocal**

Parameter	Datatype	Description
EventType	String	The name of the event type to fire.
EventStub	String	Additional information specific to the event type

Broadcasts an event of type *EventType* to all subscribers within the application's process space. The effect is identical to an event generated by the host system in that callbacks are made to subscribers via the *ICSS\_SessionEvents* interface. Unlike events generated by the host system, events generated by this call are limited to subscribers within the application's process space.

**2.16.2.2.18 EventFireRemote**

Parameter	Datatype	Description
EventType	String	The name of the event type to fire.
EventStub	String	Additional information specific to the event type
Recipients	String	Optional recipient list. If no recipients are specified, the event is broadcast to all active users on the same host.

Broadcasts an event of type *EventType* to all subscribers connected to the same host. If recipients are specified, distribution is limited to those recipients only. Unlike local events, events generated by this call are sent directly to the host system, which then redirects them to the appropriate recipients. Once an event reaches the recipient, it is further redirected to subscribers to that event within the recipient's application process space through a mechanism identical to local events (see the *ICSS\_SessionEvents* interface).

**2.16.2.2.19 EventHasSubscribers**

Parameter	Datatype	Description
EventType	String	The name of the event for which subscription information is desired.
<return value>	Boolean	Returns true if the specified event type has any local subscribers.

This function can be used to determine if any local subscribers exist for a given event type.

**2.16.2.2.20 EventSubscribe**

Parameter	Datatype	Description
EventType	String	The name of the event for which a subscription is desired.
Callback	ICSS_Session Events	A reference to the interface that will be called when an event of the specified type is received.

This method enters a subscription for the named *EventType*. The caller must specify a reference to a callback interface that will be invoked when an event of the specified type is triggered. See a description of the *ICSS\_SessionEvents* interface for more information. If a subscription already exists for the event type and callback interface, the call has no effect.

**2.16.2.2.21 EventUnsubscribe**

Parameter	Datatype	Description
EventType	String	The name of the event for which a subscription is desired.
Callback	ICSS_Session Events	A reference to the interface that was specified in the original <i>EventSubscribe</i> call.

This method revokes a subscription for the named *EventType*. The caller must specify a reference to the same callback interface that was specified in the original *EventSubscribe* call. Note that subscriptions are automatically revoked when an object is unregistered. If a subscription for the event type and callback interface does not exist, this call has no effect.

**2.16.2.2.22 FindObjectByCLSID**

Parameter	Datatype	Description
CLSID	GUID	The globally unique identifier of the object class to be located..
Last	IUnknown	Interface reference of the previously located interface.
<return value>	IUnknown	Returns a reference to the object implementing the class identified by CLSID or nil if none is found.

This function searches the list of registered objects to find one that implements the class identified by *CLSID*. If the *Last* parameter is not nil, the search begins following that object's entry in the list. In this manner, one can iterate through multiple object instances of the same class.

**2.16.2.2.23 FindObjectByIID**

Parameter	Datatype	Description
IID	GUID	The globally unique identifier of the interface to be located.
Last	IUnknown	Interface reference of the previously located interface.
<return value>	IUnknown	Returns a reference to the object implementing the interface identified by IID or nil if none is found.

This function searches the list of registered objects to find one that implements the interface identified by *IID*. If the *Last* parameter is not nil, the search begins following that object's entry in the list. In this manner, one can iterate through all objects implementing a particular interface.

**2.16.2.2.24 FindObjectByProgID**

Parameter	Datatype	Description
ProgID	String	The programmatic identifier of the object to be located.
Last	IUnknown	Interface reference of the previously located interface.
<return value>	IUnknown	Returns a reference to the object identified by ProgID or nil if none is found.

This function searches the list of registered objects to find one that possesses the programmatic identifier specified by *ProgID*. If the *Last* parameter is not nil, the search begins following that object's entry in the list. In this manner, one can iterate multiple object instances of the same class.

**2.16.2.2.25 FindServiceByCLSID**

Parameter	Datatype	Description
CLSID	GUID	The globally unique identifier of the service's class to be located.
<return value>	IUnknown	Returns a reference to the service implementing the class identified by CLSID or nil if none is found.

Request a reference to the service identified by CLSID. If the service is not already running, the CSS starts the service. If the service is not located, a nil reference is returned. Otherwise, the return value is a reference to the service's default interface.

**2.16.2.2.26 FindServiceByProgID**

Parameter	Datatype	Description
ProgID	String	The programmatic identifier of the service to be located
<return value>	IUnknown	Returns a reference to the object identified by ProgID or nil if none is found.

Request a reference to the service identified by ProgID. If the service is not already running, the CSS starts the service. If the service is not located, a nil reference is returned. Otherwise, the return value is a reference to the service's default interface.

**2.16.2.2.27 RegisterObject**

Parameter	Datatype	Description
ObjRef	IUnknown	IUnknown interface reference of the object to be registered..

The VIM uses this procedure call to register an object with the CSS. The CSS uses the IUnknown interface reference to query the object for the interfaces it supports. When the CSS generates an event for an interface supported by the object, it performs a callback on that interface to communicate the event to the object. In this manner, objects can subscribe to an event by simply implementing the corresponding interface. The *RegisterObject* procedure takes care of connecting the object's event interface (i.e., event sink) to the event source.

This procedure also adds the object reference to a registered object table so that other objects can discover it using one of the Find\* methods.

**2.16.2.2.28 ReplaceParams**

Parameter	Datatype	Description
Source	String	The value to be parsed
<return value>	IUnknown	Returns the input value with all references to replaceable parameters replaced by the corresponding values.

This function parses the input value, replacing references to replaceable parameters with their corresponding values. Replaceable parameters are of the format:

**\$(<parameter>;<format specifier>)**



where the format specifier is optional. The following parameters are recognized:

Parameter	Description
Param.<name>	Where <name> is the name of a parameter created by a call to the SetParam method.
<object>.<prop/meth>,<arg1>...<argn>	<p>Where &lt;object&gt; is the name of an object, &lt;prop/meth&gt; is the name of a property or method within that object, and &lt;arg1&gt;...&lt;argn&gt; is an optional argument list.</p> <p>An object name can either be a programmatic identifier, or a locally named object. Locally named objects include the Session object and all context objects. Context objects provide a local name to the CSS when they are registered. For example, to access a patient's name in the patient context object, use the format \$(PATIENT.NAME).</p> <p>&lt;arg1&gt;...&lt;argn&gt; is an optional list of comma-delimited arguments if required by the method call. Arguments themselves can be replaceable parameters</p>
DEFDIR	The path to the current working directory.
WINDIR	The path to the windows directory.
SYSDIR	The path to the system directory.

This function is especially useful for setting properties of components when the values are not known at design time.

#### 2.16.2.2.29 TraceAdd

Parameter	Datatype	Description
Handle	Integer	Unique handle as returned by the TraceBegin function.
Value	String	Text to add to the trace log entry.
IsHeader	Boolean	If true, text is to be formatted as a header. If false, text is assumed to be body text.

Adds text to the trace log entry.

#### 2.16.2.2.30 TraceBegin

Parameter	Datatype	Description
TraceClas	String	This is the class to which the entry belongs. For example, RPC or EVENT.

Parameter	Datatype	Description
TraceClas	String	This is the type of entry within the class. For example, for the RPC class, the type is the name of the remote procedure.
<return value>	Integer	Returns a unique handle for use in calls to TraceAdd and TraceEnd. If TraceMode is not enabled, always returns 0.

This initiates a trace log entry. TraceMode must be enabled to log entries. The class and type parameters permit classifying a trace log entry so that it can be sorted or filtered in the trace log display.

#### 2.16.2.2.31 TraceEnd

Parameter	Datatype	Description
Handle	Integer	Unique handle as returned by the TraceBegin function.

This closes the trace log entry and fires the TRACE event to all local subscribers.

#### 2.16.2.2.32 UnregisterObject

Parameter	Datatype	Description
ObjRef	IUnknown	IUnknown interface reference of the object to be unregistered.

The VIM calls this procedure when an object is about to be unloaded to instruct the CSS to remove event subscriptions for the object and to remove it from the list of registered objects.

#### 2.16.2.2.33 UnregisterServices

Unregisters each registered service, allowing it to unload.

### 2.16.2.3 ICSS\_SessionEvents

This is the default outgoing interface for the Session automation object and is used to notify components of asynchronous events. A component wishing to be notified of an asynchronous event must implement this interface in its entirety (even if only a subset of its methods is actually needed). The methods defined are:

#### 2.16.2.3.1 EventCallback

Parameter	Datatype	Description
EventType	String	Identifies the type of event that has been signaled..
VaEventStublue	String	Contains data describing details of the event that has been signaled. The format of this parameter is event specific.

The CSS invokes this callback when an event to which an object has subscribed has fired.

### 2.16.2.3.2 RPCCallback

Parameter	Datatype	Description
Handle	Integer	Unique handle of the remote procedure whose results are being returned.
Data	String	The return data of the remote procedure call in CommaText or PlainText format.

The CSS invokes this callback when an asynchronous RPC call has completed. The callback is made to the object that performed the asynchronous call. The *Handle* identifies which RPC is being reported (this is the value returned by the *CallRPCAsync* method of the Session automation object). *Data* represents any data returned by the RPC. If the *CallRPCAsync* method invoked the remote procedure with a *PlainText* parameter value of True, *Data* will be in plain text format (CR-delimited), otherwise it is in comma text format.

### 2.16.2.3.3 RPCCallbackError

Parameter	Datatype	Description
Handle	Integer	Unique handle of the remote procedure generating the error.
ErrorCode	Integer	An error code value returned by the remote procedure.
IsHeader	String	A brief text message describing the error.

When an asynchronous RPC generates an exception, this callback method is called instead of the *RPCCallback* method.

### ICSS\_Context

This interface is defined by the CSS and must be implemented by every context object. The interface properties and methods allow the CSS to interact with the context object and make context change notifications on its behalf.

Parameter	Datatype	Access	Description
Callback	GUID	R	The GUID of the callback interface that will be used to notify subscribers of changes in this context object. Must be a descendant of ICSS_ContextEvents.
ContextName	String	R	The name by which this context will be advertised. This is the name that can be referenced in the ReplaceParams method of the Session object.

Parameter	Datatype	Access	Description
Pending	Boolean	R	If true, the context object has an uncommitted pending context.
Priority	Integer	R	Used to sequence processing of context. Context objects with higher priorities (lower values) are processed before those of lower priorities.

#### 2.16.2.3.4 CommitContext

Parameter	Datatype	Description
Accept	Boolean	If true, the object should commit the pending context. If false, any pending context is cleared.

The CSS invokes this method to instruct a context object to commit or cancel its pending context.

#### 2.16.2.3.5 GetContext

Parameter	Datatype	Description
Pending	Boolean	If true, the context object should return its pending context. Otherwise, the active context is returned.
<return value>	String	The active or pending context in CCOW format.

The CSS uses this method to request context information from the context object in preparation for initiating a CCOW context change. If the object does not produce a CCOW context, it should return a null string.

#### 2.16.2.3.6 Init

The CSS invokes this method to instruct the context object to initialize itself to some default state. It is up to the context object to determine what default state to assume. For example, a patient context object might retrieve the last patient accessed by the current user.

#### 2.16.2.3.7 Reset

The CSS invokes this method to instruct the context object to reset itself to a state that represents no context.

**2.16.2.3.8 SetContext**

Parameter	Datatype	Description
Context	String	Context the object is to assume, in CCOW format.
<return value>	Boolean	If true, the object successfully set its context. If false, the Context parameter contains no context information relevant to this object.

The CSS invokes this method to instruct the context object to initialize its pending context to conform to the context specified in Context. If the object is unable to comply, it should reset its pending context to a null state. If the Context parameter contains no context information relevant to the context object, the object should set its pending context to its default state and return false.

**2.16.2.4 Context Change Events**

Each context object must declare a callback interface that is a descendant of the *ICSS\_ContextEvents* interface defined by the CSS. Though all such callback interfaces implement the identical methods, the GUIDs of each are unique to the context object that declares them. In this way, the CSS is able to notify subscribers of context change events on behalf of the respective context objects (because it declares and, therefore, understands the base interface), but is able to keep the subscriptions distinct. Components wishing to be notified of context changes must implement the callback interface declared by the context object of interest. Components should never implement the *ICSS\_ContextEvents* interface directly, but rather the descendant interface declared by the context object of interest (e.g., the *ICSS\_PatientEvents* interface if the patient context object is the target).

Note that because the method names are the same for all context change event interfaces (because they all have the same ancestor), components implementing more than one context change interface must explicitly map the COM method names to the internal method names that implement them. The technique, called method aliasing, for accomplishing this varies by programming language.

Every context change callback interface declares the following methods:

### 2.16.2.4.1 Pending

Parameter	Datatype	Description
Silent	Boolean	If true, the component should not interact with the user to confirm the context change. This parameter will always be true if the context change request originated from the CCOW context manager.
<return value>	String	If the event subscriber wishes to contest a change in patient context, it should return a non-null string containing a brief description of the reason.

The CSS invokes this function whenever a request has been made to change the selected patient, but **before** the change has taken place. Subscribers wishing to participate in the decision to change the context for the corresponding context object can respond to this event by either prompting the user to save pending changes, warning the user that changes can be lost, and/or contesting the context change by returning a non-null value to the caller. Note, however, that if the Silent parameter is true, only the latter option should be exercised. A component should never request user interaction if the Silent parameter is true.

### 2.16.2.4.2 Committed

The CSS invokes this parameterless procedure after the pending context has been committed. Subscribers can respond by examining the corresponding context object and updating their state accordingly.

### 2.16.2.4.3 Canceled

The CSS invokes this parameterless procedure when a pending context change has been canceled. This can occur when a subscriber contests a pending change.

## 2.16.3 Component Management Service

It is the responsibility of the Component Management Service (CMS) to control the deployment of and access to components defined within the Object Registry. Both the VIM and the CSS utilize the CMS to ensure the availability of other components. Because of this, it is rarely necessary for other components to access the CMS directly.

The CMS provides an object-oriented view of the contents of the Object Repository. In addition, it provides methods for triggering just-in-time deployment of components when necessary. It consists of two automation objects: the registry object and the component object.

### 2.16.3.1 Registry Object (CIA\_CMS.CMS\_Registry)

This automation object embodies the Object Registry and provides access to global settings and individual components.

**2.16.3.1.1 Properties**

Parameter	Datatype	Access	Description
Callback	Boolean	RW	If true, all updates are suppressed. In this state, the CMS will never deploy components. This is generally useful only for debugging purposes..
ContextName	ICMS_Component	R	Array of all components registered within the Object Registry.
Pending	Integer	R	Number of components registered within the Object Registry.

**2.16.3.1.2 FetchObject**

Parameter	Datatype	Description
ProgID	String	The programmatic or class identifier of the component to fetch.
ForceUpdate	Boolean	If false, the component is deployed only if an update is determined to be necessary. If true, the component is always deployed.
<return value>	ICMS_Component	Interface reference to the component automation object

This function ensures that the requested component and any supporting components are deployed from the object repository to the local machine. Deployment of the requested component occurs if one of the following criteria is met:

- The component is not yet installed.
- A newer version resides in the object repository.
- The ForceUpdate parameter is true.

**2.16.3.1.3 FindProgID**

Parameter	Datatype	Description
ProgID	String	The programmatic or class identifier of the component to locate.
ForceUpdate	Boolean	If true, an exception is raised if the requested entry was not found.
<return value>	ICMS_Component	If found, the interface reference to the component automation object. Otherwise, returns null.

This function locates an entry within the Object Registry when given either the programmatic identifier or the class identifier. It returns an interface reference to the corresponding component automation object if found, null if not.

**2.16.3.1.4 Lock**

Parameter	Datatype	Description
Lock	Boolean	If true, the object is locked from any changes. If false, the object can be updated with changes.

This method is used to prevent the object from being refreshed. Each call with a Lock parameter value of true must be matched with a call with a Lock parameter value of false. If a refresh request is received while the object is locked, it will be deferred until the object is unlocked.

**2.16.3.1.5 Refresh**

This method refreshes the contents of the component array from the Object Registry.

**2.16.3.2 Component Object (CIA\_CMS.CMS\_Component)**

The component automation object represents a single entry in the Object Registry.

**2.16.3.2.1 Properties**

Parameter	Datatype	Access	Description
Category[Index]	String	R	Array of all categories to which this component belongs.
CategoryCount	Integer	R	Number of entries in Category array.
CLSID	GUID	R	Class identifier of the component.
Disabled	Enum	R	Possible values are: dtNone (0) = Object is not disabled. dtExplicit (1) = Object has been explicitly disabled in the Object Registry. dtAccess (2) = Object is disabled because requestor lacks a required security key
DotNet	Boolean	R	If true, the object is a .Net component requiring special handling.
EditableProperties	Integer	R	Number of properties that will appear in the generic property editor
Height	Integer	R	Default height for a visual component.
Hidden	Boolean	R	If true, will not appear in the Add Object dialog of the VIM.
Index	Integer	R	The index of the component in the parent collection.
Multiple	Boolean	R	If true, multiple instances of the object can coexist in an application
Name	String	R	Friendly name for the component
NoRegisterCSS	Boolean	R	If true, the object should not be registered with the CSS
ProgID	String	R	Programmatic identifier of the component.



Parameter	Datatype	Access	Description
PropEdit	Boolean	R	If true, the object's internal property editor is used in place of the VIM's generic property editor.
PropInit[Index]	String	R	Array of property initializers.
PropInitCount	Integer	R	Number of entries in PropInit array.
PropSave[Index]	String	R	Number of entries in PropSave array.
Regress	Boolean	R	If true, the component is deployed whenever the local and the repository versions differ, even if the latter is older.
Required[Index]	String	R	Array of all required supporting files.
RequiredCount	Integer	R	Number of entries in Required array.
Service	Boolean	R	If true, the component represents a shared service
SideBySide	Boolean	R	If true, side-by-side versioning is enforced.
Source	String	R	URL to source file in object repository.
Using[Index]	String	R	Array of required components.
UsingCount	Integer	R	Number of entries in Using array.
Version	String	R	Version of the component residing in the Object Repository.
Width	Integer	R	Default width for a visual component.

### 2.16.3.2.2 PropType

Parameter	Datatype	Description
PropName	String	Name of property
<return value>	Enum	The datatype of the name property. One of: ptHidden (0) = Hidden property ptUnknown (1) = Unknown property type ptColor (2) = Color property ptFont (3) = Font property ptBoolean (4) = Boolean property ptImage (5) = Image file property ptFile (6) = File property ptText (7) = Text property ptIcon (8) = Icon property ptEnum (9) = Enumeration property ptFlag (10) = Flag property ptReserved (11) = Reserved for internal use ptInteger (12) = Integer property

This function returns the datatype of the named property.

## 2.16.4 Object Registry

The Object Registry provides information about components that are supported by VueCentric®. Only components that are registered can be accessed and then only when certain criteria are met. The Object Registry is stored in the *VUECENTRIC OBJECT REGISTRY* (#19930.2) file on the host system. See the **File List** section for details on this file.

## 2.16.5 Template Registry

A template is a snapshot of a visual interface in an XML representation. It contains all of the information required to reconstruct a visual interface including state information (like size, alignment, or color) and the parent-child relationships of the visual elements. Templates are used to create varied configurations of the VueCentric® application (user and application templates) and to create “compound objects” that can be dropped into the visual interface as if they were discrete objects (object templates).

Templates are stored in the *VUECENTRIC TEMPLATE REGISTRY* (#19930.3) file. See the **File List** section for details on this file.

When a user makes changes to the visual interface and saves them as a personal (user) configuration, the VIM writes this information to the Template Registry. For user configuration templates, the template name always begins with the ‘\$’ character followed by the user’s unique internal identifier (aka, DUZ). For application level templates, the template name begins with the ‘%’ character. Both user and application templates differ from object templates in that they also contain application level settings (e.g., default font, custom menus) whereas object templates do not.

### 2.16.5.1 Internal Representation

The XML representation of a visual interface embodies two principal kinds of information: the parent/child relationship among objects and the state of those objects. Each object is stored as an XML element with its property values (state information) represented by attribute name/value pairs. The hierarchical relationship among objects is represented by the nesting of elements, with the XML element nodes for child objects being nested within the element nodes of their parent object. In this manner, the visual interface can be reconstructed exactly as it existed at the time the snapshot was taken.

The format of state information varies by the type of associated visual object. Currently, ten internal object types (some of which are compound objects), known as stock objects, are supported:

- object containers (TObjectContainer)
- panels (TPanelEx)
- scroll boxes (TScrollBoxEx)
- labels (TLabelEx)

- page controls (TPageControlEx / TTabSheetEx)
- toolbars (TToolbarEx)
- tree views (TTreeViewEx / TTreeViewPane)
- splitter panes (TSplitterPaneEx / TPaneEx)
- group bars (TGroupBarEx/TGroupPaneEx)
- menu items (TMenuItemEx)

All of these are COM objects with interface declarations imbedded within the VIM type library. Unlike external objects, they are not ActiveX controls but rather specialized descendants of Delphi VCL controls that cannot exist outside the VIM. In contrast, external objects are standard ActiveX controls that are associated with and maintained by the object container (one per container). The object container is responsible for mediating the interaction between the contained ActiveX object and the visual interface.

All stock objects, with the single exception of TMenuItemEx, have a common set of properties. They are:

Parameter	Datatype	Description
ALIGN	Integer	The alignment of the control relative to its parent. Can be one of the following values: 0 = no alignment 1 = top aligned 2 = bottom aligned 3 = left aligned 4 = right aligned 5 = all aligned 6 = centered
ANCHORS	Integer	The anchoring of control boundaries relative to its parent. An anchored boundary maintains a constant distance from the parent boundary, even if the parent resizes. can be the additive combination of the following values: 1 = top 2 = left 4 = right 8 = bottom
BORDER	Integer	The style of border surrounding the control. can be one of the following values: 0 = no border 1 = flat 2 = groove 3 = bump 4 = lowered 5 = button down 6 = raised 7 = button up 8 = status 9 = popup 10 = flat/bold

Parameter	Datatype	Description
CAPTION	String	The caption text associated with the control. Not all controls are capable of displaying caption text
COLOR	Integer	The background color of the control
FONT	IFontDisp	The font of the control. Defaults to the font of the parent control
HEIGHT	Integer	The height of the control in pixels
LEFT	Integer	The position of the leftmost portion of the control in the coordinate system of its parent
LOCK	Integer	If nonzero, the control and all its children can only be modified by a user with compose mode privileg
TOP	Integer	The position of the topmost portion of the control in the coordinate system of its parent
WIDTH	Integer	The width of the control in pixels

In addition to these standard properties, many stock objects have additional properties as detailed below.

#### 2.16.5.2 TObjectContainer

In addition to the properties of the container itself, properties of the contained object can also be saved. These properties can be distinguished from container properties by the presence of an underscore character prefix in the property name. The underscore is not part of the property name, but rather serves to distinguish it from container properties. The object registry determines which properties of the contained object are saved when the container state is saved.

Parameter	Datatype	Description
PROGID	String	The programmatic identifier of the contained ActiveX object

#### 2.16.5.3 TPanelEx

This is an implementation of a panel control upon which other controls can be placed. This control implements only the standard set of properties.

#### 2.16.5.4 TScrollBarEx

Similar to a panel control, this control automatically displays scrollbars if any control placed upon it is outside the current visual boundaries. This control implements only the standard set of properties.

**2.16.5.5 TLabelEx**

This is a simple label that can be used to identify other components in the interface. This control implements only the standard set of properties.

**2.16.5.6 TToolBarEx**

This is a toolbar control that can have multiple controls (usually buttons) placed upon it. It automatically arranges the controls it contains. This control implements only the standard set of properties.

**2.16.5.7 TPageControlEx**

This is a page control that can have multiple tabbed pages (*TTabSheetEx*) on it.

Parameter	Datatype	Description
FIXEDWIDTH	Boolean	If true, all tabs maintain the same width. If false, tab widths are sized to match their caption widths
MULTILINE	Boolean	If true, the page control wraps tabs onto multiple lines if necessary. If false, scroll buttons appear if there are too many tabs to display within the current window boundaries
PAGECOUNT	Integer	The number of tab sheets owned by the control
REVERSE TABS	Boolean	If true, tab order is reversed
TOPPAGE	Integer	The index of the tab sheet which is initially on top.
TABPOSITION	Integer	The location of tabs on the page control. One of the following values: 0 = top 1 = bottom 2 = left 3 = right
TABSTYLE	Integer	The style of tabs. One of the following values: 0 = single slant 1 = double slant 2 = cut corner 3 = round corne

**2.16.5.8 TTabSheetEx**

These are the tab sheets that can appear on a page control.

Parameter	Datatype	Description
PAGEINDEX	Integer	Order in which tab sheet appears on the parent control

### 2.16.5.9 TSplitterPaneEx

This is a component with multiple panes separated by splitter bars that can be manually resized.

Parameter	Datatype	Description
HOTSPOTVISIBLE	Boolean	If true, a hotspot appears on the splitter that allows closing and opening of the adjacent panes
ORIENT	Integer	The orientation of panes within the control. One of: 0 = horizontal 1 = vertical
PANECOUNT	Integer	The number of panes displayed by the control

### 2.16.5.10 TPaneEx

These are the individual panes that comprise a splitter pane control.

Parameter	Datatype	Description
PAGEINDEX	Integer	The relative position of the pane within the parent control

### 2.16.5.11 TTreeViewEx

This is a component with a tree view on one side and a pane view on the other. Each node of the tree has an associated pane that becomes visible in the pane view when the node is selected.

Parameter	Datatype	Description
DEFAULT	String	The path of the node whose pane is to appear when the control is initially loaded
ICONS	String	Specifies a file containing icon resources that are to be used by the control.
LARGEICONS	Boolean	If true, the control displays large icons (32x32). If false, small icons (16x16)
ORIENT	Integer	The position of the tree view within the control. One of: 0 = left 1 = right
SPLITTER	Integer	The position of the vertical splitter relative to the left border

### 2.16.5.12 TTreePaneEx

These are the individual panes that comprise a splitter pane control.

Parameter	Datatype	Description
ICON	Integer	The index of the icon to be displayed

Parameter	Datatype	Description
PATH	String	The path of the node. A path consists of the node caption preceded by the captions of each of its ancestor nodes, separated by backslashes
VISIBLE	Boolean	If true, the node is initially visible

### 2.16.5.13 TGroupBarEx

This component displays a group bar on one side and a pane on the other. The group bar displays a list of items organized into groups. Each item has an associated pane that becomes visible in the pane view when that item is selected.

Parameter	Datatype	Description
CAPTIONCOLOR	Integer	The color of the pane view's caption bar.
DEFAULT	String	The path of the group item that is selected by default when the component is initially loaded. The path consists of the group name followed by the item name, separated by a backslash
GROUPICONS	String	The list of icon indexes used by the respective groups. This is a series of integer values separated by semicolons
GROUPSTATES	String	The list of group state values for each of the respective groups. This is a series of Boolean values separated by semicolons. A value of 1 indicates the group is initially expanded. 0 indicates the group is initially collapsed
ICONS	String	Specifies a file containing icon resources that are to be used by the control.
LARGEICONS	Boolean	If true, the control displays large icons (32x32). If false, small icons (16x16)
ORIENT	Integer	The location of the pane view. One of: 0 = left 1 = right
SPLITTER	Integer	Initial position of the splitter separating the group bar from the pane view
STYLE	Integer	The style of the group bar. One of: 0 = category view 1 = task list 2 = Outlook styl

### 2.16.5.14 TGroupPaneEx

These are the individual panes within the TGroupBarEx component.

Parameter	Datatype	Description
ICON	Integer	The index of the icon displayed next to the group item associated with this pane
PATH	String	The path of the group item associated with this pane

### 2.16.5.15 TMenuItemEx

These are custom menu items that are added to the application's main menu. Unlike other stock objects, TMenuItemEx does not implement the standard set of properties. Its properties are:

Parameter	Datatype	Description
ICON	Integer	The index of a custom icon that is to displayed next to the menu item.
INDEX	Integer	The position of the menu item relative to its siblings.
LOCK	Integer	If nonzero, the control and all its children can only be modified by a user with compose mode privilege
PATH	String	The full path of the menu item, including its parent menus. This consists of the captions of the menu item and all its parent menus separated by backslash characters
ACTION	String	The action to be taken when the menu item is clicked.

### 2.16.5.16 XML Representation

The following is a sample of a saved object template named "HEADER". The XML tags (elements) have been indented and bolded to help illustrate the parent-child relationships. This example shows a template consisting of a single splitter pane control at the top level with three child panes, each with a single child object upon them. The attributes associated with each XML element represent the property values of the associated object at the time the snapshot was taken. Attribute names beginning with an underscore represent additional properties of the associated object that have been serialized (i.e., they are properties of the contained object, not the container



itself).

```
<Template NAME="HEADER" VERSION="1.1.0.79" HEIGHT="768" WIDTH="1024">
  <TSplitterPaneEx TAG="0" LEFT="0" TOP="0" HEIGHT="48" WIDTH="1016"
    ALIGN="1" PANECOUNT="3" ORIENT="0" BORDER="0">
    <TPaneEx HEIGHT="46" WIDTH="42" COLOR="-2147483633" PANEINDEX="0" TAG="0">
      <TObjectContainer TAG="0" LEFT="1" TOP="1" HEIGHT="44" WIDTH="40"
        ALIGN="5" PROGID="VCPATPHOTO.VCPATPHOTOX"/>
    </TPaneEx>
    <TPaneEx HEIGHT="46" WIDTH="200" COLOR="-2147483633" PANEINDEX="1" TAG="0">
      <TObjectContainer TAG="0" LEFT="1" TOP="1" HEIGHT="44" WIDTH="198"
        ALIGN="5" PROGID="VCPATIENTID.VCPATIENTIDX" _COLOR="15780518"/>
    </TPaneEx>
    <TPaneEx HEIGHT="46" WIDTH="200" COLOR="-2147483633" PANEINDEX="2" TAG="0">
      <TObjectContainer TAG="0" LEFT="1" TOP="1" HEIGHT="44" WIDTH="198"
        ALIGN="5" PROGID="VCENCOUNTERINFO.VCENCOUNTERINFOX" _COLOR="8454143"/>
    </TPaneEx>
  </TSplitterPaneEx>
```

When reconstructing a saved configuration, the VIM performs a depth-first traversal of the XML document tree, instantiating the visual elements described by each XML node as it goes. The parent-child relationships represented in the document tree are reproduced as parent-child relationships in the visual interface. When fully instantiated in the VIM, the example above would appear something like this:



## 2.16.6 Object Repository

The Object Repository provides a centralized location for storing the most up-to-date versions of VueCentric<sup>®</sup> components. The Object Repository can be implemented on a web server, an FTP server, a shared network directory, or any combination of these. The Object Repository works in concert with the Object Registry to permit the automatic updating of components. The Object Registry provides information about the components stored in the Object Repository including version information and a URL to be used to locate an updated version of a component.

Typically a site will implement its Object Repository in one of the three locations mentioned. However, it is entirely possible that a site can implement components that are developed and maintained by another site. In such a scenario, it would be logical to retrieve updates to such a component directly from the originating site, typically using the FTP or HTTP protocol. However, the implementing site must still update its Object Registry to indicate when a new version is available.

When a component is requested by the VIM, which can occur in design mode when a component is dropped into the visual interface or during the loading of a saved configuration, the VIM checks the locally installed version of the component with the version available from the Object Repository. If the Object Repository has a newer version, or if the component has not yet been installed on the local machine, the VIM downloads a copy of the component from the Object Repository (using the URL specified in that component's Object Registry entry). It then automatically registers the updated component before instantiating it within the interface. Other than a slightly perceptible delay while the download occurs, this process is essentially transparent to the user and occurs without direct intervention.

## 3.0 Remote Monitoring Service

### 3.1 Introduction

The Remote Monitoring Service provides support for the remote monitoring feature of the VueCentric System Management Utility.

### 3.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	VCMONITOR.MONITOR
Version	1.1.0.6
Class Identifier	{C3D57A62-DF34-44DF-851C-61DF27F7B8EC}
Image File	vcMonitor.dll
Property Initializations	none
Serializable Properties	none
Required Files	none
Security Keys	none
Multiple Instances Allowed	no
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	yes
.Net Component	no
Associated Build	VUECENTRIC FRAMEWORK 1.1V2

To ensure that the Remote Monitoring Service is always started, it is recommended that it be flagged as a dependency for the Session object. The Session object (CIA\_CSS.CSS\_SESSION) is a framework component that is not normally displayed in the VueCentric System Management Utility unless the Framework Objects checkbox is checked under the list restrictions (see below).

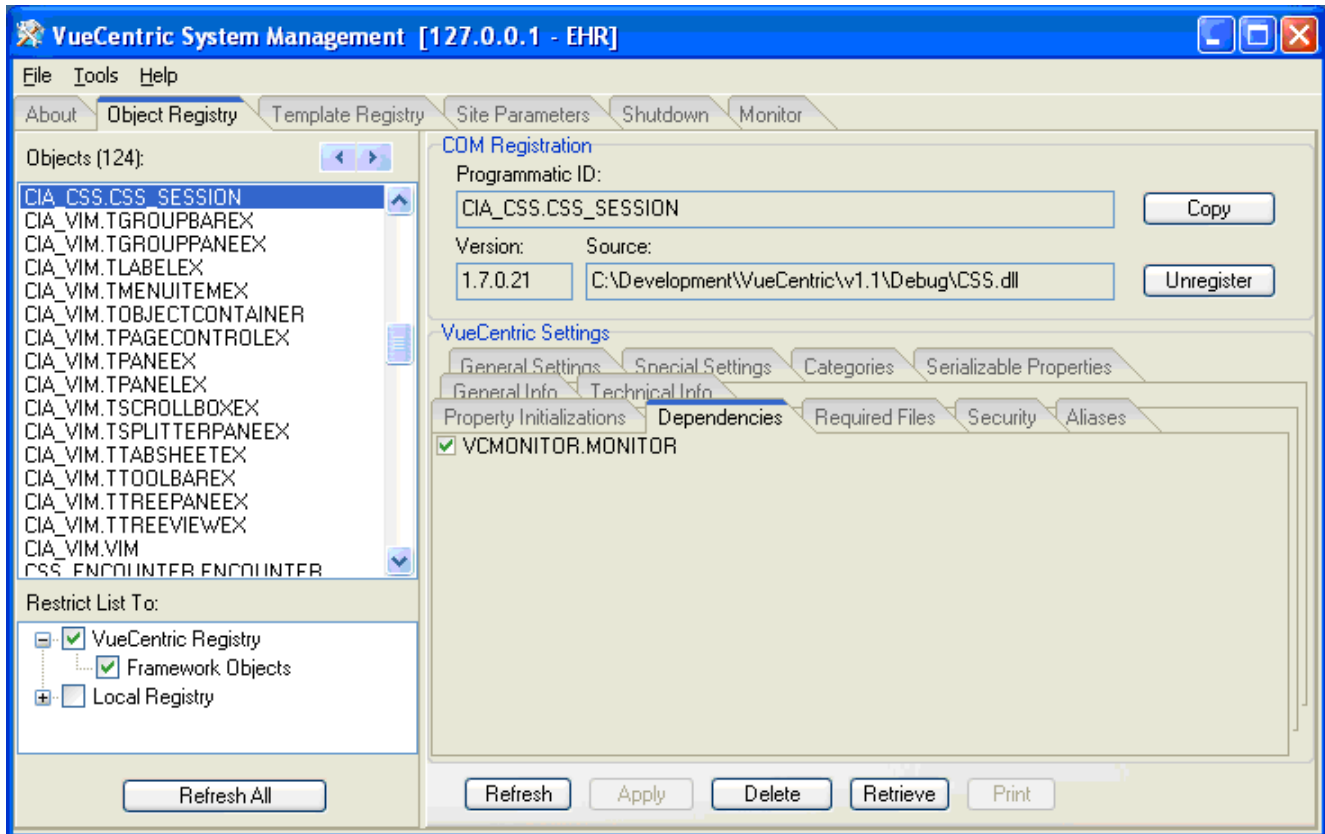


Figure 3-1: Dependencies

Select the CIA\_CSS.CSS\_SESSION object from the object pane and check the VCMONITOR.MONITOR object on the Dependencies tab (right-click on the dependencies tab and select Show All to see all available objects). This will ensure that the Remote Monitoring Service is started whenever a session starts.

### 3.3 Routine Descriptions

None.

### 3.4 File List

None.

### 3.5 Cross References

None.

### 3.6 Exported Options

None.

### 3.7 Exported Security Keys

None.

### 3.8 Exported Protocols

None.

### 3.9 Exported Parameters

None.

### 3.10 Exported Mail Groups

None.

### 3.11 Callable Routines

None.

### 3.12 External Relations

None.

### 3.13 Internal Relations

None.

### 3.14 Archiving and Purging

There are no archiving or purging requirements within this software.

### 3.15 Components

This component supports the following properties and methods:

#### 3.15.1 Properties

Parameter	Datatype	Access	Description
RemoteManager	Integer	R	Session ID of the remote manager.

### 3.15.2 GetData

Scope: private.

Parameter	Datatype	Description
Group	Enum	Information group to retrieve. One of: 0 = Active event subscriptions 1 = Local variables 2 = Session variables 3 = Local operating system 4 = Local hardware 5 = Local memory usage 6 = Local network settings 7 = Environment variable settings 8 = Local folder assignments 9 = Loaded modules 10 = Running processes 11 = Registered record locks 12 = Pending asynchronous RPC's
<return value>	String	Requested information

Retrieves data on the specified group of system parameters.

## 4.0 Date Service

### 4.1 Introduction

The Date Service provides access to methods and dialogs for common tasks related to handling of dates, both FileMan and local system formats.

### 4.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	VCDATE.VCDATES
Version	1.0.1.4
Class Identifier	{33D515AC-4062-46F1-8E97-3871BA1C4289}
Image File	vcDate.dll
Property Initializations	
Serializable Properties	
Required Files	
Security Keys	none
Multiple Instances Allowed	no
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	yes
.Net Component	no
Associated Build	VUECENTRIC FRAMEWORK 1.1V2

There are no specific implementation or maintenance tasks associated with this component.

### 4.3 Routine Descriptions

None.

### 4.4 File List

None.

### 4.5 Cross References

None.

### 4.6 Exported Options

None.

## 4.7 Exported Security Keys

None.

## 4.8 Exported Protocols

None.

## 4.9 Exported Parameters

None.

## 4.10 Exported Mail Groups

None.

## 4.11 Callable Routines

None.

## 4.12 External Relations

None.

## 4.13 Internal Relations

None.

## 4.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 4.15 Components

This component supports the following methods:

### 4.15.1 DateRange

Scope: public.



Parameter	Datatype	Description
DateRec	Structure	Has the following fields: Date1: DateTime Date2: DateTime Date1FM: String Date2FM: String UseFMDateStr: Boolean DateOnly: Boolean RequireTime: Boolean Instruction: String Date1Label: String Date2Label: String
DateLogic	Enum	One of: 0 = None 1 = Start date must be <= end date 2 = End date must be <= start date
<return value>	Boolean	False if dialog was cancelled.

Invokes the date range dialog, allowing the user to input a date range.

#### 4.15.2 DateSelect

Scope: public.

Parameter	Datatype	Description
Date	Date/Time	Date passed and returned.
DateOnly	Boolean	If true, time component is ignored.
ReqTime	Boolean	If true, a time is required.
<return value>	Boolean	False if dialog was cancelled.

Invokes the date selection dialog, allowing the user to input a single date.

#### 4.15.3 DateToFMDate

Scope: public.

Parameter	Datatype	Description
Value	Date/Time	Date to convert.
<return value>	Double FP	FM date equivalent.

Converts a system date/time to a FileMan date/time.

#### 4.15.4 DateToFMDateStr

Scope: public.

Parameter	Datatype	Description
Value	Date/Time	Date to convert.
<return value>	String	FM date equivalent as a string.

Converts a system data/time to a FileMan date/time as a string.

#### 4.15.5 DefaultDateFormat

Scope: public.

Parameter	Datatype	Description
Value	Date/Time	Date to convert.
<return value>	String	Formatted date.

Returns the date in the default display format.

#### 4.15.6 FMDateStrToDate

Scope: public.

Parameter	Datatype	Description
Value	String	FM date as a string.
<return value>	Date/Time	Converted date.

Converts a FileMan date string to local date/time format.

#### 4.15.7 FMDateStrToFMDate

Scope: public.

Parameter	Datatype	Description
Value	String	FM date as a string.
<return value>	Double FP	FM date equivalent as a real number.

Converts a FileMan date string to its real number equivalent.

#### 4.15.8 FMDateToDate

Scope: public.

Parameter	Datatype	Description
Value	Double FP	FM date.
<return value>	Date/Time	Date equivalent in local format.

Converts a FileMan date to local date format.

#### 4.15.9 FMDateToFMDateStr

Scope: public.

Parameter	Datatype	Description
Value	Double FP	FileMan date.
<return value>	String	FileMan date as a string.

Converts a FileMan date to its string equivalent.

#### 4.15.10 FormatAge

Scope: public.

Parameter	Datatype	Description
DOB	Date/Time	Date of birth in local date/time format.
<return value>	String	Age based on today's date and formatted as a string.

Computes an age based on today's date and returns it as a string complete with units.

#### 4.15.11 HL7DateToDate

Scope: public.

Parameter	Datatype	Description
HL7Date	String	Date in HL7 format.
<return value>	Date/Time	Converted date in local date/time format.

Converts an HL7-format date/time to local date/time format.

#### 4.15.12 HODateToDate

Scope: public.

Parameter	Datatype	Description
HODate	String	Date/time in M (\$HOROLOG) format.
<return value>	Date/Time	Converted date in local date/time format.

Converts a date in M (\$HOROLOG) format to local date/time format.

## 5.0 Print Service

### 5.1 Introduction

The Date Service provides access to methods and dialogs for common tasks related to handling of dates. Multiple date formats are supported (Windows, FileMan, HL7, \$HOROLOG).

### 5.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	VCPRINT.VCPRINTX
Version	1.1.0.45
Class Identifier	{114589C7-2335-46BA-8AD3-8A835D92358A}
Image File	vcPrint.dll
Property Initializations	
Serializable Properties	
Required Files	Interop.vcPrint.dll;1.3.0.0
Security Keys	none
Multiple Instances Allowed	no
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	yes
.Net Component	no
Associated Build	VUECENTRIC FRAMEWORK 1.1V2

There are no specific implementation or maintenance tasks associated with this component.

### 5.3 Routine Descriptions

Routine	Description
CIAVUTIO	Device I/O support.

### 5.4 File List

None.

### 5.5 Cross References

None.

### 5.6 Exported Options

None.

## 5.7 Exported Security Keys

None.

## 5.8 Exported Protocols

None.

## 5.9 Exported Parameters

Parameter	Instance Type	Value Type	Precedence	Description
CIAVUTIO DEFAULT FOOTER	String	Word processing	User, Service, Division, System	The instance specifies the name of a report type while the value contains the footer text. Text can contain replaceable parameters.
CIAVUTIO DEFAULT HEADER	String	Word processing	User, Service, Division, System	The instance specifies the name of a report type while the value contains the header text. Text can contain replaceable parameters.
CIAVUTIO DEFAULT PRINTER		String	User, Location	Specifies the default printer for user or location.
CIAVUTIO LOCAL PRINTER		Boolean	User, Location, System	<p>If set to yes, the EHR will display the Windows standard printer selection dialog instead of the RPMS/VistA printer selection dialog.</p> <p>If set to no, the standard RPMS/VistA printer selection dialog will be displayed, still allowing selection of a Windows printer, but requiring an additional prompt.</p>

## 5.10 Exported Mail Groups

None.

## 5.11 Callable Routines

### 5.11.1 OUTPUT^CIAVUTIO

Scope: public.

Parameter	Datatype	Description
Execute	String	M code to be executed for report generation.
Global Root	String	Closed global reference where report output is to be stored.
Right Margin	Integer	Right margin setting to use for output.

Invokes CAPTURE^CIAUHFS to redirect report output to a global.

### 5.11.2 RPC: CIAVUTIO PRINT

Scope: public.

Parameter	Datatype	Description
Handle	Integer	Unique handle for identifying this print request. Pass a value of 0 on the initial call and use the value returned for subsequent calls.
Text	String List	Contains a block of report text.
Output Device	Numeric	Internal entry number of output device (pass on final call only) or any negative value to abort the print request. Pass a value of 0 for all other cases.
Title	String	Title of report (optional).
Line Break Marker	String	Optional marker that indicates where a mandatory line break should occur.
Indent	Integer	Indicates number of characters to indent output.
<return value>	Integer	Unique handle assigned to this report or, if this is the final call, the identifier of the tasked job.

This remote procedure call is used to print report text to an RPMS/VistA printer. Since report text can be large, it permits the submission of a print request across multiple calls, passing partial blocks of report text with each call. This permits the application to allow the user to abort a lengthy print request while it is in progress. To identify a specific print request across multiple calls, a unique handle is assigned on the initial call that should be used in all subsequent calls for the same print request. The value of the Output Device parameter indicates to the server whether this is the final call (when a positive integer value is specified indicating the internal entry number of the device to which the report is directed), the print request is to be aborted (when any negative value is passed), or additional calls can be expected (when a value of zero is passed). Only when the final call is issued is the report queued for printing to the specified output device.

### 5.11.3 RPC: CIAVUTIO PRTGETDF

Scope: private.

Parameter	Datatype	Description
Location	Pointer (#44)	Optional IEN of a hospital location (for location-specific printer assignments).
<return value>	String	The currently assigned default printer for the current user.

Returns the default printer setting from the CIAVUTIO DEFAULT PRINTER parameter.

### 5.11.4 RPC: CIAVUTIO PRTSETDF

Scope: private.

Parameter	Datatype	Description
Device	String	Specifier for the default printer device.
<return value>	String	The return value from the EN^XPAR call.

Sets the default printer setting for the current user in the CIAVUTIO DEFAULT PRINTER parameter.

### 5.11.5 RPC: CIAVUTIO DEVICE

Scope: public.

Parameter	Datatype	Description
From	String	Starting point for device search.
Direction	Integer	1=forward search; -1=backward search
Maximum	Integer	Maximum number of entries to return. Defaults to 20.
<return value>	String List	Returns a list of entries from the DEVICE file in the format: IEN;Name^Display Name^Location^Right Margin^Page Length

Returns a list of entries from the device file.

## 5.12 External Relations

None.

## 5.13 Internal Relations

None.

## 5.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 5.15 Components

This component supports the following properties and methods:

### 5.15.1 Properties

Property	Datatype	Access	Description
ActivePrinter	String	R	Returns the identifier of the currently selected printer, or null if no printer has been selected.
DefaultPrinter	String	R	Returns the identifier of the default printer for the currently location context (or global default if no location has been selected).
DefaultHeader	String	R	Returns the raw text for the default header for the specified category, or null if none exists.
DefaultFooter	String	R	Returns raw text for the default footer for the specified category, or null if none exists.

### 5.15.2 ClosePreview

Scope: public.

Parameter	Datatype	Description
Handle	Integer	Unique preview handle.

Closes the preview dialog identified by the specified handle.

### 5.15.3 FindPreview

Scope: public.

Parameter	Datatype	Description
Handle	Integer	Unique preview handle.
BringToFront	Boolean	If true and preview dialog is found, it will be brought to the top of the window Z-order.
<return value>	Boolean	True if the specified preview was found.

Verifies that the specified preview dialog exists and, optionally, brings it to the top of the window Z-order.

### 5.15.4 Format

Scope: public.



Parameter	Datatype	Description
Text	String	This is the raw text of the report to be formatted.
Title	String	This is optional title text which can be displayed within a header or footer.
Header	String	This can be null (if no header is desired), text (with optional imbedded macros), or an '@' character followed by a header category name (to use a standardized header).
Footer	String	Same format as the Header parameter.
PageBreak	String	Text to be used as a placeholder for page breaks. When the report is actually printed, this placeholder is replaced by device-specific control codes to generate a new page. If this value is null, no page breaks are inserted.
PageWidth	Integer	This is the maximum page width. Currently, lines are not wrapped and this value is only used to determine width of the underline. If less than or equal to zero, the active printer's default page width is used.
PageHeight	Integer	This is the maximum page length. It is used to determine placement of page breaks and footer text. If zero, it defaults to active printer's page length. If less than zero, it suppresses page breaks altogether.
<return value>	String	The fully formatted text including headers, footers, and page breaks.

Transforms raw text into formatted text, complete with headers, footers, and page breaks.

### 5.15.5 Preview

Scope: public.

Parameter	Datatype	Description
Text	String	This is the raw text of the report to be formatted.
Title	String	This is optional title text which can be displayed within a header or footer.
Header	String	This can be null (if no header is desired), text (with optional imbedded macros), or an '@' character followed by a header category name (to use a standardized header).
Footer	String	Same format as the Header parameter.
PageBreak	String	Text to be used as a placeholder for page breaks. When the report is actually printed, this placeholder is replaced by device-specific control codes to generate a new page. If this value is null, no page breaks are inserted.
PageWidth	Integer	This is the maximum page width. Currently, lines are not wrapped and this value is only used to determine width of the underline. If less than or equal to zero, the active printer's default page width is used.
AllowPrint	Boolean	If true, a Print button appears on the Preview Dialog.
Modal	Boolean	If true, the Preview Dialog is modal. If false, the dialog is amodal.

Generates a preview of the fully formatted report. The Preview Dialog can be modal or nonmodal and optionally allow the report to be printed.

### 5.15.6 Preview2

Scope: public.

Parameter	Datatype	Description
Text	String	This is the raw text of the report to be formatted.
Title	String	This is optional title text which can be displayed within a header or footer.
Header	String	This can be null (if no header is desired), text (with optional imbedded macros), or an '@' character followed by a header category name (to use a standardized header).
Footer	String	Same format as the Header parameter.
PageBreak	String	Text to be used as a placeholder for page breaks. When the report is actually printed, this placeholder is replaced by device-specific control codes to generate a new page. If this value is null, no page breaks are inserted.
PageWidth	Integer	This is the maximum page width. Currently, lines are not wrapped and this value is only used to determine width of the underline. If less than or equal to zero, the active printer's default page width is used.
AllowPrint	Boolean	If true, a Print button appears on the Preview Dialog.
Modal	Boolean	If true, the Preview Dialog is modal. If false, the dialog is amodal.
<return value>	Integer	Unique handle associated with the Preview Dialog.

Generates a preview of the fully formatted report. The Preview Dialog can be modal or nonmodal and optionally allow the report to be printed. This differs from the Preview method only in that it returns a handle to the newly created Preview Dialog.

### 5.15.7 Print

Scope: public.

Parameter	Datatype	Description
Text	String	This is the raw text of the report to be formatted.
Title	String	This is optional title text which can be displayed within a header or footer.
Header	String	This can be null (if no header is desired), text (with optional imbedded macros), or an '@' character followed by a header category name (to use a standardized header).
Footer	String	Same format as the Header parameter.
PageBreak	String	Text to be used as a placeholder for page breaks. When the report is actually printed, this placeholder is replaced by device-specific control codes to generate a new page. If this value is null, no page breaks are inserted.

Sends a report to the selected output device. This procedure first displays the Printer Selection Dialog to allow the user to choose the desired output device.

### 5.15.8 Print2

Scope: public.

Parameter	Datatype	Description
Text	String	This is the raw text of the report to be formatted.
Title	String	This is optional title text which can be displayed within a header or footer.
Header	String	This can be null (if no header is desired), text (with optional imbedded macros), or an '@' character followed by a header category name (to use a standardized header).
Footer	String	Same format as the Header parameter.
PageBreak	String	Text to be used as a placeholder for page breaks. When the report is actually printed, this placeholder is replaced by device-specific control codes to generate a new page. If this value is null, no page breaks are inserted.
PrinterTypes	Integer	Determines whether selectable printers are server-based, client-based, or both. Can be one of the following: 1 = Client-based 2 = Server-based 3 = Both
<return value>	Boolean	True if the print operation was completed.

Sends a report to the selected output device. This function first displays the Print Selection Dialog to allow the user to choose the desired output device. It differs from the Print procedure in that one can constrain the type of selectable printers and in that it returns an indication of whether or not the print operation was completed.

### 5.15.9 Reset

Scope: public.

Closes all open preview dialogs.

### 5.15.10 SelectPrinter

Scope: public.

Parameter	Datatype	Description
PrinterTypes	Integer	Determines whether selectable printers are server-based, client-based, or both. Cany be one of the following: 1 = Client-based 2 = Server-based 3 = Both

Invokes the Printer Selection Dialog and returns the identifier of the selected device. The act of selecting a device also sets the ActivePrinter property to that device.

### 5.15.11 UpdatePreview

Scope: public.

Parameter	Datatype	Description
Handle	Integer	Unique preview handle.
Text	String	Report text.
BringToFront	Boolean	If true and preview dialog is found, it will be brought to the top of the window Z-order.
<return value>	Boolean	Returns true if the operation succeeded.

Updates the contents of the specified Preview Dialog.

## 6.0 Remote Procedure Call (RPC) Broker

### 6.1 Introduction

The Medsphere RPC Broker mediates data exchange between a Windows-based client application and an M-based host system. In support of this data exchange, the Broker provides the following services:

- User authentication via one of three methods
- Data access via remote procedure calls
- Access control via security contexts
- Asynchronous messaging
- Event subscription/propagation
- State variable management

The Broker client is packaged as a standard Delphi VCL component and as an automatic-compatible COM object. This permits its use with a wide range of commercial development tools. The Broker is also imbedded within the VueCentric<sup>®</sup> Framework Communication Service and is the primary means of communicating with the RPMS application.

### 6.2 Architecture

The Medsphere RPC Broker performs data exchange via a standard TCP connection between a client application running under a Windows-based operating system and a server application running on one of several supported M platforms (Caché, DSM, and MSM).

The client application initiates the Broker connection by opening a predetermined port on the target host system. This establishes a brief dialog with a primary listener daemon during which the client and host negotiation connection and authentication strategies. In previous versions (see Figure 6-1), the client directed the host to perform a callback connection to a specified port on the client machine. The primary listener daemon spawns a secondary listener process that initiates a TCP connection back to the specified client port. Starting with version 1.1, this connection strategy is only used when the broker is in debug mode.

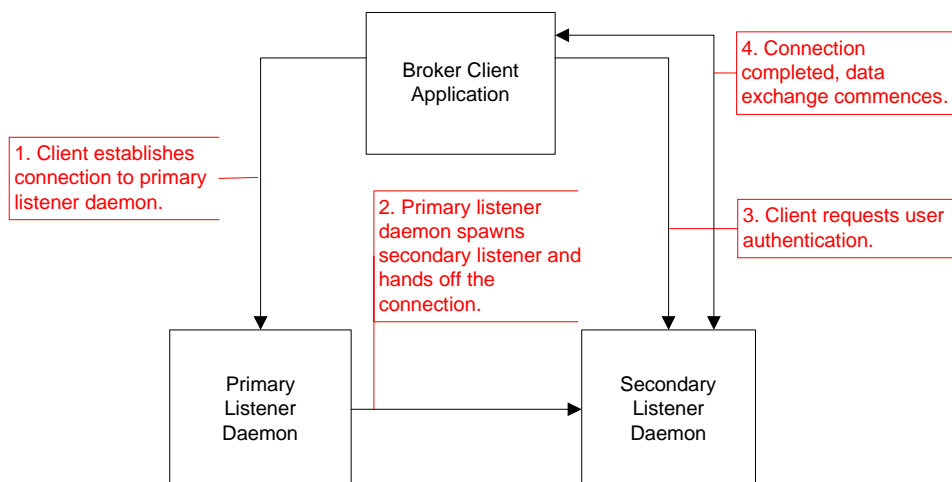


Figure 6-1: Previous Version Architecture

Because of problems initiating callbacks in some network environments (notably, some virtual private network configurations and networks where network address translation is in effect), version 1.1 of the CIA Broker eliminates the need to do callbacks to establish the working connection. Under this connection strategy (see Figure 6-2), the primary listener hands off the TCP connection directly to the secondary listener. This connection strategy is more efficient and robust than is the use of callbacks.

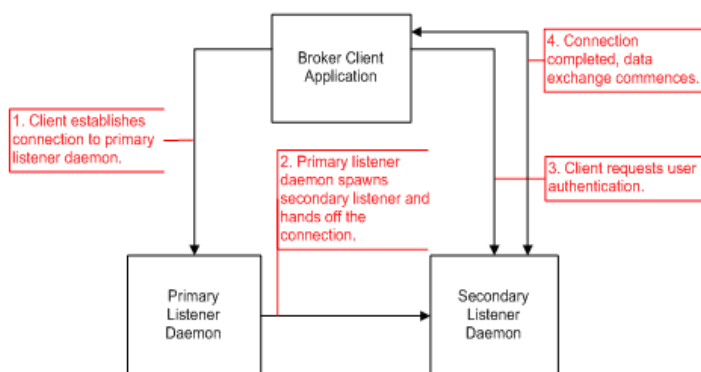


Figure 6-2: Version 1.1 Architecture

Regardless of the connection strategy, once the final connection has been established, the user is authenticated through one of three mechanisms and, if authentication is successful, the connection is complete and ready for use.

## 6.3 Implementation and Maintenance

Information relating to the implementation and maintenance of this package can be found in the Medsphere Broker Installation Guide.

## 6.4 Routine Descriptions

This package has been assigned the namespace designation of “CIANB”. The following routines are distributed in binary form only:

Routine	Description
CIANBACT	Broker protocol actions.
CIANBASY	Asynchronous remote procedure support.
CIANBEVT	Event management.
CIANBINI	KIDS installation support.
CIANBLIS	Primary and secondary listener daemons.
CIANBRPC	Remote procedure calls for basic broker operations.
CIANBUTL	Miscellaneous utility functions.

## 6.5 File List

This package has been assigned the file number range of 19941.2 through 19941.2999. The following files are distributed:

### 6.5.1 CIA AUTHENTICATION File (#19941.2)

The *server-cached* authentication method (see the Medsphere Broker Installation Guide for more details on authentication options) uses this file to store the Windows to RPMS bindings. This file has the following fields:

Field Name	#	Datatype	Indexes	Description
SID	.01	Text	B – Standard	This is the user security identifier.
USER	1	Pointer (#200)		This is a pointer to file 200 that indicates the user associated with this security identifier. If this field is null, the associated security identifier is excluded from auto-authentication.
CREATED	2	Date		This is the date and time that the entry was created.

This file is automatically populated when a user authenticates for the first time. By removing an entry, one can force a given user to re-authenticate. By deleting the USER field value, one can prevent the associated Windows user from ever auto-authenticating. This latter technique is useful for preventing auto-authentication for generic user accounts.

### 6.5.2 CIA EVENT LOG File (#19941.23)

This file provides a location for logging selected event activity. An event is logged here if the LOG EVENT field in the *CIA EVENT TYPE* file is set to YES. The file has the following fields:

Field Name	#	Datatype	Indexes	Description
TIMESTAMP	.01	Date/Time	B – Standard	This is the date and time the event was logged.
EVENT NAME	1	Text	C – Standard	This is the event type name.

Field Name	#	Datatype	Indexes	Description
USER	2	Pointer	D – Standard	This is the user whose session generated the event. Points to file 200.
SESSION	3	Integer	E – Standard	This is the identifier of the session that generated the event.
EVENT STUB	10	Word processing		This is the data associated with the event.

Retention of entries in this file is determined by the LOG RETENTION field in the *CIA EVENT TYPE* file.

### 6.5.3 CIA EVENT TYPE File (#19941.21)

This file provides a means to register an event and to control who is able to subscribe or publish it. The file has the following fields:

Field Name	#	Datatype	Indexes	Description
NAME	.01	Text	B – Standard	This is the name of the event type.
MONITOR	1	M code		The M code to execute to determine if an event of this type needs to be signaled. This is not required for events that are signaled by other means.
DISABLE	2	Boolean		If true, the event is disabled and attempts to signal the event will be ignored.
INTERVAL	3	Integer		The optimal polling interval for the event, in seconds. This applies only to events that have an associated event monitor. The actual polling interval depends on the client polling interval but will never be more frequent than what is specified here.
LOG EVENT	4	Boolean		If true, each occurrence of this event will be logged.
LOG RETENTION	5	Integer		Number of days that log entries for this event type are to be retained.
POLLING METHOD	6	Set		For monitored events, this setting determines how the event monitor is invoked during each polling interval. A value of 0 causes the event monitor to be executed once for each subscribing session and in that session's context. A value of 1 causes the event monitor to be executed once each polling interval and without session context information.
DISPLAY LOGIC	10	M code		Logic to display the event stub data in the log viewer. This feature is not currently implemented.



Field Name	#	Datatype	Indexes	Description
PUBLICATION KEY	20	Pointer (multiple)	B - Standard	If specified, the user must have at least one of the security keys to signal (publish) the event.
SUBSCRIPTION KEY	21	Pointer (multiple)	B - Standard	If specified, the user must have at least one of the security keys to subscribe to the event.
DESCRIPTION	99	Word Processing		This should be a detailed description of the event.
ERROR DATE/TIME	100	Date		The date and time the last error was logged.
ERROR TEXT	101	Text		The text of the last encountered error.

:While it is technically not necessary to create an entry in this file in order to use an event, it is strongly encouraged to do so for two reasons. First, this file provides a place to document each event and helps avoid naming collisions between events. Second, this file permits applying business rules that can restrict publication and subscription rights.

#### 6.5.4 CIA LISTENER File (#19941.22)

This file provides a means to autostart one or more primary listener daemons. It has the following fields:

Field Name	#	Datatype	Indexes	Description
NAME	.01	Text	B – Standard	This is a brief descriptive name to identify the listener.
PORT	1	Integer		This is the TCP port that the listener will monitor for connections.
UCI	1.5	Text		This is the UCI under which the listener will run.
DISABLE	2	Boolean		If true, suppresses the autostart feature for this listener.

This file is only used by Caché installations since other platforms have alternate means for starting the primary listener daemons.

## 6.6 Cross References

Cross references are described in the preceding section.

## 6.7 Exported Options

The descriptions of the options are shown in the following table.

Option	Type	Description
CIANB MAIN MENU	Menu	This is the main menu for broker management tasks. It contains the following items: CIANB STARTALL CIANB STOPALL CIANB SITE PARAMETERS CIANB PURGE EVENT LOG CIANB REGISTERED LISTENERS
CIANB NIGHTLY TASK	Run routine	This is the nightly cleanup task. It cleans up orphaned session contexts and purges the event log.
CIANB SITE PARAMETERS	Action	Permits editing broker configuration parameters.
CIANB PURGE EVENT LOG	Action	Permits interactive purging of the event log.
CIANB REGISTERED LISTENERS	Edit	Permits editing registered listener daemons.
CIANB STARTALL	Action	Starts primary listener daemons for each entry in the CIA Listener file.
CIANB STOPALL	Action	Stops primary listener daemons for each entry in the CIA Listener file.

## 6.8 Exported Security Keys

None.

## 6.9 Exported Protocols

None.

## 6.10 Exported Parameters

Parameter	Instance Type	Value Type	Precedence	Description
CIANB AUTHENTICATION METHOD	UCI	Set	System	Controls the authentication method employed by the associated UCI: 0=normal; 1=client-cached; 2=server-cached.
CIANB POLLING INTERVAL		Numeric	System	This is the interval in seconds that the client will poll the server for signaled events or asynchronous remote procedure calls.
CIANB RESOURCE DEVICE COUNT		Numeric	System	The maximum number (1-20) of resource devices that can be created.
CIANB RESOURCE DEVICE SLOTS		Numeric	System	Maximum number (1-20) of slots per resource device.

## 6.11 Exported Mail Groups

None.

## 6.12 Callable Routines

This section and those that follow describe the various routines that comprise the Broker and the supported means for interacting with those routines.

### 6.12.1 Server Management

These API calls perform such functions as starting and stopping the listener daemons.

#### 6.12.1.1 **DEBUG^CIANBLIS**

Scope: public.

Starts the secondary listener daemon as a foreground process for debugging purposes. The application will prompt for an IP address (defaults to localhost) and port which will be used to perform the client callback. This information is provided by the client application when a connection request is made in debug mode.

#### 6.12.1.2 **MSERVER^CIANBLIS**

Scope: public.

This is the entry point for use by MSM when the primary listener daemon is setup to run as a service (see configuration section).

#### 6.12.1.3 **START^CIANBLIS**

Scope: public.

Parameter	Datatype	Description
PORT	Integer	TCP port number that the listener will monitor for connection requests.

Starts the primary listener daemon on the specified port.

#### 0.0.0.1 **STARTALL^CIANBLIS**

Scope: public.

Starts all enabled listener daemons that are registered in the CIA LISTENER file.

#### 0.0.0.2 **STOP^CIANBLIS**

Scope: public.

Parameter	Datatype	Description
PORT	Integer	Port number of the listener process that is to be stopped.
IP	String (optional)	IP address of the connection to be terminated. If not specified, assumes a primary listener is being stopped. Otherwise, stops the secondary listener connected to the named IP address.

Stops a primary or secondary listener daemon on the specified port and (optionally) IP address.

#### 6.12.1.4 STOPALL^CIANBLIS

Scope: public.

Stops all running listener daemons that are registered in the CIA LISTENER file.

### 6.12.2 Session Management

#### 6.12.2.1 RPC: CIANBRPC GETSESS

Scope: public.

Parameter	Datatype	Description
VAR	String (optional)	^-delimited list of state variables to return. Defaults to the standard list of: UID^WID^AID^DUZ^USER^LDT
AID	String (optional)	If specified, restricts the list to sessions registered under the specified application identifier.
<return value>	String array	List of active sessions in the format determined by the VAR parameter.

Returns a list of active sessions.

#### 6.12.2.2 \$\$SESSION^CIANBUTL

Scope: public.

Parameter	Datatype	Description
UID	Integer	Session identifier
VAR	String (optional)	^-delimited list of state variables to return. Defaults to the standard list of: UID^WID^AID^DUZ^USER^LDT
<return value>	String	^-delimited list of state variable values as specified in VAR parameter.

Returns a ^-delimited string of state variable values for the specified session. By default, the list consists of the session identifier, the workstation identifier, the application identifier, the user internal identifier (DUZ), the user's name, and the logon date/time, respectively. However, the data elements returned can be tailored by specifying a different list of state variable in the VAR parameter.

**6.12.2.3 \$\$SHOWSESS^CIANBUTL / SHOWSESS^CIANBUTL**

Scope: public.

Parameter	Datatype	Description
AID	String (optional)	If specified, list is limited to sessions registering the specified application identifier.
<return value>	Integer	Returns the number of sessions displayed.

Displays information about currently running sessions. If an application identifier is specified, displays only sessions registering that identifier. When called as an extrinsic, returns the number of sessions displayed.

**6.12.2.4 \$\$GETUID^CIANBUTL**

Scope: public.

Retrieves the identifier of the current session. If the process is running within a Telnet window, an attempt is made to discover the parent session by issuing an answerback request to the Telnet application. If the Telnet application responds with an appropriately formatted session identifier, and the user assigned to that session matches the user associated with the Telnet application, that identifier is returned. If a session identifier cannot be determined, a null value is returned.

**6.12.2.5 \$\$NXTUID^CIANBUTL / NXTUID^CIANBUTL**

Scope: public.

Parameter	Datatype	Description
UID (by reference)	Integer (optional)	In: identifier of last session Out: identifier of next session
FLT	Integer (optional)	Session filter. Can be: <0 = all sessions 0 = inactive only >0 = active only (default)
AID	Integer (optional)	If specified, only sessions associated with the application identifier are considered.
<return value>	Boolean	If true, a session matching the specified criteria was found. If false, a session was not found.

Returns the identifier of the next (or first if UID is not specified) session that matches the specified criteria. Successive calls to this procedure can be used to return all sessions matching the specified criteria.

**6.12.2.6 \$\$CLRVAR^CIANBUTL / CLRVAR^CIANBUTL**

Scope: public.

Parameter	Datatype	Description
NMSP	String (optional)	Name of namespace to search. Defaults to the global namespace.
UID	Integer (optional)	Identifier of session whose state data is to be cleared. Defaults to current session.
<return value>	Boolean	Returns true if the operation was successful.

Deletes all state variables from the specified namespace and session.

### 6.12.2.7 \$\$GETVAR^CIANBUTL

Scope: public.

Parameter	Datatype	Description
NAME	String	State variable name to retrieve
DFLT	String (optional)	Default value to return if variable does not exist. If not specified, returns null if it does not exist.
NMSP	String (optional)	Name of namespace to search. Defaults to the global namespace.
UID	Integer (optional)	Identifier of session whose state data is to be searched. Defaults to current session.
<return value>	String array	Value of specified state variable.

Retrieves the value of a state variable.

State variables are the preferred way to store session-level information on the server. This ensures that this information will persist across RPC boundaries. The use of namespaces is strongly recommended to avoid naming collisions with other applications.

### 6.12.2.8 \$\$SETVAR^CIANBUTL / SETVAR^CIANBUTL

Scope: public.

Parameter	Datatype	Description
NAME	String	Name of state variable whose value is to be set.
VALUE	String (optional)	Value to assign. If not specified, the state variable is deleted if it exists.
NMSP	String (optional)	Name of namespace to search. Defaults to the global namespace.
UID	Integer (optional)	Identifier of session whose state data is to be modified. Defaults to current session.
<return value>	Boolean	Returns true if the operation was successful.

Sets a state variable to the specified value or, if no value is specified, then deletes the state variable if it exists.

### 6.12.2.9 RPC: CIANBRPC GETVAR

Scope: public.

Parameter	Datatype	Description
LIST	String or string array	Name or names of state variables whose values are to be retrieved. can be specified as a single variable name or an indexed list of variable names.
NMSP	String (optional)	Name of namespace to search. Defaults to the global namespace.
<return value>	String array	List of values in the format: <variable name>=<value> The order matches that specified in the LIST parameter.

Retrieves the value(s) of one or more state variables.

### 6.12.2.10 RPC: CIANBRPC SETVAR

Scope: public.

Parameter	Datatype	Description
LIST	String or string array	Zero or more name-value pairs to be set. Pass multiple pairs as an indexed list. Format for each entry is: <variable name>=<value>
NMSP	String (optional)	Name of namespace to search. Defaults to the global namespace.
RESET	Boolean (optional)	If true, the namespace is cleared of all state variables before setting the values.
<return value>	Integer	Number of variables set.

Sets the value(s) of one or more state variables.

## 6.12.3 Event Management

### 6.12.3.1 \$\$BRDCAST^CIANBEVT / BRDCAST^CIANBEVT / RPC: CIANBEVT BCAST

Scope: public.

Parameter	Datatype	Description
TYPE	String	Event type to broadcast
STUB	String	Event stub
LST	String array (optional)	Recipient list See description of \$\$BRDCAST^CIANBEVT
AID	String (optional)	Application ID – if not specified, all applications are included.
<return value>	Integer	The number of events broadcast.

Broadcasts an event to all subscribers (LST not specified) or to a list of recipients (LST specified). If LST is specified, it should contain entries in one of two formats:

LST(“DUZ”,<DUZ>) or LST(“UID”,<UID>)

where <DUZ> is the DUZ of the intended recipient (in which case the event is broadcast to all sessions for that user) and <UID> is the identifier of a session.

If AID is specified, only sessions registered under that application identifier are signaled.

### 6.12.3.2 DOPURGE^CIANBEVT

Scope: public.

Parameter	Datatype	Description
SILENT	Boolean (optional)	If true, no user interaction occurs. If false or not specified, the user receives feedback as the purge progresses.

Purges the *CIA EVENT LOG* file according to the settings specified in the *CIA EVENT TYPE* file.

### 6.12.3.3 RPC: CIANBEVT GETSUBSC

Scope: public.

Parameter	Datatype	Description
TYPE	String	Event type
<return value>	String array	List of subscribing sessions.

Returns a list of subscribing sessions for the specified event. The return format is the same as that for the CIANBRPC GETSESSN RPC.

### 6.12.3.4 QUEUE^CIANBEVT

Scope: public.

Parameter	Datatype	Description
TYPE	String	Event type to send
STUB	String	Event stub
UID	Integer	Session ID of recipient.

Send an event to the specified session. If the recipient is not a subscriber to the specified event, no action is taken.

### 6.12.3.5 \$\$RELATES^CIANBEVT

Scope: public.

Parameter	Datatype	Description
EVENTA	String or Integer	Name or internal entry number of first event type.
EVENTB	String or Integer	Name or internal entry number of second event type.
<return value>	Integer	Returns a value that reflects the relationship between the two events. Possible values are: 0 = none 1 = same 2 = A is parent of B 3 = B is parent of A

Returns a value that indicates the hierarchical relationship between two event types.

### 6.12.3.6 SUBSCR^CIANBEVT / \$\$SUBSCR^CIANBEVT

Scope: public.

Parameter	Datatype	Description
TYPE	String	Event type
SUBSCR	Boolean	If true, the session is making a subscription request. If false, the session is requesting to unsubscribe.
<return value>	Boolean	Returns the new subscription status: true=subscribed, false=not subscribed.

Requests or withdraws a subscription to the named event.



**6.12.3.7 TASKPRG^CIANBEVT**

Scope: public.

Tasks the event log purge to run in the background.

**6.12.3.8 UNSUBALL^CIANBEVT**

Scope: public.

Withdraws all subscriptions for the session.

**6.12.4 Miscellaneous****6.12.4.1 CLEANUP^CIANBUTL**

Scope: public.

Purges data for all inactive sessions. Sessions which terminate abnormally can not clean up their persistent data store. This procedure performs this cleanup for inactive sessions.

**6.12.4.2 REBLDCTX^CIANBUTL**

Scope: public.

Marks cached RPC context tables for all sessions to be rebuilt. This is useful when a change has been made to an RPC context that needs to be propagated to running sessions and has no adverse effect on running sessions.

**6.12.4.3 RPC: CIANBRPC CANRUN**

Scope: public.

Parameter	Datatype	Description
RPC	String	Name of remote procedure to check.
<return value>	Boolean	Returns true if the specified remote procedure can be called in the current context.

Determines if a remote procedure can be executed. A remote procedure can be executed if it exists on the remote host and access is allowed for the user.

**6.12.4.4 \$\$GETDLG^CIANBUTL / GETDLG^CIANBUTL**

Scope: private.

Parameter	Datatype	Description
NUM	Integer	Dialog number to retrieve. If this value is < 10,000, it is assumed to be relative to the base dialog number assigned to the Broker. Otherwise, it is assumed to be an absolute dialog number.
DLG (by reference)	String array	Receives the text of the requested dialog.

Parameter	Datatype	Description
P1, P2, P3	String (optional)	Optional parameters that can replace imbedded parameters within the dialog text.
<return value>	String	The first line of dialog text.

Retrieves the specified dialog text.

#### 6.12.4.5 RPC: CIANBRPC DIALOG

Scope: private.

Parameter	Datatype	Description
NUM	Integer	Dialog number to retrieve. If this value is < 10,000, it is assumed to be relative to the base dialog number assigned to the Broker. Otherwise, it is assumed to be an absolute dialog number.
P1, P2, P3	String (optional)	Optional parameters that can replace imbedded parameters within the dialog text.
<return value>	String array	Receives the text of the requested dialog.

#### 6.12.5 External Relations

Entity	Name	Description
Package	VA FileMan 22	Uses support APIs.
Package	Kernel 8.0	Uses support APIs.
Package	Kernel Toolkit 8.0	Uses support APIs.
Package	CIA Utilities 1.1	Uses support APIs.

### 6.13 Internal Relations

None.

### 6.14 Archiving and Purging

This package has no archiving capabilities. Purging is performed by the CIANB NIGHTLY TASK which, as the name implies, should be scheduled to run on a nightly basis. Failure to do this will result in a progressive deterioration in performance over time.

### 6.15 Components

#### 6.15.1 Delphi Component

The Delphi VCL component, TvcRPCBroker has the following methods and properties:

### 6.15.1.1 Properties

Property	Datatype	Access	Description
AppID	String	RW	The identifier of the application context. The application context is the name of an entry in the OPTION file and determines accessibility of the Broker application by the authenticated user.
Authentication	Enum: TRPCAuthMethod	R	Authentication method used. One of: amNormal – Prompt for username and password and authenticate against remote host. amCache – Cache the username and password for the remote host in encrypted form in the user section of the Windows registry. amNT – Use NT authentication credentials for authenticating against the remote host. Credentials are cached in the CIA AUTHENTICATION file on the remote host.
Caption	String	RW	Specifies the caption of the authentication dialog.
Connected	Boolean	RW	Set to true to initiate a server connection. Set to false to terminate a server connection.
DebugMode	Boolean	RW	Set to true to enable debug mode. In this mode, the Broker instructs the user to start the secondary listener using the DEBUG^CIANBLIS entry point.
DomainName	String	R	Returns the domain name of the connected host or null if no active connection.
Options	Enum: TRPCOptions	RW	One of: opNoDivPrompt – Suppresses display of Select Division dialog. Always logs in to the default division for the user. opNoPwdPrompt – Suppresses display of authentication dialog, even if NT-based or cached authentication fails. opNoPwdChange – Suppresses display of Change Password dialog.
Password	String	RW	The password (verify code) to be used for authentication. Reading this value returns no meaningful information.
Param	TParams	RW	Provides access to the parameter list to be used in the upcoming remote procedure call. See discussion of RPC parameters below.
Picture	TPicture	RW	Graphic image used as background for the authentication dialog.
PictureSettings	Set: TPictureSettings	RW	Controls sizing of graphic image. Any combination of: psProportional – Maintain original aspect ratio. psStretch – Stretch the image to fill the available space. psCenter – Center the image in the available space.
Port	Integer	RW	The TCP port number used by the primary listener daemon to listen for connection requests.
RemoteProcedure	String	RW	The name of the remote procedure.
Results	TStrings	RW	The data returned by the remote procedure call.

Property	Datatype	Access	Description
RPCContext	String	RW	The context to be used for remote procedure calls. The context is the name of an entry in the OPTION file of type RPC that specifies which remote procedures can be executed.
RPCTimeLimit	Integer	RW	The timeout period, in seconds, after which the wait for completion of a pending synchronous remote procedure expires.
RPCVersion	String	RW	The version of the remote procedure being called. This value can be inspected by the remote procedure in the CIA("VER") variable.
Server	String	RW	The name or IP address of the remote host.
ServerAddress	String	R	The IP address of the remote host or null if there is no active connection.
ServerID	String	R	Returns information about the active connection in the format: <server IP>:<server port>:<UCI>
SessionID	Integer	R	The identifier of the active session, or zero if there is no active connection.
SignonMessage	TStrings	R	The signon message text as provided by the host system.
SiteName	String	R	The name of the facility the user is currently logged into, or null if there is no active connection.
UCI	String	RW	The UCI under which the secondary listener daemon is to run.
Username	String	RW	The username (access code) to use for authentication.

Event	Parameters	Description
OnAsync	Sender: TObject Handle: Integer Data: String	Triggered when an asynchronous remote procedure call has completed successfully. Handle uniquely identifies the call and data is the data returned by the call.
OnAsyncError	Sender: TObject Handle: Integer Code: Integer Text: String	Triggered when an asynchronous remote procedure call has generated an unhandled exception. Handle uniquely identifies the call and code and text are the error code and text returned by the host.
OnConnect	Sender: TObject	Triggered when the Connected property transitions from false to true.
OnDisconnect	Sender: TObject	Triggered when the Connected property transitions from true to false.
OnEvent	Sender: TObject Name: String Data: String	Triggered when the host signals an event to which the Broker session has subscribed. Name is the event name and data is its associated data (event stub).

### 6.15.1.2 Call

Performs a synchronous remote procedure call using existing property values, returning data in the Results property.

**6.15.1.3 CallAsync**

Parameter	Datatype	Description
<return value>	Integer	Returns a handle uniquely identifying the pending call.

Performs an asynchronous remote procedure call using existing property values.

**6.15.1.4 CallRPCAsync**

Parameter	Datatype	Description
RPC	String	The remote procedure name.
Args	Array of const	List of arguments to be passed to the remote procedure.
<return value>	Integer	Returns a handle uniquely identifying the pending call.

Performs an asynchronous remote procedure call using the passed parameter values.

**6.15.1.5 CallList**

Parameter	Datatype	Description
List	TStrings	Receives the data returned by the remote procedure.

Performs a synchronous remote procedure call using existing property values, returning data in the List parameter.

**6.15.1.6 CallRPCStr**

Parameter	Datatype	Description
RPC	String	The remote procedure name.
Args	Array of const	List of arguments to be passed to the remote procedure.
<return value>	String	The string data returned by the remote procedure.

Performs a synchronous remote procedure call using the passed parameter values and returning a string value.

**6.15.1.7 CallStr**

Parameter	Datatype	Description
<return value>	String	The string data returned by the remote procedure.

Performs a synchronous remote procedure call using existing property values and returning a string value.

**6.15.1.8 Connect**

Parameter	Datatype	Description
<return value>	Boolean	Returns true if an active connection exists, false otherwise.

Attempts a connection to a remote host if an active connection does not already exist. Return true if an active connection exists upon completion of the call.

**6.15.1.9 Disconnect**

Terminates the connection to the remote host. If an active connection does not exist, this procedure has no effect.

**6.15.1.10 EventSubscribe**

Parameter	Datatype	Description
EventName	String	Name of the event for which a subscription is requested.
DoSubscribe	Boolean	If true, a subscription is being requested. If false, a subscription is being revoked.
<return value>	Boolean	Returns the subscription status at the completion of the call: true if an active subscription exists, false if not.

Use this function to establish and revoke event subscriptions. Events are signaled through the OnEvent callback.

**6.15.1.11 GetServerInfo**

Parameter	Datatype	Description
<return value>	Boolean	Returns true if a server was selected.

This function sets the Broker properties, Server, Port, UCI from information provided in the vcBroker.ini configuration file. If more than one server is listed in the configuration file's Servers section, a Select Server dialog is presented. If only one server is listed, it is used without user intervention. If no servers are listed or the configuration file could not be located, an exception is raised. The function returns true if the server-related properties were successfully initialized.

**6.15.1.12 Lock**

Parameter	Datatype	Description
Wait	Boolean	If true, the application blocks until the lock request is successful.
<return value>	Boolean	Returns true if the lock request was successful.

Locks the broker via a critical section. This can be used to lock out other threads from performing operations that can interfere with the current thread.

**6.15.1.13 LockGlobal**

Parameter	Datatype	Description
GlobalRef	String	Closed reference of global to lock.
Release	Boolean	If true, an existing lock is released. If false, an incremental lock is applied.
Timeout	Integer	Timeout interval in seconds. Defaults to zero.
<return value>	Boolean	Returns true if lock request was successful.

Applies or releases an incremental lock on the specified global reference.

**6.15.1.14 RestoreState**

Returns the Broker to the state prior to the last call to SaveState. If there is no saved state, this procedure has no effect. See discussion of the SaveState procedure for more information.

**6.15.1.15 SaveState**

Saves the current state of the Broker. The following property values are saved: RemoteProcedure, RPCContext, RPCVersion, Params, and Results. To restore the Broker to its saved state, use the RestoreState procedure.

**6.15.1.16 Unlock**

Removes a lock established by the Lock method.

**6.15.2 RPC Parameters**

The Broker's Params property exposes a TParams class that represents an indexed array of parameters that are to be passed to the remote procedure. Each parameter is of type TParamItem and is capable of storing a single scalar value and any number of subscripted values. The TParamItem class has the following methods and properties:

Property	Datatype	Access	Description
Count	Integer	R	The number of values stored in this parameter.
HasData	Set: THasData	R	Returns information about the data stored in this parameter. This is a set which can include any combination of: hdValue – The value property has been set. hdMult – At least one subscripted value has been set.
Value	String	RW	Gets or sets the scalar value associated with this parameter.

**6.15.2.1 Assign**

Parameter	Datatype	Description
Source	TPersistent	The object to be copied. This can be another TParamItem object or a TStrings descendant.

This procedure copies the Source object into the TParamItem object. If the Source is another TParamItem, all values are copied to the destination. If the Source is a TStrings descendant, its contents are copied into the destination's value list with subscript values corresponding to the index values of the original offset by one (i.e., the first TStrings entry, index 0, corresponds to a subscript value of one on the server).

**6.15.2.2 Clear**

The internal value list is cleared.

**6.15.2.3 Delete**

Parameter	Datatype	Description
Subscript	Array of const	Deletes the specified subscript and its associated value.

This procedure deletes a subscript and its associated value.

**6.15.2.4 Get**

Parameter	Datatype	Description
Subscript	String	Comma-delimited, quote-enclosed list of subscripts.
<return value>	String	The value stored at the specified subscript.

This function returns the value at a specified subscript.

**6.15.2.5 Get**

Parameter	Datatype	Description
Subscript	Array of const	List of subscript values.
<return value>	String	The value stored at the specified subscript.

This function returns the value at a specified subscript.

**6.15.2.6 Put**

Parameter	Datatype	Description
Subscript	String	Comma-delimited, quote-enclosed list of subscripts.
Value	String	The value to store at the specified subscript.

This procedure stores a value at the specified subscript. If a value already exists at that location, it is overwritten.

**6.15.2.7 Put**

Parameter	Datatype	Description
Subscript	Array of const	List of subscript values.
Value	String	The value to store at the specified subscript.

This procedure stores a value at the specified subscript. If a value already exists at that location, it is overwritten.

**6.15.2.8 SubscriptAt**

Parameter	Datatype	Description
Index	Integer	The internal index of the value to be returned.
<return value>	String	The subscript corresponding to the specified index.

This function returns the subscript at the specified position within the internally maintained list of values. This is useful for iterating over all subscripts. Note that the Value property is stored internally with a null subscript value. Therefore, the subscript value returned can be null.



### 6.15.2.9 ValueAt

Parameter	Datatype	Description
Index	Integer	The internal index of the value to be returned.
<return value>	String	The subscript corresponding to the specified index.

This function returns the value at the specified position within the internally maintained list of values. This is useful for iterating over all values. Note that the Value property is stored in this list along with all subscripted values.

## 7.0 Site Context Object

### 7.1 Introduction

The site context object is a shared service that contains information about the current site.

### 7.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	CSS_SITE.SITE
Version	4.3.0.9
Class Identifier	{528DA158-7C9E-4AE9-B1A0-48B40EDD66AF}
Image File	CSSSite.dll
Property Initializations	none
Serializable Properties	none
Required Files	Interop.CSS_Site.dll
Security Keys	none
Multiple Instances Allowed	no
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	yes
.Net Component	no
Associated Build	BEHO*1.1*007001

There are no specific implementation or maintenance tasks associated with this component.

### 7.3 Routine Descriptions

This component has been assigned the namespace designation of “BEHOSI”. The following routines are distributed:

Routine	Description
BEHOSICX	Site context support

### 7.4 File List

None.

### 7.5 Cross References

None.

## 7.6 Exported Options

None.

## 7.7 Exported Security Keys

None.

## 7.8 Exported Protocols

None.

## 7.9 Exported Parameters

None.

## 7.10 Exported Mail Groups

None.

## 7.11 Callable Routines

This section describes supported entry points for routines exported with this component.

### 7.11.1 RPC: BEHOSICX SITEINFO

Scope: private.

Parameter	Datatype	Description
<return value>	String List	Returns context data about the current site. The data is returned as a list of the following elements from the INSTITUTION file (#4): Domain Name Name Station Number State Short Name Street Address 1 Street Address 2 City Zip Internal Entry Number

Returns context data about the current site.

## 7.12 External Relations

None.

## 7.13 Internal Relations

None.

## 7.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 7.15 Components

This component supports the following properties:

<b>Property</b>	<b>Datatype</b>	<b>Access</b>	<b>Description</b>
Address1	String	R	First line of the facility's address.
Address2	String	R	Second line of the facility's address
City	String	R	City in which facility is located.
DomainName	String	R	The name of the domain.
Handle	Integer	R	The unique internal identifier for the facility.
FacilityID	String	R	The unique identifier for the facility.
LongName	String	R	The full name of the facility.
ShortName	String	R	The abbreviated name of the facility.
State	String	R	State in which facility is located.
ZipCode	String	R	Zipcode of the facility.

## 8.0 User Context Object

### 8.1 Introduction

The user context object is a shared service that contains information about the current user.

### 8.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	CSS_USER.USER
Version	4.3.0.8
Class Identifier	{C8C185BB-360C-4FAD-BCB7-7AE680550423}
Image File	CSSUser.dll
Property Initializations	none
Serializable Properties	none
Required Files	Interop.CSS_User.dll
Security Keys	none
Multiple Instances Allowed	no
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	yes
.Net Component	no
Associated Build	BEHO*1.1*006001

There are no specific implementation or maintenance tasks associated with this component.

### 8.3 Routine Descriptions

This component has been assigned the namespace designation of “BEHOUS”. The following routines are distributed:

Routine	Description
BEHOUSCX	User context support.

### 8.4 File List

None.

### 8.5 Cross References

None.

## 8.6 Exported Options

None.

## 8.7 Exported Security Keys

None.

## 8.8 Exported Protocols

None.

## 8.9 Exported Parameters

None.

## 8.10 Exported Mail Groups

None.

## 8.11 Callable Routines

This section describes supported entry points for routines exported with this component.

## 8.12 RPC: BEHOUSCX USERINFO

Scope: public.

Parameter	Datatype	Description
User	Pointer (#200)	User for which information is being requested. Defaults to current user.
<return value>	String	User information in the format: DUZ^NAME^USRCLS^CANSIGN^ISPROVIDER^ORDERROLE^ NOORDER^PTMOUT;STMOUT;CNTDN^SRVIEN^SRVNAME

Returns information about the specified user.

## 8.12.1 \$\$ORDROLE^BEHOUSCX

Scope: private.

Parameter	Datatype	Description
<return value>	Integer	User role. One of: 0 = nokey 1 = clerk 2 = nurse 3 = physician 4 = student 5 = bad keys

Returns a code that reflects the ordering role of the current user based on security key possession (OREMAS, ORELSE, ORES, PROVIDER).

## 8.12.2 \$\$ISPROV^BEHOUSCX

Scope: private.

Parameter	Datatype	Description
<return value>	Boolean	<b>True if the current user possesses the PROVIDER security key.</b>

Returns true if the current user possesses the provider key.

## 8.12.3 \$\$HASKEY^BEHOUSCX

Scope: public.

Parameter	Datatype	Description
Name	String	Name of security key or, if prefixed by the “@” character, a Boolean parameter.
User	Pointer (#200)	IEN of user to check. Defaults to current user.
<return value>	Boolean	True if the user possesses the specified security key or has the specified parameter with a value of true.

Checks for the presence of a specified security key or, if the Name parameter begins with an “@” character, returns the value of the named Boolean parameter.

## 8.12.4 RPC: BEHOUSCX HASKEYS

Scope: public.

Parameter	Datatype	Description
Names	String	^~delimited list of security key or Boolean parameter names.

Parameter	Datatype	Description
User	Pointer(#200)	IEN of user to check. Defaults to current user.
<return value>	Boolean	^-delimited list of Boolean values corresponding to Names input list.

Checks for the presence of a specified security keys or Boolean parameters.

### 8.12.5 RPC: BEHOUSCX NEWPERS

Scope: public.

Parameter	Datatype	Description
From	String	Starting point for file search.
Direction	Integer	1=forward search; -1=backward search.
Key	String	Optional security key. If specified, only users possessing the key will be returned.
Date	Date	If specified, only users active on the specified date will be returned.
Filter	String	Any combination of: A=active users only; D=current division only. Defaults to "AD".
Maximum	Integer	Maximum number of entries to return. Defaults to 44.
<return value>	String List	Returns a list of users meeting the above criteria in the format: <ien>^<name>

Returns a list of entries from the NEW PERSON file that match the specified criteria.

### 8.12.6 \$\$ACTIVE^BEHOUSCX

Scope: public.

Parameter	Datatype	Description
User	Pointer (#200)	IEN of user to check.
Date	Date	Reference date to check.
<return value>	Boolean	Returns true if the user was active on or before the specified date.

Determines if the specified user was active on or before the specified date.

### 8.12.7 RPC: BEHOUSCX VALIDSIG

Scope: public.

Parameter	Datatype	Description
Esig	String	Encrypted electronic signature code.
<return value>	Boolean	True if the specified electronic signature code matches that on file for the user.

Validates an electronic signature code.



### 8.12.8 RPC: BEHOUSCX VALIDPSW

Scope: public.

Parameter	Datatype	Description
Password	String	Encrypted verify code.
<return value>	Boolean	True if the specified verify code matches that on file for the user.

Validates a verify code.

### 8.12.9 RPC: BEHOUSCX HASFMCD

Scope: public.

Parameter	Datatype	Description
Code	String	FileMan access code.
<return value>	Boolean	True if the user possesses the specified FileMan access code.

Verifies if the current user possesses the specified FileMan access code. This will always return true for users that possess the “@” FileMan access code.

## 8.13 External Relations

None.

## 8.14 Internal Relations

None.

## 8.15 Archiving and Purging

There are no archiving or purging requirements within this software.

## 8.16 Components

This component supports the following properties and methods:

## 8.16.1 Properties

Property	Datatype	Access	Description
CanSignOrders	Boolean	R	True if user is authorized to sign orders.
Countdown	Integer	R	Determines how long countdown timer dialog lingers. Comes from the CIAVM COUNTDOWN INTERVAL parameter.
Esig	String	R	User's electronic signature code.
Handle	Integer	R	Internal entry number (DUZ) for the user.
IsProvider	Boolean	R	True if user possesses the PROVIDER security key.
Name	String	R	Users full name in format: Last, First Middle
NoOrdering	Boolean	R	True if ordering is disabled. Comes from the ORWOR DISABLE ORDERING parameter.
OrderRole	Integer	R	User's ordering role. One of: 0=nokey, 1=clerk, 2=nurse, 3=physician, 4=student, 5=bad keys
TimeOut	Integer	R	Primary timeout interval for the user. Comes from the CIAVM PRIMARY TIMEOUT parameter.
TimeOut2	Integer	R	Secondary timeout interval for the user. Comes from the CIAVM SECONDARY TIMEOUT parameter.
Service	Integer	R	Internal entry number of the service to which the user is assigned.
ServiceName	String	R	Name of the service to which the user is assigned.
UserClass	Integer	R	User's class. Determined by which ordering key is possessed: 3 = ORES 2 = ORELSE 1 = OREMAS 0 = none

## 8.16.2 ESigValidate

Parameter	Datatype	Description
Esig	String	Unencrypted electronic signature code to validate. If null, the current electronic signature code is returned in encrypted form.
<return value>	String	Null if the specified code failed validation. Otherwise, the validated electronic signature code in encrypted form is returned.

Validates the specified electronic signature code and returns it in encrypted form.

### 8.16.3 HasKey

Parameter	Datatype	Description
KeyName	String	Name of security key or Boolean parameter to check.
<return value>	Boolean	True if user possesses specified security key or if the specified parameter evaluates to true.

Determines if the user possesses the specified security key or, if the KeyName is prefixed with an "@" character, returns the value of the specified Boolean parameter.

### 8.16.4 HasKeys

Parameter	Datatype	Description
KeyNames	String	List of ^-delimited security key or Boolean parameter names.
<return value>	String	^-delimited list of values corresponding to the KeyNames input.

Returns a ^-delimited list of Boolean values corresponding to an input list of security key or Boolean parameter names.

## 9.0 Patient Context Object

### 9.1 Introduction

The patient context object is a shared service that contains information about the current patient.

### 9.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	CSS_PATIENT.PATIENT
Version	4.3.0.59
Class Identifier	{8955EBC9-3342-40B1-8295-FE20415CCA4D}
Image File	CSSPatient.dll
Property Initializations	PHOTOMASK=@BEHOPTCX PHOTO MASK LOOKUPIDS=NNNN;NNNNN;NNNNNN;TNNNNN;tNNNNN^BEHOPTPL HRNLKP N;NN;`NNN;`NNNN;`NNNNN;`NNNNNN;`NNNNNNN;`NNNN NNNN^BEHOPTPL IENLKP BNNNNNN;BNNNNNNNN^BEHOPTPL DOBLKP
Serializable Properties	none
Required Files	CSSPatient.chm Interop.CSS_Patient.dll
Security Keys	none
Multiple Instances Allowed	no
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	yes
.Net Component	no
Associated Build	BEHO*1.1*004002

There are no specific implementation or maintenance tasks associated with this component.

### 9.3 Routine Descriptions

This component has been assigned the namespace designation of “BEHOPT”. The following routines are distributed:

Routine	Description
BEHOPTCX	Patient context object support
BEHOPTIN	Installation support
BEHOPTPL	Patient list support
BEHOPTP1	
BEHOPTP2	
BEHOPTPC	Primary provider data access

## 9.4 File List

This component has been assigned the file number range of 90460.03 through 90460.0399. The following file is distributed:

### 9.4.1 BEH PATIENT LIST (#90460.03)

This file contains control information for patient lists available in the patient selection dialog.

Field Name	#	Datatype	Indexes	Description
NAME	.01	Text	B – Standard	Unique name for the list.
FLAGS	1	Text		These flags control how the list behaves on the client. Can be any combination of the following values: D – Date range required E – Sets encounter context L – Item retrieval uses long list M – Convert selection list to mixed case N – Do not cache list S – Sort selection list U – List can be managed by user
ENTITY	2	Text		Name of the entity that subcategorizes the list (e.g., clinic location).
SEQUENCE	3	Integer		Controls sequencing of lists in the patient selection dialog.
DISABLE	4	Boolean		If yes, the list is disabled and will not appear in the patient selection dialog.
PATIENT RETRIEVAL	10	M Code		M code for retrieving patients for a given entity.
ITEM RETRIEVAL	11	M Code		M code for retrieving items for the entity list.
LIST MANAGEMENT	12	M Code		M code for performing list management operations such as adding or removing patients.
SCREEN	13	M Code		M code for screening list entries.

## 9.5 Cross References

Cross references are described in the preceding section.

## 9.6 Exported Options

Option	Type	Description
BEHOPT MAIN	menu	Main menu for patient context.
BEHOPTCX DEMO MODE	action	Limit viewing to demo patients only for a specific user.
BEHOPTCX DETAIL REPORT	action	Set logic for patient detail report.

Option	Type	Description
BEHOPTCX RECALL LAST	action	Set parameter for recalling last patient viewed.
BEHOPTPL DATE RANGES	action	Set default date ranges for patient selection dialog.
BEHOPTPL EXAMINE/PRINT	action	Examine or print a patient list.
BEHOPTPL TEAM ADD	action	Allows team list creation or the addition of autolinks, providers, and/or patients to existing lists.
BEHOPTPL TEAM DELETE	action	Delete a team list.
BEHOPTPL TEAM DELETE AUTOLINKS	action	Remove existing autolinks from a team list and the patients associated with the removed autolinks.
BEHOPTPL TEAM DELETE	action	Remove patients from a team list.
BEHOPTPL TEAM DELETE USERS	action	Remove users from a team list.
BEHOPTPL TEAM MENU	menu	Menu to manage team lists.
BEHOPTPL TEAM RENAME	action	Rename a team list.
BEHOPTPL TEAM USER	action	Displays teams from the OE/RR LIST file linked to a user.
BEHOPTPL TEAM USER PTS	action	Displays patients linked to a user via teams from the OE/RR LIST file.

## 9.7 Exported Security Keys

None.

## 9.8 Exported Protocols

None.

## 9.9 Exported Parameters

Parameter	Instance Type	Value Type	Precedence	Description
BEHOPTCX DEMO MODE		Boolean	User, Division, System	If yes, only demo patients can be selected.
BEHOPTCX DETAIL REPORT		String	User, Service, Division, System	M code to generate a patient detail report. Replace this code to create a custom report.

Parameter	Instance Type	Value Type	Precedence	Description
BEHOPTCX LAST PATIENT	Facility	Pointer (#2)	User	Saves the last patient selected for the current facility.
BEHOPTCX PHOTO MASK		String	System	This is the file mask to be used to retrieve patient photographs. A null entry disables display of patient photos. Format is: <directory path>\%d.<file extension> where <directory path> is the path to the shared directory where photos are stored. &#x25;d is automatically replaced by the patient's internal entry number. <file extension> is the extension that reflects the graphics format.  For example,  \\server1\c\photos\%d.jpg
BEHOPTCX RECALL LAST		Boolean	User, Division, System	If yes, the patient context is set to the last patient selected upon startup.
BEHOPTPL DATE RANGES		Word Processing	System	Default date ranges for patient selection dialog.
BEHOPTPL DEFAULT ITEM	Pointer (#90460.03)	String	User, System	Determines which list subcategory is selected by default for each patient list category.
BEHOPTPL DEFAULT SOURCE		Pointer (#90460.03)	User, System	Determines which patient list is displayed by default in the patient selection dialog.
BEHOPTPL PERSONAL LIST	String	Word Processing	User	Each instance specifies the name of a personal list and its value is a list of patient IEN's belonging to the list.

## 9.10 Exported Mail Groups

None.

## 9.11 Callable Routines

This section describes supported entry points for routines exported with this component.

### 9.11.1 RPC: BEHOPTCX CHKDUP

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number.
<return value>	String List	Checks for patients similar to specified patient by last name and last 4 digits of the SSN. If any found, returns a text message listing the possible alternates.

Returns a formatted list of patients similar to requested patient.

### 9.11.2 \$\$HRN^BEHOPTCX

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number.
<return value>	String	Returns the patient health record number for the active facility or null if the patient is not registered to the active facility.

Returns the active facility's health record number for a patient.

### 9.11.3 \$\$ICN^BEHOPTCX

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number.
<return value>	String	The patient's integration control number or null if none has been assigned.

Returns the patient's integration control number as assigned by the Master Patient Index.

### 9.11.4 RPC: BEHOPTCX ICN2DFN

Scope: private.

Parameter	Datatype	Description
ICN	String	Patient's integration control number as assigned by the Master Patient Index.
<return value>	Pointer (#2)	The internal entry number of the corresponding patient, or null if none found or if this feature is not installed.



Returns the internal entry number for the patient with the specified integration control number. If there is no Master Patient Index capability installed, this function will always return null.

### 9.11.5 RPC: BEHOPTCX INPLOC

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number.
<return value>	String	Returns information about the patient's current inpatient location containing the following ^-delimited elements: Ward Location IEN (42) Ward Name Service If the patient is not an inpatient, these elements will be null.

Returns information about the patient's current inpatient location.

### 9.11.6 \$\$ISACTIVE^BEHOPTCX

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number.
Demo (optional)	Boolean	If true, patient must be a demo patient. If not specified, defaults to the value of the BEHOPTCX DEMO MODE parameter.
<return value>	Boolean	Returns true if the patient is currently active within the system. For RPMS sites, further restricts this to patients registered within the active facility.

Returns true if the specified patient is active within the system.

### 9.11.7 \$\$ISSENS^BEHOPTCX

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number.
<return value>	Boolean	Returns true if the patient is marked as sensitive.

Returns true if the specified patient is marked as a sensitive patient.

### 0.0.1 RPC: BEHOPTCX LAST

Scope: private.

Parameter	Datatype	Description
DFN (optional)	Pointer (#2)	Patient's internal entry number. If not specified, the default patient is not changed.
<return value>	Pointer (#2)	The internal entry number of the default patient. If this feature is disabled, returns null.

Gets or sets the last patient selected. This patient is used as the default context when the patient context object is initialized. The BEHOPTCX RECALL LAST parameter must be set to true to enable this function.

### 9.11.8 RPC: BEHOPTCX LEGACY

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number.
<return value>	String List	Message text if data resides in legacy system.

This RPC is provided for sites that are the product of the consolidation of two or more sites where the patient can have data residing on one of the legacy systems that is not available for review within the current system. The message returned in this case can be used to alert the user to this fact.

### 9.11.9 RPC: BEHOPTCX PCDETAIL

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number.
<return value>	String List	Returns a detailed summary about the patient's primary care team.

Generates a detailed summary about a patient's assigned primary care team.

### 9.11.10 RPC: BEHOPTCX PTINFO

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number.
SLCT (optional)	Boolean	If true, patient is saved as the last patient selected. If this feature is enabled, this becomes the default patient context for the user.

Parameter	Datatype	Description
<return value>	String	Returns information about the patient as a ^-delimited string containing the follow elements: Name Sex Date of Birth SSN Location IEN (44) Location Name Room/Bed Location Veteran Flag Sensitive Patient Flag Date of Admission Health Record Number Service Connected Flag Service Connect % Integration Control # Date of Death Treating Specialty Primary Team Primary Provider Attending Physician

Returns basic information about the specified patient and, optionally, sets that patient as the last selected.

### 9.11.11 RPC: BEHOPTCX PTINQ

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number.
<return value>	String List	Returns a detailed patient inquiry report.

Generates a detailed patient inquiry report. The BEHOPTCX DETAIL REPORT parameter controls the format and content of this report by specifying the code that generates it. To supply an alternate format, modify this parameter.

### 9.11.12 \$\$SETCTX^BEHOPTCX

Scope: public.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number.
<return value> (optional)	Boolean	Returns true if the context change request was successfully submitted. Does not guarantee that the context change was accepted by the application.

Issues a patient context change request to the current session. If this is executed in an imbedded Telnet session using the vcTelnet component, the routine will correctly detect the session context of the application and direct the context change request to that session context.

## 9.11.13 RPC: BEHOPTPC DETAIL

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number.
<return value>	String List	Returns formatted information about a patient's primary care team.

Retrieves detailed information about a patient's primary care team.

## 9.11.14 \$\$OUTPTR^BEHOPTPC

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number.
<return value>	String	Returns the patient's primary care provider in the format: IEN (#200)^Name

Retrieves the patient's primary care provider or null if none found.

## 9.11.15 \$\$OUTPTM^BEHOPTPC

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number.
<return value>	String	Returns the patient's primary care team in the format: IEN (#9009017.5)^Name

Retrieves the patient's primary care team or null if none found.

## 9.11.16 TEAM^BEHOPTPC

Scope: private.

Parameter	Datatype	Description
USER	Pointer (#200)	IEN of primary care provider.

Returns all providers on the team associated with the given primary care provider in the global reference ^TMP("ORIHS",\$J,IEN) where IEN is the IEN (#200) of each associated provider.

## 9.11.17 RPC: BEHOPTPL CLINRNG

Scope: private.

Parameter	Datatype	Description
<return value>	String List	Returns a list of standard date ranges for patient list types that require these. Each entry has the format: <div style="text-align: center;">&lt;range specifier&gt;^&lt;display text&gt;</div> Where <range specifier> is either “S” for user selectable, or the format: <div style="text-align: center;">&lt;start&gt;;&lt;end&gt;</div> where <start> and <end> can be any date or relative date in external format.  Example: T-7;T^Past Week

Returns a list of standard date ranges for patient list types that require these. For example, a patient list whose entity type is appointment might require the selection of a date range over which appointment will be displayed.

### 9.11.18 RPC: BEHOPTPL DOBLKP

Scope: private.

Parameter	Datatype	Description
DOB	String	Date of birth in external format, prefixed with a “B” character.
<return value>	String List	List of patients with a matching identifier. Each entry has the following ^-delimited elements: Patient IEN (#2) Patient Name HRN + Date of birth (formatted)

Returns a list of patients with the matching date of birth.

### 9.11.19 RPC: BEHOPTPL GETDFLT

Scope: private.

Parameter	Datatype	Description
<return value>	String	Returns the list specifier for the default list in the same format as the return value for the BEHOPTPL LISTINFO remote procedure.

Returns information about the default list. If there is no default defined, returns null.

### 9.11.20 RPC: BEHOPTPL HRNLKP

Scope: private.

Parameter	Datatype	Description
HRN	String	Health record number (with our without hyphens).

Parameter	Datatype	Description
<return value>	String List	List of patients with a matching identifier. Each entry has the following ^-delimited elements: Patient IEN (#2) Patient Name HRN + Date of birth (formatted)

Returns a list of patients with the matching health record number.

### 9.11.21 RPC: BEHOPTPL IENLKP

Scope: private.

Parameter	Datatype	Description
IEN	String	Patient's internal entry number prefixed with a "" character.
<return value>	String List	List of patients with a matching identifier. Each entry has the following ^-delimited elements: Patient IEN (#2) Patient Name HRN + Date of birth (formatted)

Returns a list of patients with the matching internal entry number. Because there can be at most one match for this identifier, the list will be at most one entry in length.

### 9.11.22 RPC: BEHOPTPL LISTALL

Scope: private.

Parameter	Datatype	Description
FROM	String	Patient name just prior to start of list.
DIR	Integer	Direction of search. 1=forward, -1=reverse.
MAX (optional)	Integer	Maximum number of entries. Defaults to 44.
<return value>	String List	Alphabetical list of patients in the form: <IEN (#2)>^<name>

Returns an alphabetical list of patients, starting after the specified position. Only patients registered to the active facility are included.

### 9.11.23 RPC: BEHOPTPL LISTINFO

Scope: private.

Parameter	Datatype	Description
LIST (optional)	IEN (19930.4)	If specified, the internal entry number of the patient list type for which data is to be returned. If not specified, data for all list types are returned.

Parameter	Datatype	Description
<return value>	String or String List	Either a single value or a string list (depending on whether one or all list types are requested). Each entry contains the following ^-delimited elements: List Type IEN (19930.4) Entity Name Flags List Name Default Settings

Returns information about one or more patient lists. Any screening logic defined for the list is applied here. Lists not meeting the screening criteria will be omitted.

### 9.11.24 RPC: BEHOPTPL LISTPTS

Scope: private.

Parameter	Datatype	Description
LIST	Pointer (#19930.4)	Internal entry number of the patient list type.
IEN	Pointer	Internal entry number of the list item selected. The target file for the IEN depends upon the list type.
START (optional)	Date	Optional start date for search.
END (optional)	Date	Optional end date for search.
<return value>	String List	List of patients in the specified list. Each entry is of the form: <IEN (#2)>^<Patient Name>

Returns a list of patients for the specified list. A patient list is specified by the combination of a list type (IEN of list type definition in file 19930.4) which defines the logic for interacting with lists within it, and a list IEN (the target file depends upon the list type) that identifies the specific list within the list type.

### 9.11.25 RPC: BEHOPTPL LISTSEL

Scope: private.

Parameter	Datatype	Description
LIST	String	List specifier in the same format as the return value for the BEHOPTPL LISTINFO RPC.
FROM	String	Starting position for list generation.
DIR	Integer	Search direction. 1=forward, -1=reverse.
MAX	Integer	Maximum number of entries returned.
<return value>	String List	List of subcategories in the form: <IEN>^<Name>

Returns a list of subcategories for the specified list and its associated entity type. For example, for a patient list with an entity type of clinic, this would be a list of clinic locations.

## 9.11.26 RPC: BEHOPTPL LOOKUP

Scope: private.

Parameter	Datatype	Description
ID	String	Social security number in one of the following formats: Full SSN (with or without hyphens) Last 4 digits First initial last name + last 4 digits
<return value>	String List	List of patients with a matching identifier. Each entry has the following ^-delimited elements: Patient IEN (#2) Patient Name Full SSN + Date of birth (external format)

Returns a list of patients with the matching full or partial social security number.

## 9.11.27 RPC: BEHOPTPL MANAGE

Scope: private.

Parameter	Datatype	Description
LIST	Pointer (#19930.4)	Internal entry number of the patient list type.
ACTION	Byte	Specifies the type of action to be performed on the list. One of: C – Create list D – Delete list R – Rename list S – Set list contents
NAME	String	The name of the list.
VAL	String List	Format depends upon the action performed: C – not used D – not used R – new name for list S – contents for list
<return>	String	If an error occurred, returns <error code>^<error text>. Otherwise, returns null.

For patient lists that can be managed by the user, this RPC provides support for list management activities.

## 9.11.28 RPC: BEHOPTPL SAVEDFLT

Scope: private.

Parameter	Datatype	Description
LIST (optional)	Pointer (#19930.4)	Internal entry number of the patient list type. If not specified, the default list is set to none.
VAL (optional)	String List	List of default settings for each list type in the format described for the return value for the BEHOPTPL LISTINFO remote procedure.
<return value>	String	If an error occurred, returns <error code>^<error text>. Otherwise, returns null.



Save default settings for patient lists, including the default list type and the default settings for each list type.

### 9.11.29 \$\$SSN^BEHOPTPL

Scope: private.

Parameter	Datatype	Description
DFN	IEN (2)	Patient's internal entry number.
<return value>	String	Social security number formatted with hyphens.

Returns the patient's formatted social security number.

### 9.11.30 External Relations

Entity	Name	Description
Package	PIMS	version 5.3

## 9.12 Internal Relations

None.

## 9.13 Archiving and Purging

There are no archiving or purging requirements within this software.

## 9.14 Components

This component supports the following properties and methods:

### 9.14.1 Properties

Patient identifier properties that are writable (*Handle* and *ICN*) can be conditionally modified by an application. This means that requesting a change to one of these properties is honored only if all context participants (both local and global) agree to the change. Therefore, an application must not assume that the change occurred but instead should either check the value after the assignment or wait for acknowledgement of the change (the *ICSS\_PatientEvents* event set).

Property	Datatype	Access	Description
AdmitDate	DateTime	R	If an inpatient, this is the date and time of the current admission.
Age	Single	R	The patient's age, computed from the current date.
Attending	String	R	If an inpatient, the name of the patient's attending physician.
DOB	DateTime	R	The patient's date of birth.
DOD	Date	R	Date of the patient's death.

Property	Datatype	Access	Description
Handle	Integer	RW	The host-specific handle identifying the patient (a.k.a., DFN). See description that follows for discussion of writable properties.
HRN	String	RW	Health record number for patient. See description that follows for discussion of writable properties.
HistoryLength	Integer	RW	Maximum number of entries in the patient selection history list.
ICN	String	RW	The patient's integration control number if one has been assigned. See description that follows for discussion about writable properties.
IsInpatient	Boolean	R	True if the patient is currently an inpatient.
IsRestricted	Boolean	R	If true, access to this patient's information is restricted.
IsServiceConnected	Boolean	R	If true, the patient has a service-connected disability.
Name	String	R	The patient's full name, formatted as Last, First, Middle.
Location	Integer	R	If an inpatient, the internal identifier of the ward location.
LocationName	String	R	If an inpatient, the name of the ward location.
LookupDelay	Integer	RW	Delay in milliseconds before a lookup is attempted on data entered into the patient lookup selection dialog.
LookupIDs	String	RW	Specifies additional lookup masks that can trap specific inputs in the patient selection dialog and perform special lookups.
PercentServiceConnected	Integer	R	Returns the service-connected status of the patient as a percentage.
PhotoMask	String	RW	Provides a mask that translates to the full path and filename of the image file for the patient's photograph. Use %d in the specification to indicate where the patient handle is to be stored. For example, \\server\photos%d.jpg
PhotoPath	String	R	Returns the full path to where patient photos are stored.
PrimaryProvider	String	R	The name of the patient's primary care provider.
PrimaryTeam	String	R	If an inpatient, the name of the patient's primary team.
RoomBed	String	R	If an inpatient, the room and bed number.
Sex	String	R	Indicates the patient's sex. One of: M = male; F = female; U = unknown
Specialty	Integer	R	Treating specialty.
SSN	String	R	The patient's social security number, formatted as nnn-nn-nnnn.

### 9.14.2 Clear

Clears the patient context.

### 9.14.3 Detail

<b>Parameter</b>	<b>Datatype</b>	<b>Description</b>
Modeless	Boolean	If true, the dialog is displayed amodally. If false, the dialog is displayed modally.
AllowPrint	Boolean	If true, a print button appears on the dialog allowing the content to be directed to a print device.

Presents a dialog containing the patient detail report.

### 9.14.4 Select

This procedure displays the standard patient selection dialog.

## 10.0 Encounter Context Object

### 10.1 Introduction

The encounter context object is a shared service that contains information about the current encounter. It is important to note that an encounter does not necessarily equate to a visit. It is possible to set an encounter context without an associated visit. Some operations require an encounter context but do not require a visit (e.g., placing orders). Other operations require both (e.g., updating immunizations). The encounter context object has methods that accommodate these needs and can coerce the creation of a visit based on the current encounter context data when a visit is required.

The encounter context object is linked to the patient context object. A change to the patient context will clear the encounter context. In addition, the selection of an encounter context for a patient other than the one in the current context will trigger a patient context change.

### 10.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	CSS_ENCOUNTER.ENCOUNTER
Version	4.3.0.121
Class Identifier	{5C87BBCA-BC10-462B-9DB7-6F1E886C3D3F}
Image File	CSSEncounter.dll
Property Initializations	none
Serializable Properties	none
Required Files	Interop.CSS_Encounter.dll
Security Keys	none
Multiple Instances Allowed	no
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	yes
.Net Component	no
Associated Build	BEHO*1.1*005001

There are no specific implementation or maintenance tasks associated with this component.

### 10.3 Routine Descriptions

This component has been assigned the namespace designation of “BEHOEN”. The following routines are distributed:

<b>Routine</b>	<b>Description</b>
BEHOENCX	Encounter context support.
BEHOENIN	Installation support.
BEHOENPC BEHOENPI	PCC data management.
BEHOENPP BEHOENPR BEHOENPS BEHOENPV	Encounter summary report.

## 10.4 File List

None.

## 10.5 Cross References

None.

## 10.6 Exported Options

<b>Option</b>	<b>Type</b>	<b>Description</b>
BEHOEN MAIN	menu	Encounter context configuration main menu.
BEHOENCX CREATE VISIT	action	Allow user to create new visits.
BEHOENCX OTHER LOCATION	action	Specify a general location for outside encounters.
BEHOENCX PROVIDER	action	Allow a user to be a visit provider.
BEHOENCX SEARCH RANGE START	action	Visit search start date.
BEHOENCX SEARCH RANGE STOP	action	Visit Search Stop Date
BEHOENCX VISIT LOCK OVERRIDE	action	Temporarily override a visit lock for a user.
BEHOENCX VISIT LOCKED	action	Set number of days after which a visit is locked.
BEHOENCX VISIT TYPES	action	Set selectable service categories.

## 10.7 Exported Security Keys

None.

## 10.8 Exported Protocols

None.

## 10.9 Exported Parameters

Parameter	Instance Type	Value Type	Precedence	Description
BEHOENCX CREATE VISIT		Boolean	User, Class, Service, Location, Division, System	If yes, the user can create visits. This enables the new visit tab of the encounter selection dialog.
BEHOENCX PROVIDER		Boolean	User, Class	If yes, user can be a provider associated with a visit. This controls which users appear in the provider list of the encounter selection dialog.
BEHOENCX SEARCH RANGE START		String	User, Service, Division, System	Returns the relative date to start listing visits for a patient. For example, 'T-90' will list visits beginning 90 days before today.
BEHOENCX SEARCH RANGE STOP		String	User, Service, Division, System	Returns the relative date to end listing visits for a patient. For example, 'T' will not list visits later than today. 'T+30' will not list visits after 30 days from now.
BEHOENCX VISIT TYPES	Numeric (sequence #)	String	Division, System	Specifies the service categories selectable from the encounter selection dialog. The format for each entry is: Category Code~Short Descriptor~Long Descriptor
BEHOENCX VISIT LOCKED		Numeric	Division, System	This parameter determines the maximum # of days (1-180) following the creation of a visit after which the visit cannot be modified. Once this period has passed, no additional PCC data can be attached to a visit.
BEHOENCX OTHER LOCATION		Pointer (#9999999.06)	Division, System	This is a general location is stored for visits that have an outside location.
BEHOENCX VISIT LOCK OVERRIDE	Pointer (#9000010)	Boolean	User	Use this parameter to temporarily override a locked visit for a specific user. Be sure to remove the override after the user has completed the necessary modifications.

## 10.10 Exported Mail Groups

None.

## 10.11 Callable Routines

This section describes supported entry points for routines exported with this component.

### 10.11.1 \$\$ACTLOC^BEHOENCX

Scope: public.

Parameter	Datatype	Description
LOC	Pointer (#44)	Internal entry number of locations.
DAT (optional)	Date	Date to check for active status. Defaults to today.
<return value>	Boolean	True if location was active on given date. False if location was inactive, is non-existent, or is not part of active facility.

Returns true if given location was active on the given date for the current facility.

### 10.11.2 \$\$ADDP RV^BEHOENCX

Scope: public.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number.
VSTR	String	Visit string.
PRV	String Array	String array where subscript is the provider IEN (200) and each entry has the following ^-delimited elements: Provider IEN (#200) Provider Name Primary Flag Encounter Date/Time
<return value> (optional)	String	If an error occurred, contains <error code>^<error text>. Otherwise, returns null.

Replaces current list of providers associated with a visit with the specified list.

### 10.11.3 RPC: BEHOENCX ADMITCUR

Scope: public.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number
<return value>	String	Information for the current admission. Contains the following ^-delimited elements: Visit String Location Name Date of Admission (FM) Type Visit Lock Flag

Returns information about the current admission, or null if the patient is not an inpatient.

## 10.11.4 \$\$ADMITINF^BEHOENCX

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number
MOV	Pointer (#405)	IEN of entry in PATIENT MOVEMENT file.
<return value>	String	Information on the specified admission. Contains the following ^-delimited elements: Visit String Location Name Date of Admission (FM) Type Visit Lock Flag

Returns information on the specified admission.

## 10.11.5 RPC: BEHOENCX ADMITLST

Scope: public.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number
<return value>	String List	List of past hospital admissions. Each entry contains the following ^-delimited elements: Visit String Location Name Date of Admission (FM) Type Visit Lock Flag

Returns a list of recent hospital admissions for a given patient.

## 10.11.6 RPC: BEHOENCX APPTLST

Scope: public.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number
<return value>	String List	List of appointments within the last 30 days. Each entry contains the following ^-delimited elements: Appointment Date/Time (FM) Location IEN Location Name Status

Returns a list of recent appointments for a given patient.

## 10.11.7 RPC: BEHOENCX CLINLOC

Scope: public.



Parameter	Datatype	Description
FROM (optional)	String	Name of location that precedes start of list. Defaults to begin listing at first/last entry (depending on DIR).
DIR (optional)	Integer	Direction of search (1=forward, -1=reverse). Defaults to forward.
MAX (optional)	Integer	Maximum number of entries to return. Defaults to 44.
<return value>	String List	Returns list of clinic locations. Each entry consists of the following ^-delimited elements: Location IEN Location Name

Returns an alphabetic list of clinic locations.

### 10.11.8 RPC: BEHOENCX INPLOC

Scope: public.

Parameter	Datatype	Description
FROM (optional)	String	Name of location that precedes start of list. Defaults to begin listing at first/last entry (depending on DIR).
DIR (optional)	Integer	Direction of search (1=forward, -1=reverse). Defaults to forward.
MAX (optional)	Integer	Maximum number of entries to return. Defaults to 44.
<return value>	String List	Returns list of inpatient locations. Each entry consists of the following ^-delimited elements: Location IEN Location Name

Returns an alphabetic list of inpatient locations.

### 10.11.9 RPC: BEHOENCX FETCH

Scope: public.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number
VSTR	String	Visit string
PRV (optional)	Pointer (#200)	Internal entry number of the providers associated with the visit. Can be specified as a single value, or an indexed list of values.
CREATE (optional)	Integer	Controls creation of a visit. One of: -1=Always create 0=never 1=Create if match not found

Parameter	Datatype	Description
<return value>	String	Visit data containing the following ^-delimited elements: Location Name Location Abbreviation Room/Bed Assignment Provider IEN Primary Provider Name Visit File IEN Visit Identifier Lock Flag Error Text If no visit was found (and the CREATE flag was false), the return value will be null.

Fetches information about the specified visit and, optionally, creates a visit if a corresponding visit was not found. If the PRV parameter is presented, those providers replace any providers attached to the visit and the first provider specified is designated the primary.

#### 10.11.10 \$\$FNDVIS^BEHOENCX

Scope: public.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number
DAT	FM Date/Time	Visit date/time.
CAT	String	Service category.
LOC	Pointer (#44)	Hospital Location IEN.
CRE	Integer	Controls creation of a visit. One of: -1=Always create 0=never 1=Create if match not found
PRV (optional)	Pointer (#200)	Provider IEN to restrict search.
ELC (optional)	Pointer (#4) or String	Encounter location. If numeric, is assumed to be a pointer. Otherwise, is assumed to be a free text location in which case the BEHOENCX OTHER LOCATION parameter will determine the pointer value used.
<return value>	Pointer (#9000010)	Returns a pointer to the visit file entry if successful, or -1^Error Text if not.

Used to locate a matching visit or create a new visit with the specified characteristics. Calls the GETVISIT^BSDAPI4 visit lookup/creation API.

#### 10.11.11 RPC: BEHOENCX GETPRV

Scope: public.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number.
VSTR	String	Visit string.
PRI (optional)	Boolean	If true, return only the primary provider. Defaults to false.
<return value>	String Array	Returns a list of providers associated with the specified visit. Each entry has the following ^-delimited elements: Provider IEN (#200) Provider Name Primary Flag Encounter Date/Time

Returns a list of providers associated with the given visit. If the PRI flag is set, returns only the primary provider.

### 10.11.12 RPC: BEHOENCX GETPRV2

Scope: public.

Parameter	Datatype	Description
IEN	Pointer (#9000010)	Internal entry number of visit.
PRI (optional)	Boolean	If true, return only the primary provider. Defaults to false.
<return value>	String Array	Returns a list of providers associated with the specified visit. Each entry has the following ^-delimited elements: Provider IEN (#200) Provider Name Primary Flag Encounter Date/Time

Returns a list of providers associated with the given visit. If the PRI flag is set, returns only the primary provider. Similar to BEHOENCX GETPRV except that it takes the visit IEN instead of a visit string.

### 10.11.13 RPC: BEHOENCX GETVISIT

Scope: public.

Parameter	Datatype	Description
IEN	Pointer (#9000010)	Internal entry number of the visit in the VISIT file.
<return value>	String	Visit data containing the following ^-delimited elements: Location Name Visit Date (FM format) Service Category Patient IEN Visit ID Lock Flag If no entry is found, the return value is null.

Returns information about a specific VISIT file entry.

## 10.11.14 \$\$ISLOCKED^BEHOENCX

Scope: public.

Parameter	Datatype	Description
IEN	Pointer (#9000010)	IEN of VISIT file entry.
<return value>	Boolean	Returns true if the specified visit is locked from changes.

Returns true if the specified visit is locked from changes.

## 10.11.15 RPC: BEHOENCX LOCIEN

Scope: public.

Parameter	Datatype	Description
LOC	String	Location name to lookup.
<return value>	Pointer (#44)	Internal entry number of the location corresponding to the specified location name, or 0 if none found.

Returns the internal entry number of the HOSPITAL LOCATION file entry corresponding to the given location name.

## 10.11.16 RPC: BEHOENCX LOCINFO

Scope: public.

Parameter	Datatype	Description
LOC	Pointer (#44)	Location IEN to retrieve.
<return value>	String	Entire zero node of the HOSPITAL LOCATION file entry.

Returns the entire zero node of the specified HOSPITAL LOCATION file entry.

## 10.11.17 \$\$SC2LOC^ BEHOENCX

Scope: public.

Parameter	Datatype	Description
SC	Pointer (#40.7)	Internal entry number of the stop code.
DAT (optional)	Date	Limit to only locations active on a given date. Defaults to today.
<return value>	Pointer (#44)	Returns the first active clinic associated with the given stop code. Requires the ASTOP cross reference on the HOSPITAL LOCATION file. Returns 0 if no active location found.

Returns an active clinic location associated with the given stop code. If there are multiple active clinic locations associated with the stop code, returns only the first. This requires the addition of the ASTOP cross reference on the STOP CODE NUMBER field of the HOSPITAL LOCATION file. This call is used to resolve clinic locations where only a stop code is provided.

## 10.11.18 \$\$SETCTX^BEHOENCX

Scope: public.

Parameter	Datatype	Description
VST	Pointer (#9000010) or String	Internal entry number of the visit or a visit string.
<return value>	Boolean	Returns true if the context change request was successfully submitted. Does not guarantee that the context change was accepted by the application.

Issues an encounter context change request to the current session. If this is executed in an imbedded Telnet session using the vcTelnet component, the routine will correctly detect the session context of the application and direct the context change request to that session context.

## 10.11.19 RPC: BEHOENCX VID2IEN

Scope: public.

Parameter	Datatype	Description
VID	String	Visit identifier to resolve.
<return value>	Pointer (#9000010)	Internal entry number of the visit corresponding to the specified visit identifier, or 0 if none found.

Returns the internal entry number of the VISIT file entry corresponding to the given visit identifier.

## 10.11.20 RPC: BEHOENCX VISITLST

Scope: public.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number.
BEG (optional)	Date	Starting date of search. Defaults to setting of the BEHOENCX SEARCH RANGE START parameter.
END (optional)	Date	Ending date of search. Defaults to setting of the BEHOENCX SEARCH RANGE STOP parameter.
<return value>	String Array	Returns list of appointment and visits within the specified date range. Each entry contains the following ^-delimited elements: Visit String Location Name Date (FM) Status

Returns a list of visits and appointments within the specified date range.

## 10.11.21 \$\$VIS2VSTR^BEHOENCX

Scope: public.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number
IEN	Pointer (#9000010)	IEN of a VISIT file entry.
ERR (returned)	String	If an error is encountered, returned as -1^Error Text.
<return value>	String	Visit string corresponding to specified visit.

Returns a visit string given a visit IEN.

## 10.11.22 \$\$VSTR2VIS^BEHOENCX

Scope: public.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number
VSTR	String	Visit string
CREATE (optional)	Integer	Controls creation of a visit. One of: -1=Always create 0=Never create 1=Create if match not found Defaults to 0.
PRV (optional)	Pointer (#200)	IEN of provider to further restrict search.
<return value>	Pointer (#9000010)	Returns the internal entry number of the corresponding visit, or 0 if no visit was found.

Returns the internal entry number of the entry in the VISIT file corresponding to the specified visit string.

## 10.12 External Relations

Entity	Name	Description
Package	PIMS	version 5.3

## 10.13 Internal Relations

None.

## 10.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 10.15 Components

This component supports the following properties and methods:

### 10.15.1 Properties

Encounter identifier properties that are writable (*Handle*, *VisitID*, and *VisitStr*) can be conditionally modified by an application. This means that requesting a change to one of these properties is honored only if all context participants (both local and global) agree to the change. Therefore, an application must not assume that the change occurred but instead should either check the value after the assignment or wait for acknowledgement of the change (the *ICSS\_EncounterEvents* event set).

Property	Datatype	Access	Description
DateTime	DateTime	R	The date and time of the encounter.
Handle	Integer	RW	The internal identifier of the visit file entry associated with the encounter context. If set, the context is set to the visit represented by that entry.
Inpatient	Boolean	R	True if this is an inpatient encounter.
LocationName	String	R	The name of the encounter location.
LocationAbbr	String	R	The abbreviated name of the encounter location.
Location	Integer	R	The unique identifier of the encounter location.
Locked	Boolean	R	Returns true if the associated visit is locked from further changes.
Prepared	Boolean	R	True if a valid encounter has been recorded.
ProviderName	String	R	The name of the provider associated with the encounter.
Provider	Integer	R	The unique identifier of the provider associated with the encounter.
RoomBed	String	R	The room and bed # for an inpatient.
Standalone	Boolean	R	A true value indicates that this is a standalone encounter.
VisitCategory	Byte	R	Indicates the category of the visit. Corresponds to the <i>Service Category</i> field of the <i>Visit</i> file.
VisitID	String	RW	The unique identifier for the encounter. If set, the context is set to the encounter represented by that visit identifier.
VisitStr	String	RW	A concatenation of the Location, DateTime, and VisitCategory properties. These values, along with a patient identifier, represent the minimum set necessary to uniquely define an encounter.

### 10.15.2 EnsureHandle

Parameter	Datatype	Description
<return value>	Integer	Returns the value of the Handle property. If a visit has not been associated with the current context and the context is valid, one will be created automatically.

Ensures that a visit is associated with an encounter context.

### 10.15.3 Prepare

Parameter	Datatype	Description
OptionFlags	Integer	Sets various options for the function. Can be any combination of: ofProvider (1) = Limit user selection to providers only ofSuppress (2) = Suppress user selection ofNotLocked (4) = Visit cannot be locked ofValidateOnly (8) = Valid context with no user interaction ofPromptOnInvalid (16) = Prompt only if not valid ofForceVisit (32) = Force visit creation
<return value>	Boolean	True if a valid encounter was prepared.

Invokes the encounter selection dialog that allows the user to select or create an encounter context.

### 10.15.4 SelectLocation

Parameter	Datatype	Description
LocationType	Enum	Can be one of: ItAll (0)=All locations ItOutpatient (1)=Outpatient locations ItInpatient(2)=Inpatient locations.
HelpInfo	String	Help information to be displayed on the dialog.
<return value>	Integer	Internal entry number of the selected location.

This function presents a standard location selection dialog and returns the internal entry number of the location selected or zero if the dialog is cancelled.



## 11.0 Patient Identification Header

### 11.1 Introduction



\*Littlewolf,Peggy LYNN  
2184 20-Mar-1938 (66)

Figure 11-1: Sample Patient Identification Header

The patient identification header displays basic demographic information about the currently selected patient. It is closely tied to, but separate from, the patient context object. Clicking on the patient identification header produces the standard patient selection dialog of the patient context object.

### 11.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHPATIENTID.PATIENTID
Version	4.2.0.14
Class Identifier	{6606CA1B-9B9B-4718-BD57-98EE36D0292D}
Image File	BEHPatientID.ocx
Property Initializations	MINHEIGHT=50 MINWIDTH=200
Serializable Properties	AUTOSIZE=BOOL BORDERSTYLE=ENUM COLOR=COLOR
Required Files	BEHPatientID.chm
Security Keys	none
Multiple Instances Allowed	yes
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BEHO*1.1*025001

There are no specific implementation or maintenance tasks associated with this component.

### 11.3 Routine Descriptions

None.

### 11.4 File List

None.

## 11.5 Cross References

None.

## 11.6 Exported Options

None.

## 11.7 Exported Security Keys

None.

## 11.8 Exported Protocols

None.

## 11.9 Exported Parameters

None.

## 11.10 Exported Mail Groups

None.

## 11.11 Callable Routines

None.

## 11.12 External Relations

None.

## 11.13 Internal Relations

Entity	Name	Description
Component	Patient Context Object	Uses support APIs.

## 11.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 11.15 Components

This component supports the following properties:

## 11.15.1 Properties

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
AUTOSIZE	Boolean	RW	If true, the component automatically resizes itself to accommodate its contents.
BORDERSTYLE	Enum	RW	Sets the style of the border surrounding the component. Can be one of: 0 = None 1 = Single 2 = Sunken 3 = Raised
COLOR	Color	RW	Sets the background color of the component.
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPPFILE	String	RW	Sets the name of the help file associated with the component.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
MINHEIGHT	Integer	RW	Sets the minimum height, in pixels, that the component can attain.
MINWIDTH	Integer	RW	Sets the minimum width, in pixels, that the component can attain.
THEMEAWARE	Boolean	RW	If true, the component is rendered as a themed button.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 12.0 Encounter Information Header

### 12.1 Introduction

A screenshot of a yellow rectangular box containing the text: 01GENERAL 15-Oct-2004 12:10 DOCTOR\_TEST

Figure 12-1: Sample Encounter Information Header

The encounter information header displays information about the current encounter context. It is closely tied to, but separate from, the encounter context object. Clicking on the encounter information header produces the standard encounter selection dialog of the encounter context object.

### 12.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHENCOUNTERINFO.ENCOUNTERINFO
Version	4.2.0.12
Class Identifier	{52117103-AA97-40FF-82D4-66FDBF2FD26D}
Image File	BEHEncounterInfo.ocx
Property Initializations	MINHEIGHT=50, MINWIDTH=200
Serializable Properties	AUTOSIZE=BOOL, BORDERSTYLE=ENUM, COLOR=COLOR
Required Files	none
Security Keys	none
Multiple Instances Allowed	yes
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BEHO*1.1*031002

There are no specific implementation or maintenance tasks associated with this component.

### 12.3 Routine Descriptions

None.

### 12.4 File List

None.

## 12.5 Cross References

None.

## 12.6 Exported Options

None.

## 12.7 Exported Security Keys

None.

## 12.8 Exported Protocols

None.

## 12.9 Exported Parameters

None.

## 12.10 Exported Mail Groups

None.

## 12.11 Callable Routines

None.

## 12.12 External Relations

None.

## 12.13 Internal Relations

Entity	Name	Description
Component	Encounter Context Object	Uses support APIs.

## 12.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 12.15 Components

This component supports the following properties:

## 12.15.1 Properties

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
AUTOSIZE	Boolean	RW	If true, the component automatically resizes itself to accommodate its contents.
BORDERSTYLE	Enum	RW	Sets the style of the border surrounding the component. Can be one of: 0 = None 1 = Single 2 = Sunken 3 = Raised
COLOR	Color	RW	Sets the background color of the component.
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPPFILE	String	RW	Sets the name of the help file associated with the component.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
MINHEIGHT	Integer	RW	Sets the minimum height, in pixels, that the component can attain.
MINWIDTH	Integer	RW	Sets the minimum width, in pixels, that the component can attain.
THEMEAWARE	Boolean	RW	If true, the component is rendered as a themed button.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 13.0 Vital Measurement Entry

### 13.1 Introduction

Default Units	18-Jun-2007 09:43	Range	Units
<input checked="" type="radio"/> Temperature			F
<input type="radio"/> Pulse		60 - 100	/min
<input type="radio"/> Respirations			/min
<input type="radio"/> O2 Saturation			%
<input type="radio"/> Peak Flow			
<input type="radio"/> Blood Pressure		90 - 150	mmHg
<input type="radio"/> Height			in
<input type="radio"/> Weight			lb
<input type="radio"/> Pain			

Figure 13-1: Sample Vital Measurement Entry Form

Vital measurement entry is implemented as a service that can be invoked from another component (e.g., the Vital Measurement Display component on the cover sheet) or from a custom menu and as a visual component that can be dropped directly into the user interface in design mode. Both are contained with the same executable image.

### 13.2 Implementation and Maintenance

This object has the following configurations:

#### 13.2.1 Vital Measurement Entry (service)

Entity	Value
Programmatic Identifier	BEHVITALENTRY.VITALENTRY
Version	2.0.0.107
Class Identifier	{F599A6C4-D16C-4C29-8F24-0CC53434BAE1}
Image File	BEHVitalEntry.dll
Property Initializations	none
Serializable Properties	none
Required Files	none
Security Keys	none
Multiple Instances Allowed	no
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no

Entity	Value
Side-by-Side Versioning	yes
Service	yes
.Net Component	no
Associated Build	BEHO*1.1*001002

### 13.2.2 Vital Measurement Entry (visual component)

Entity	Value
Programmatic Identifier	BEHVITALENTRY.VITALENTRY2
Version	2.0.0.107
Class Identifier	{7E7F5068-B8FC-4B4D-9285-5F59CEB1A145}
Image File	BEHVitalEntry.dll
Property Initializations	none
Serializable Properties	AUTOSIZE=BOOL, BORDERSTYLE=ENUM, CAPTION=TEXT, CAPTIONCOLOR1=COLOR, CAPTIONCOLOR2=COLOR, CAPTIONSTYLE=ENUM, COLOR=COLOR
Required Files	none
Security Keys	none
Multiple Instances Allowed	no
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BEHO*1.1*001002

There are no specific implementation or maintenance tasks associated with this component.

## 13.3 Routine Descriptions

This component has been assigned the namespace designation of “BEHOVM”. The following routines are distributed:

Routine	Description
BEHON001	NTEG routine
BEHOVM	Vital measurement support
BEHOVMIN	Installation support

## 13.4 File List

This component has been assigned the file number range of 90460.01 through 90460.0199. The following files are distributed:



### 13.4.1 BEH MEASUREMENT CONTROL (#90460.01)

This file controls which measurement types can be viewed and manipulated within the RPMS-EHR application.

Field Name	#	Datatype	Indexes	Description
NAME	.01	Text	B – Standard	Full display name for this measurement type.
ABBREVIATION	.5	Text	B – Standard	Mnemonic abbreviation for this measurement type.
DEFAULT UNITS	1	Set		Default units for storage. One of: 0=US; 1=Metric
UNITS (US)	2	Text		Units of measurement for US system.
UNITS (METRIC)	3	Text		Units of measurement for metric system.
NORMAL LO	4	Numeric		Low normal value in default units.
NORMAL HI	5	Numeric		High normal value in default units.
INPUT TRANSFORM	6	M Code		M code to transform value in X to internal form.
US TO METRIC	7	M Code		M code to convert value in X from US to metric units.
METRIC TO US	8	M Code		M code to convert value in X from metric to US units.
PERCENTILE DATA	9	Word Processing		Control data for generating percentile values. These data are in the format as published by the CDC.
RETRIEVAL LOGIC	10	M Code		Special retrieval logic. This would include measurement types that are computed values (e.g., BMI).
DESCRIPTION	99	Word Processing		Text to be displayed when help is requested.

## 13.5 Cross References

Cross references are described in the preceding section.

## 13.6 Exported Options

Option	Type	Description
BEHOVM DATA ENTRY	action	Set data entry permissions.
BEHOVM DEFAULT UNITS	action	Override default units.
BEHOVM MAIN	menu	Vital measurement configuration menu.
BEHOVM TEMPLATE	action	Create/edit data entry templates.

## 13.7 Exported Security Keys

None.

## 13.8 Exported Protocols

None.

## 13.9 Exported Parameters

Parameter	Instance Type	Value Type	Precedence	Description
BEHOVM USE VMSR		Boolean	System	If true, all functions use PCC for vital measurement storage and retrieval. If false, the Vital Measurement package is used. This is set automatically at installation and should not be modified.
BEHOVM TEMPLATE	Numeric (sequence #)	Pointer (#90460.01)	User, Class, Service, Location, Division, System	Controls the formatting of the vital measurement data entry dialog.
BEHOVM DATA ENTRY		Boolean	User, Class, Service, Location, Division, System	If yes, the user is permitted to enter vital measurement data.
BEHOVM DEFAULT UNITS	Pointer (#90460.01)	Set	User, Class, Service, Location, Division, System	Permits overriding the default units for any measurement type. Possible values are: 0=US; 1=Metric

## 13.10 Exported Mail Groups

None.

## 13.11 Callable Routines

This section describes supported entry points for routines exported with this component.

### 13.11.1 RPC: BEHOVM DETAIL

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number.
START (optional)	FM Date/Time	Start date/time for search.
END (optional)	FM Date/Time	End date/time for search.
RMAX (optional)	Integer	Maximum number of results per measurement type. Defaults to all results.
VITS (optional)	Array	Array of IENs or names of entries in BEH Measurement Control file. If not specified, default to values specified by BEHOVM VITAL LIST parameter.
VSTR (optional)	String	Optional visit string specifier to limit retrieval to a given visit.

Parameter	Datatype	Description
METRIC (optional)	Flag	Controls which units of measure are returned. Possible values are: -1=use default; 0=use US; 1=use metric. Defaults to -1.
<return value>	String List	Text containing detailed information about each requested measurement type.

Returns measurement data for detail view.

### 13.11.2 RPC: BEHOVM GRID

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number.
START (optional)	FM Date/Time	Start date/time for search.
END (optional)	FM Date/Time	End date/time for search.
RMAX (optional)	Integer	Maximum number of results per measurement type. Defaults to all results.
VITS (optional)	Array	Array of IENs or names of entries in BEH Measurement Control file. If not specified, default to values specified by BEHOVM VITAL LIST parameter.
VSTR (optional)	String	Optional visit string specifier to limit retrieval to a given visit.
METRIC (optional)	Flag	Controls which units of measure are returned. Possible values are: -1=use default; 0=use US; 1=use metric. Defaults to -1.
SD (optional)	Integer	Number of significant digits to return.
<return value>	String List	Data formatted for display in a chronological grid.

Returns measurement data for grid view.

### 13.11.3 RPC: BEHOVM HELP

Scope: private.

Parameter	Datatype	Description
VCTL	Pointer (#90460.01)	BEH MEASUREMENT CONTROL file entry.
<return value>	String List	Text of DESCRIPTION field.

Returns text from the DESCRIPTION field of the specified BEH MEASUREMENT CONTROL file entry.

### 13.11.4 RPC: BEHOVM LASTVIT

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number.
START (optional)	FM Date/Time	Start date/time for search.

END (optional)	FM Date/Time	End date/time for search.
VITS (optional)	Array	Array of IENs or names of entries in BEH Measurement Control file. If not specified, default to values specified by BEHOVM VITAL LIST parameter.
METRIC (optional)	Flag	Controls which units of measure are returned. Possible values are: -1=use default; 0=use US; 1=use metric. Defaults to -1.
<return value>	String List	Data formatted for display in a list view.

Returns most recent measurement data for display in a list view.

### 13.11.5 RPC: BEHOVM LIST

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number.
START (optional)	FM Date/Time	Start date/time for search.
END (optional)	FM Date/Time	End date/time for search.
VITS (optional)	Array	Array of IENs or names of entries in BEH Measurement Control file. If not specified, default to values specified by BEHOVM VITAL LIST parameter.
VSTR (optional)	String	Optional visit string specifier to limit retrieval to a given visit.
METRIC (optional)	Flag	Controls which units of measure are returned. Possible values are: -1=use default; 0=use US; 1=use metric. Defaults to -1.
<return value>	String List	Most recent vitals in format: vfile ien^vital name^vital abbr^date/time taken^value+units (US & metric)

Returns most recent vital measurements.

### 13.11.6 RPC: BEHOVM PCTILE

Scope: private.

Parameter	Datatype	Description
VCTL	Pointer (#90460.01)	BEH MEASUREMENT CONTROL file entry.
DFN	Pointer (#2)	Patient's internal entry number.
START	FM Date/Time	Start date/time for search.
END	FM Date/Time	End date/time for search.
METRIC (optional)	Flag	Controls which units of measure are returned. Possible values are: -1=use default; 0=use US; 1=use metric. Defaults to -1.
<return value>	String List	Percentile data.

Returns percentile data for the given BEH MEASUREMENT CONTROL entry that corresponds to the given patient's demographics.

### 13.11.7 RPC: BEHOVM SAVE

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number.

VITS (optional)	Array	Array of records of vital measurement data to store. These are in the same format used for CPRS.
<return value>	String	Returns 0 if successful or -1^error text if not.

Saves vital measurement data to V MEASUREMENT file.

### 13.11.8 RPC: BEHOVM TEMPLATE

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number.
VSTR	String	Visit string specifier to limit retrieval to a given visit.
METRIC (optional)	Flag	Controls which units of measure are returned. Possible values are: -1=use default; 0=use US; 1=use metric. Defaults to -1.
<return value>	String List	Returns data for vital entry template as specified by the BEHOVM TEMPLATE parameter.

Retrieves vital measurement data for populating the data entry grid.

### 13.11.9 RPC: BEHOVM VALIDATE

Scope: private.

Parameter	Datatype	Description
VCTL	Pointer (#90460.01)	BEH MEASUREMENT CONTROL file entry.
METRIC (optional)	Flag	Controls which units of measure are returned. Possible values are: -1=use default; 0=use US; 1=use metric. Defaults to -1.
VALUE	String	Value to be validated.
<return value>	String	If the input value validates successfully, this is the fully validated input value. Otherwise, it is -1^Error Text.

Validates the input value.

## 13.12 External Relations

Entity	Name	Description
File	MEASURE TYPE (#9999999.07)	Read access.
File	V MEASUREMENT (#9000010.01)	Read and write access.

## 13.13 Internal Relations

None.

## 13.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 13.15 Components

### 13.15.1 Service

The Vital Measurement Entry service supports the following properties and methods:

#### 13.15.1.1 Properties

Property	Datatype	Access	Description
Enabled	Boolean	R	If false, user does not have permission to enter vitals.
DefaultDate	Enum	RW	Default date to use for vital measurement entry. One of: 0=current date/time; 1=encounter date/time.
DefaultUnits	Enum	RW	Default units to use for date entry. One of: -1=default for each type; 0=US units; 1=metric units
ItemCount	Integer	R	Count of items in current template.

#### 13.15.1.2 Execute

Invokes the vital measurement data entry dialog.

### 13.15.2 Visual Component

The Vital Measurement Entry visual component supports the following properties and methods:

#### 13.15.2.1 Properties

The properties are described in the following table:

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom

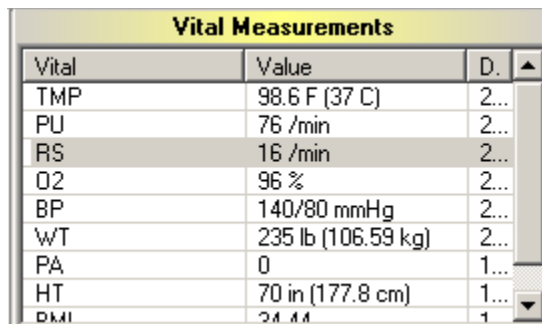
Property	Datatype	Access	Description
AUTOSIZE	Boolean	RW	If true, the component automatically resizes itself to accommodate its contents.
BORDERSTYLE	Enum	RW	Sets the style of the border surrounding the component. Can be one of: 1 = None 2 = Single 3 = Sunken 4 = Raised
CAPTION	String	RW	Sets the text displayed in the title bar. To justify portions of the caption text, use the “\” character to delimit the left-, center-, and right-justified portions of the caption text.
CAPTIONCOLOR1 CAPTIONCOLOR2	Color	RW	Colors to apply to the title bar. If the two colors differ and a gradient style is set, a gradient effect is created. For a standard title bar style, only the first color is applied.
CAPTIONSTYLE	Enum	RW	Sets the caption style. Can be one of: 5 = None – No caption (hides title bar) 6 = Title – Standard title bar 7 = Frame – Framed title bar (group box style) 8 = Left – Left gradient title bar 9 = Right – Right gradient title bar 10 = Center – Center gradient title bar
COLOR	Color	RW	Sets the background color of the component.
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPPFILE	String	RW	Sets the name of the help file associated with the component.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
MINHEIGHT	Integer	RW	Sets the minimum height, in pixels, that the component can attain.
MINWIDTH	Integer	RW	Sets the minimum width, in pixels, that the component can attain.
POPUP	Boolean	RW	If true, data entry screen is invoked as a popup dialog. If false, data entry screen is displayed within the component.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.

<b>Property</b>	<b>Datatype</b>	<b>Access</b>	<b>Description</b>
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.



## 14.0 Vital Measurement Display

### 14.1 Introduction



Vital	Value	D.
TMP	98.6 F (37 C)	2...
PU	76 /min	2...
RS	16 /min	2...
O2	96 %	2...
BP	140/80 mmHg	2...
WT	235 lb (106.59 kg)	2...
PA	0	1...
HT	70 in (177.8 cm)	1...
PKM	24.44	1

Figure 14-1: Sample Vital Measurements Display

The Vital Measurement Display component provides a quick overview of the most recent vital measurements for display on the cover sheet.

### 14.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHVITALS.VITALDISPLAY
Version	5.0.0.183
Class Identifier	{DC21A968-F30E-4E37-B2B0-43B3382E74A4}
Image File	BEHVitals.ocx
Property Initializations	none
Serializable Properties	ALLOWPRINT=BOOL, AUTOSIZE=BOOL, BORDERSTYLE=ENUM, CAPTION=TEXT, CAPTIONCOLOR1=COLOR, CAPTIONCOLOR2=COLOR, CAPTIONSTYLE=ENUM, COLOR=COLOR, DEFERUPDATE=BOOL, DETAILPANE=BOOL, ORIENTATION=ENUM, LAYOUT=HIDDEN
Required Files	BEHVitals.chm
Security Keys	none
Multiple Instances Allowed	yes
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BEHO*1.1*001002

There are no specific implementation or maintenance tasks associated with this component.

### 14.3 Routine Descriptions

None.

### 14.4 File List

None.

### 14.5 Cross References

None.

### 14.6 Exported Options

Option	Type	Description
BEHOVM VITAL LIST	action	Specify measurements listed on cover sheet.

### 14.7 Exported Security Keys

None.

### 14.8 Exported Protocols

None.

### 14.9 Exported Parameters

Parameter	Instance Type	Value Type	Precedence	Description
BEHOVM VITAL LIST	Numeric (sequence #)	Pointer (#90460.01)	User, Class, Service, Location, Division, System	Lists which vitals appear on the cover sheet and in what order.

### 14.10 Exported Mail Groups

None.

### 14.11 Callable Routines

None.

### 14.12 External Relations

Entity	Name	Description
File	MEASURE TYPE (#999999.07)	Read access.
File	V MEASUREMENT (#9000010.01)	Read access.

### 14.13 Internal Relations

Entity	Name	Description
Component	Vital Measurement Entry	Uses supported APIs.

### 14.14 Archiving and Purging

There are no archiving or purging requirements within this software.

### 14.15 Components

This component supports the following properties and methods:

#### 1.15.1 Properties

The following table describes the properties.

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ALLOWPRINT	Boolean	RW	If true, a print button will appear on the detail dialog allowing printing of the contents.
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
AUTOSIZE	Boolean	RW	If true, the component automatically resizes itself to accommodate its contents.

Property	Datatype	Access	Description
BORDERSTYLE	Enum	RW	Sets the style of the border surrounding the component. Can be one of: 0 = None 1 = Single 2 = Sunken 3 = Raised
CAPTION	String	RW	Sets the text displayed in the title bar. To justify portions of the caption text, use the “\” character to delimit the left-, center-, and right-justified portions of the caption text.
CAPTIONCOLOR1 CAPTIONCOLOR2	Color	RW	Colors to apply to the title bar. If the two colors differ and a gradient style is set, a gradient effect is created. For a standard title bar style, only the first color is applied.
CAPTIONSTYLE	Enum	RW	Sets the caption style. Can be one of: 0 = None – No caption (hides title bar) 1 = Title – Standard title bar 2 = Frame – Framed title bar (group box style) 3 = Left – Left gradient title bar 4 = Right – Right gradient title bar 5 = Center – Center gradient title bar
COLOR	Color	RW	Sets the background color of the component.
DEFERUPDATE	Boolean	RW	If true, data refresh is deferred until the component becomes visible. If false, data refresh happens immediately.
DETAILPANE	Boolean	RW	If true, a detail pane appears next to the list view that displays detail text of the selected entry. If false, detail text appears in a popup dialog when an entry is clicked.
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPPFILE	String	RW	Sets the name of the help file associated with the component.
LAYOUT	String	RW	Property representing the internal layout of the form.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
ORIENTATION	Enum	RW	Sets the orientation of the detail pane. Can be one of: 0 = Horizontal 1 = Vertical
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 15.0 Activity Time

### 15.1 Introduction

Figure 15-1: Sample Activity Time

The Activity Time component permits tracking encounter and travel time as related to a specific encounter.

### 15.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	IHSBGOACTIVITYTIME.IHSBGOACTTIMECTRL
Version	1.1.0.176
Class Identifier	{E5836EA1-5EF5-4B1B-AE4B-0EE7E3F8FE3F}
Image File	IhsBgoActivityTime.ocx
Property Initializations	
Serializable Properties	
Required Files	IhsBgoActivityTime.chm
Security Keys	none
Multiple Instances Allowed	no
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BGO*1.1*3

There are no specific implementation or maintenance tasks associated with this component.

### 15.3 Routine Descriptions

This component has been assigned the namespace designation of “BGOVTM.” The following routines are distributed:

<b>Routine</b>	<b>Description</b>
BGOVTM	Support for managing activity time.

## 15.4 File List

None.

## 15.5 Cross References

None.

## 15.6 Exported Options

None.

## 15.7 Exported Security Keys

None.

## 15.8 Exported Protocols

None.

## 15.9 Exported Parameters

None.

## 15.10 Exported Mail Groups

None.

## 15.11 Callable Routines

This section describes supported entry points for routines exported with this component.

## 15.12 RPC: BGOVTM DEL

Scope: private.

<b>Parameter</b>	<b>Datatype</b>	<b>Description</b>
VFIEN	Pointer (#9000010.19)	V file IEN.
<return value>	String	Failure: -n^error text Success: null

Delete specified V ACTIVITY TIME entry.

### 15.12.1 RPC: BGOVTM GET

Scope: private.

Parameter	Datatype	Description
INP	String	Visit IEN ^ User IEN
<return value>	String	Activity Time IEN ^ Activity Time ^ Travel minutes

Get activity time entry associated with given user and visit.

### 15.12.2 RPC: BGOVTM SET

Scope: private.

Parameter	Datatype	Description
INP	String	V File IEN [1] ^ Visit IEN [2] ^ Provider IEN [3] ^ Activity Time [4] ^ Travel Minutes [5]
<return value>	String	Failure: -n^error text Success: null

Add/edit activity time entry.

## 15.13 External Relations

Entity	Name	Description
File	V ACTIVITY TIME (#9000010.19)	Read and write access

## 15.14 Internal Relations

None.

## 15.15 Archiving and Purging

There are no archiving or purging requirements within this software.

## 15.16 Components

This component supports the following properties and methods:

## 15.16.1 Properties

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.



## 16.0 Chief Complaint

### 16.1 Introduction

Author	Chief Complaint
USER_POWER	1.) Symptoms: Chills, Cough, Fever. 2.) Disease: Diabetes. 3.) Requests: Injection, Med Refill, Work Excuse.

Figure 16-1: Sample Chief Complaint

This component permits viewing and managing chief complaint entries. Chief complaint entries are stored in the V NARRATIVE TEXT file with a type of CHIEF COMPLAINT. This component also supports viewing (but not editing) of chief complaint entries in the CHIEF COMPLAINT field of the VISIT file entry.

### 16.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	IHSBGOCHIEFCOMPLAINT.BGOCC
Version	1.1.0.294
Class Identifier	{A7F77B51-F027-42CC-8940-7404D6BC3178}
Image File	IhsBgoChiefComplaint.ocx
Property Initializations	
Serializable Properties	
Required Files	
Security Keys	none
Multiple Instances Allowed	no
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BGO*1.1*3

There are no specific implementation or maintenance tasks associated with this component.

## 16.3 Routine Descriptions

This component has been assigned the namespace designation of “BGOCC”. The following routines are distributed:

Routine	Description
BGOCC	Support for managing chief complaint entries.

## 16.4 File List

The following files are distributed:

### 16.4.1 BGO CHIEF COMPLAINT PICK LIST (#90362.2)

This file contains entries for the pick list choices within the Chief Complaint authoring dialog.

Field Name	#	Datatype	Indexes	Description
NAME	.01	Text	B – Standard	Display text to appear in the pick list.
TYPE	.02	Set	AC – Standard	Determines to which pick list the entry belongs. One of: 1 = Symptom 2 = Disease 3 = Request
BODY LOCATION RELATED	.03	Boolean		If true, the entry can have body location attributes associated with it.

## 16.5 Cross References

Cross references are described in the preceding section.

## 16.6 Exported Options

Option	Type	Description
BGOCC MAIN	menu	Chief complaint configuration menu.
BGO CC PREFIX TEXT	action	Modify prefix text for chief complaint pick lists.

## 16.7 Exported Security Keys

None.

## 16.8 Exported Protocols

None.

## 16.9 Exported Parameters

Parameter	Instance Type	Value Type	Precedence	Description
BGO CC PREFIX TEXT	Set (Pick List)	String	User, Class, Location, Division, System	This parameter allows the overriding of the default prefix text that appears before selections from a chief complaint pick list. The instance value can be one of: Symptom Disease Request
BGO DISABLE CC EDITING		Boolean	User, Class	Disable chief complaint editing.

## 16.10 Exported Mail Groups

None.

## 16.11 Callable Routines

This section describes supported entry points for routines exported with this component.

### 16.11.1 RPC: BGOCC DEL

Scope: private.

Parameter	Datatype	Description
VNT	Pointer (#9000010.34)	IEN of V NARRATIVE TEXT file entry to be deleted.
<return value>	String	Failure: -n^error text Success: null

Deletes the chief complaint entry.

### 16.11.2 RPC: BGOCC DELPL

Scope: private.

Parameter	Datatype	Description
IEN	Pointer (#90362.2)	IEN of BGO CHIEF COMPLAINT PICK LIST entry to be deleted.
<return value>	String	Failure: -n^error text Success: null

Deletes a chief complaint pick list item.

## 16.11.3 RPC: BGOCC GET

Scope: private.

Parameter	Datatype	Description
INP	Pointer (#9000010)	Visit IEN
<return value>	String List	List of records with the format: V Narrative IEN [1] ^ Author (IEN~Name) [2] ^ Line Count [3] Chief Complaint Narrative (multiple lines)

Gets chief complaint entries associated with the specified visit.

## 16.11.4 RPC: BGOCC GETPL

Scope: private.

Parameter	Datatype	Description
TYP	Set	Determines which pick list to retrieve. One of: Symptom Disease Request
<return value>	String List	List of records in the format: IEN ^ Display Text ^ Body Location Related

Gets all entries for the specified chief complaint pick list.

## 16.11.5 RPC: BGOCC SET

Scope: private.

Parameter	Datatype	Description
INP	String	Visit IEN ^ V Narrative IEN ^ Chief Complaint
<return value>	String	Failure: -n^error text Success: V Narrative IEN

Add/edit a chief complaint entry.

## 16.11.6 RPC: BGOCC SETPL

Scope: private.

Parameter	Datatype	Description
INP	String	Name ^ Type ^ Body Related
<return value>	String	Failure: -n^error text Success: null

Add a chief complaint pick list item.

## 16.12 External Relations

Entity	Name	Description
File	V NARRATIVE TEXT (#9000010.34)	Read and write access

## 16.13 Internal Relations

None.

## 16.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 16.15 Components

This component supports the following properties and methods:

### 16.15.1 Properties

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HIDEBUTTONS	Boolean	RW	If true, hides the button controls on the component.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 17.0 Evaluation and Management Coding

### 17.1 Introduction

Figure 17-1: Sample Evaluation and Management

The Evaluation and Management Coding component permits the capture of E&M service codes associated with a visit. These codes are stored in the V CPT file.

### 17.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	IHSBGOEM.BGOEMCTRL
Version	1.1.0.198
Class Identifier	{46F491B5-4E6D-4EB5-8055-44A4F0B9B930}
Image File	IhsBgoE&M.ocx
Property Initializations	none
Serializable Properties	none
Required Files	IhsBgoE&M.chm
Security Keys	none
Multiple Instances Allowed	no
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BGO*1.1*3

There are no specific implementation or maintenance tasks associated with this component.

## 17.3 Routine Descriptions

This component shares routines with the SuperBill component. Please refer to the SuperBill component section for further information.

## 17.4 File List

None.

## 17.5 Cross References

None.

## 17.6 Exported Options

<b>Option</b>	<b>Type</b>	<b>Description</b>
BGO DISABLE E&M EDITING	action	Disable editing of evaluation & management codes.
BGO E&M SUPPRESS CONFIRMATORY	action	Suppress confirmatory E&M codes.
BGO E&M SUPPRESS ER CODES	action	Suppress emergency room E&M codes.
BGO E&M SUPPRESS HOSP CODES	action	Suppress hospital E&M codes.

## 17.7 Exported Security Keys

None.

## 17.8 Exported Protocols

None.

## 17.9 Exported Parameters

Parameter	Instance Type	Value Type	Precedence	Description
BGO DISABLE E&M EDITING		Boolean	User, Class	Disable editing of E&M codes.
BGO E&M SUPPRESS CONFIRMATORY		Boolean	User, Class, Division, Package	Suppress confirmatory E&M codes.
BGO E&M SUPPRESS ER CODES		Boolean	User, Class, Division, Package	Suppress emergency room E&M codes.
BGO E&M SUPPRESS HOSP CODES		Boolean	User, Class, Division, Package	Suppress hospital E&M codes.

## 17.10 Exported Mail Groups

None.

## 17.11 Callable Routines

This component shares routines with the SuperBill component. Please refer to the SuperBill component section for further information.

## 17.12 External Relations

Entity	Name	Description
File	V CPT (#9000010.18)	Read and write access

## 17.13 Internal Relations

Entity	Name	Description
Component	SuperBill	Shares routines

## 17.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 17.15 Components

This component supports the following properties and methods:

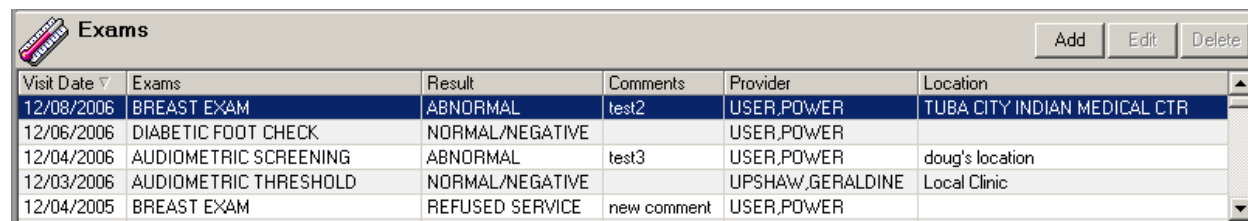


## 17.15.1 Properties

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 18.0 Exams

### 18.1 Introduction



Visit Date	Exams	Result	Comments	Provider	Location
12/08/2006	BREAST EXAM	ABNORMAL	test2	USER,POWER	TUBA CITY INDIAN MEDICAL CTR
12/06/2006	DIABETIC FOOT CHECK	NORMAL/NEGATIVE		USER,POWER	
12/04/2006	AUDIOMETRIC SCREENING	ABNORMAL	test3	USER,POWER	doug's location
12/03/2006	AUDIOMETRIC THRESHOLD	NORMAL/NEGATIVE		UPSHAW,GERALDINE	Local Clinic
12/04/2005	BREAST EXAM	REFUSED SERVICE	new comment	USER,POWER	

Figure 18-1: Sample Exams

The Exams component permits viewing and documenting exams performed in the course of a visit. It stores data in the V EXAM file.

### 18.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	IHSBGOEXAMS.BGOEXAMS
Version	1.1.0.330
Class Identifier	{2B702069-50E1-4885-8492-1888FF0DB443}
Image File	IhsBgoExams.ocx
Property Initializations	none
Serializable Properties	HIDEBUTTONS=BOOL
Required Files	IhsBgoExams.chm
Security Keys	none
Multiple Instances Allowed	yes
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BGO*1.1*3

There are no specific implementation or maintenance tasks associated with this component.

### 18.3 Routine Descriptions

This component has been assigned the namespace designation of “BGOVEX”. The following routines are distributed:

Routine	Description
BGOVEXAM	Exam documentation support.

## 18.4 File List

None.

## 18.5 Cross References

None.

## 18.6 Exported Options

Option	Type	Description
BGOEXAM MAIN	menu	Exam configuration main menu.
BGO DISABLE EXAM EDITING	action	Disable editing of exam entries.

## 18.7 Exported Security Keys

None.

## 18.8 Exported Protocols

None.

## 18.9 Exported Parameters

Parameter	Instance Type	Value Type	Precedence	Description
BGO DISABLE EXAM EDITING		Boolean	User, Class	Disable editing of exam entries.

## 18.10 Exported Mail Groups

None.

## 18.11 Callable Routines

This section describes supported entry points for routines exported with this component.

### 18.11.1 RPC: BGOVEXAM DEL

Scope: private.

Parameter	Datatype	Description
INP	String	IEN* ^ Type where Type is "R" if refusal, otherwise null and IEN is the IEN of the refusal or the V EXAM file entry.
<return value>	String	Failure: -n^error text Success: null

Deletes a V EXAM or exam refusal entry.

### 18.11.2 RPC: BGOVEXAM GET

Scope: private.

Parameter	Datatype	Description
INP	Pointer (#2)	Patient's internal entry number.
<return value>	String List	Returned as a list of record in one of two formats:  For exams: E ^ Exam Name [2] ^ Visit Date [3] ^ Result [4] ^ Comment [5] ^ Provider Name [6] ^ Facility Name [7] ^ Provider IEN [8] ^ Location Name [9] ^ Exam IEN [10] ^ V File IEN [11] ^ Visit IEN [12] ^ Visit Category [13] ^ Visit Locked [14]  For refusals: R ^ Refusal IEN [2] ^ Type IEN [3] ^ Type Name [4] ^ Exam IEN [5] ^ Exam Name [6] ^ Provider IEN [7] ^ Provider Name [8] ^ Date [9] ^ Locked [10] ^ Reason [11] ^ Comment [12]

Returns a list of V EXAM and exam refusal records for the specified patient.

### 18.11.3 RPC: BGOVEXAM GETTYPES

Scope: private.

Parameter	Datatype	Description
<return value>	String List	Returned as a list of records in the format: Exam Type IEN ^ Exam Name ^ Exam Code ^ CPT Code

Returns a list of exam types from the EXAM (#9999999.15) file.

### 18.11.4 RPC: BGOVEXAM PRIPRV

Scope: private.

Parameter	Datatype	Description
VXAM	Pointer (#9000010.13)	IEN of the V EXAM entry.
<return value>	String	Returned as: Provider IEN ^ Provider Name ^ V Provider IEN

Returns the primary provider associated with the specified V EXAM entry.

### 18.11.5 RPC: BGOVEXAM SET

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: V Exam IEN (if edit) [1] ^ Exam IEN [2] ^ Visit IEN [3] ^ Provider IEN [4] ^ Result [5] ^ Comment [6] ^ Event Date [7] ^ Location IEN [8] ^ Other Location [9] ^ Historical Flag [10] ^ Patient IEN [11]
<return value>	String	Failure: -n^error text Success: V File IEN

Add/edit a V EXAM entry.

## 18.12 External Relations

Entity	Name	Description
File	V EXAM (#9000010.13)	Read and write access
File	PATIENT REFUSALS FOR SERVICE/NMI (#9000022)	Read and write access

## 18.13 Internal Relations

None.

## 18.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 18.15 Components

This component supports the following properties and methods:

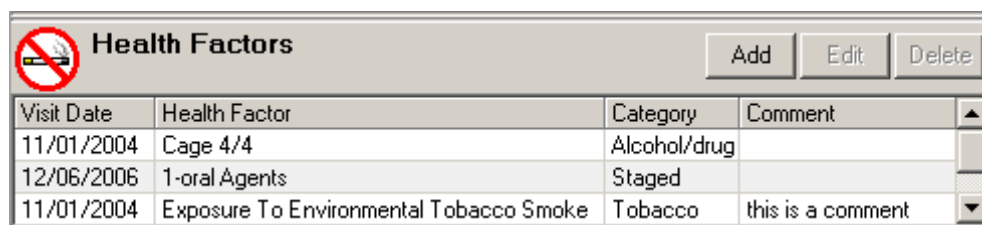
### 18.15.1 Properties

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom

<b>Property</b>	<b>Datatype</b>	<b>Access</b>	<b>Description</b>
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HIDEBUTTONS	Boolean	RW	If true, hides the button controls on the component.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 19.0 Health Factors

### 19.1 Introduction



Visit Date	Health Factor	Category	Comment
11/01/2004	Cage 4/4	Alcohol/drug	
12/06/2006	1-oral Agents	Staged	
11/01/2004	Exposure To Environmental Tobacco Smoke	Tobacco	this is a comment

Figure 19-1: Sample Health Factors

The Health Factors component permits management of entries in the V HEALTH FACTORS file.

### 19.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	IHSBGOHEALTHFACTORS.BGOHF
Version	1.1.0.265
Class Identifier	{62B454F8-BDEF-4534-B119-1F30ABE581EB}
Image File	IhsBgoHealthFactors.ocx
Property Initializations	none
Serializable Properties	HIDEBUTTONS=BOOL
Required Files	IhsBgoHealthFactors.chm
Security Keys	none
Multiple Instances Allowed	yes
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BGO*1.1*3

There are no specific implementation or maintenance tasks associated with this component.

### 19.3 Routine Descriptions

This component has been assigned the namespace designation of “BGOVHF.” The following routines are distributed:

Routine	Description
BGOVHF	Supports management of V HEALTH FACTOR entries.

## 19.4 File List

None.

## 19.5 Cross References

None.

## 19.6 Exported Options

Option	Type	Description
BGOHF MAIN	menu	Health Factor Configuration
BGO DISABLE HF EDITING	action	Disable Health Factor Editing

## 19.7 Exported Security Keys

None.

## 19.8 Exported Protocols

None.

## 19.9 Exported Parameters

Parameter	Instance Type	Value Type	Precedence	Description
BGO DISABLE HF EDITING		Boolean	User, Class	Disable editing of health factors.

## 19.10 Exported Mail Groups

None.

## 19.11 Callable Routines

This section describes supported entry points for routines exported with this component.

### 19.11.1 RPC: BGOVHF DEL

Scope: private.

Parameter	Datatype	Description
INP	Pointer (#9000010.23)	IEN of V HEALTH FACTORS entry.
<return value>	String	Failure: -n^error text Success: null



Deletes the specified V HEALTH FACTORS entry.

### 19.11.2 RPC: BGOVHF GET

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Patient IEN ^ Learn Only Flag ^ V HF IEN (optional)
<return value>	String List	Returned as a list of records in the format: Category [1] ^ HF Name [2] ^ Visit Date [3] ^ Severity [4] ^ Quantity [5] ^ Visit IEN [6] ^ V File IEN [7] ^ Health Factor Type [8] ^ Comment [9] ^ Visit Locked [10]

Return health factors associated with the specified patient. Optionally, restrict retrieval to a single health factor.

### 19.11.3 RPC: BGOVHF GETTYPES

Scope: private.

Parameter	Datatype	Description
INP	Integer	One of: All (default) Learning only
<return value>	String List	Returns a list of records in the format: Name [1] ^ Category Name [2] ^ Gender [3] ^ Type [4] ^ HF Type IEN [5] ^ Quantity Phrase [6] ^ Level Phrase [7]

Returns a list of records from the HEALTH FACTORS (#9999999.64) file.

### 19.11.4 RPC: BGOVHF REFLIST

Scope: private.

Parameter	Datatype	Description
INP	String	One of: PAP SMEAR, MAMMOGRAM, or EKG.
<return value>	String	Failure: -n^error text Success: IEN of specified entity

Looks up the specified entry to return the appropriate IEN for storage in the refusal file.

### 19.11.5 RPC: BGOVHF SET

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: HF Type IEN [1] ^ V File IEN [2] ^ Visit IEN [3] ^ Severity [4] ^
<return value>	String	Failure: -n^error text

Add/edit a V HEALTH FACTORS entry.

### 19.11.6 RPC: BGOVHF SETREF

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Refusal Type [1] ^ Item IEN [2] ^ Patient IEN [3] ^ Refusal Date [4] ^ Comment [5] ^ Provider IEN [6]
<return value>	String	Failure: -n^error text Success: null

Set a health factor refusal.

## 19.12 External Relations

Entity	Name	Description
File	V HEALTH FACTORS (#9000010.23)	Read and write access
File	PATIENT REFUSALS FOR SERVICE/NMI (#9000022)	Read and write access

### 19.13 Internal Relations

None.

### 19.14 Archiving and Purging

There are no archiving or purging requirements within this software.

### 19.15 Components

This component supports the following properties and methods:

## 19.15.1 Properties

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HIDEBUTTONS	Boolean	RW	If true, hides the button controls on the component.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 20.0 ICD Pick List

### 20.1 Introduction

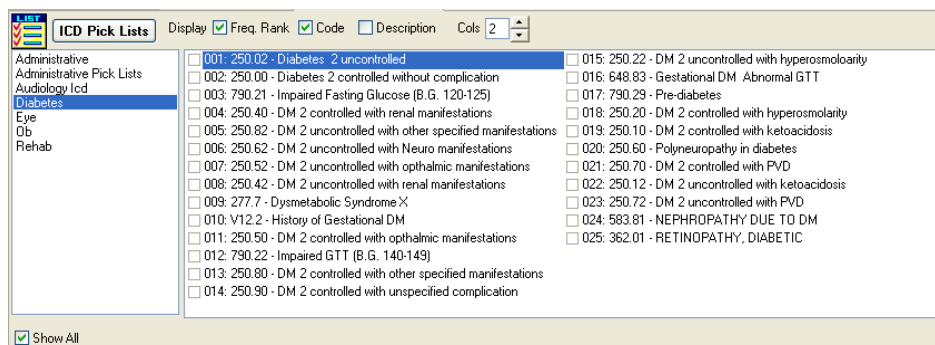


Figure 20-1: Sample ICD Pick List

The ICD Pick List component facilitates the documentation of diagnoses through the use of configurable pick lists.

### 20.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	IHSBGOICDPICKLIST.ICDPICKLIST
Version	1.1.0.237
Class Identifier	{3E10EFB2-4731-4BC1-A88B-F09FA2FB2432}
Image File	IhsBgoIcdPickList.ocx
Property Initializations	
Serializable Properties	RequireCACKey=BOOL
Required Files	IhsBgoIcdPickList.chm
Security Keys	none
Multiple Instances Allowed	yes
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BGO*1.1*3

There are no specific implementation or maintenance tasks associated with this component.

### 20.3 Routine Descriptions

This component has been assigned the namespace designation of “BGOICD.” The following routines are distributed:

Routine	Description
BGOICDLK	ICD9 code lookup.
BGOICDPR	ICD9 pick list support.
BGOICDP2	

## 20.4 File List

The following files are distributed:

### 20.4.1 BGO ICD PREFERENCES (#90362.35)

This file contains ICD9 pick list definitions.

Field Name	#	Datatype	Indexes	Description
NAME	.01	Text	B – Standard	Display name for the pick list.
HOSPITAL LOCATION	.02	Pointer (#44)	AH – Standard AHP – M	Hospital location associated with this pick list.
CLINIC	.03	Pointer (#40.7)	AC – Standard ACP – M	Clinic stop associated with this pick list.
PROVIDER	.04	Pointer (#200)	AP – Standard AHPTOO – M ACPTOO – M	Provider associated with this pick list.
OWNER	.05	Pointer (#200)		Owner of this pick list.
DISCIPLINE	.06	Pointer (#7)	AD - Standard	Provider class associated with this pick list.
ICD DIAGNOSIS	1	Subfile (#90362.351)		ICD9 codes associated with this pick list
MANAGERS	2	Subfile (#90362.352)		Managers associated with this pick list.

#### 20.4.1.1 ICD DIAGNOSIS subfile (#90362.351)

Field Name	#	Datatype	Indexes	Description
ICD DIAGNOSIS	.01	Pointer (#80)	B – Standard	ICD9 code associated with this pick list.
DISPLAY TEXT	.02	Text		Display text for this code.
FREQUENCY	.03	Integer	AC – Standard	Frequency count for this code.
DATE UPDATED	.04	Date/Time		Date/time this entry last updated.

#### 20.4.1.2 MANAGERS subfile (#90362.352)

Field Name	#	Datatype	Indexes	Description
MANAGERS	.01	Pointer (#200)	B – Standard	Managers associated with this pick list.

## 20.5 Cross References

Cross references are described in the preceding section.

## 20.6 Exported Options

None.

## 20.7 Exported Security Keys

None.

## 20.8 Exported Protocols

None.

## 20.9 Exported Parameters

None.

## 20.10 Exported Mail Groups

None.

## 20.11 Callable Routines

This section describes supported entry points for routines exported with this component.

### 20.11.1 RPC: BGOICDLK IC DLKUP

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Lookup Value [1] ^ Use Lexicon [2] ^ Visit Date [3] ^ Patient Gender [4] ^ ECodes [5] ^ VCodes [6]
<return value>	String List	Multiple records in the format: Descriptive Text^ICD IEN^Narrative Text^ICD Code

Lookup ICD9 text using either the Lexicon Utility or direct lookup.

### 20.11.2 RPC: BGOICDPR CLONE

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Source Category IEN ^ Target Category IEN
<return value>	String	Failure: -n^error text Success: null

Clone a pick list.

## 20.11.3 RPC: BGOICD CLONEOTH

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Provider IEN ^ ICD Preference IEN
<return value>	String	Failure: -n^error text Success: null

Clone a pick list from the VEN EHP ICD PREFERENCES (#19707.1) file.

## 20.11.4 RPC: BGOICDPR GETCATS

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Category IEN [1] ^ Hospital Location IEN [2] ^ Provider IEN [3] ^ Manager IEN [4] ^ Show All [5]
<return value>	String List	Returns a list of records in the format: Category Name [1] ^ Category IEN [2] ^ Hosp Loc Name [3] ^ Hosp Loc IEN [4] ^ Clinic Stop Name [5] ^ Clinic Stop IEN [6] ^ Provider Name [7] ^ Provider IEN [8] ^ Owner Name [9] ^ Owner IEN [10] ^ Provider Class Name [11] ^ Provider Class IEN [12]

Return pick lists matching specified criteria.

## 20.11.5 RPC: BGOICDPR GETITEMS

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Category IEN [1] ^ Group [2] ^ Visit IEN [3] ^ Display Freq Order [4]
<return value>	String List	Failure: -n^error text Success: List of records in the format ICD9 IEN [1] ^ ICD9 Code [2] ^ ICD9 Text [3] ^ Short Text [4] ^ Freq [5] ^ VPOV IEN [6] ^ Rank [7] ^ Pref IEN [8] ^ Long Text [9]

Returns a list of ICD9 entries associated with the specified pick list.

## 20.11.6 RPC: BGOICDPR GETLNAME

Scope: private.

Parameter	Datatype	Description
IEN	Pointer (#80)	IEN of ICD9 code
<return value>	String	Long text associated with the specified ICD9 entry.

Returns the long text associated with the specified ICD9 code.

### 20.11.7 RPC: BGOICDPR GETMGRS

Scope: private.

Parameter	Datatype	Description
CAT	Pointer (#90362.35)	IEN of pick list.
<return value>	String List	List of records in the format: Provider Name ^ Provider IEN

Returns a list of managers associated with the specified pick list.

### 20.11.8 RPC: BGOICDPR OTHCATS

Scope: private.

Parameter	Datatype	Description
<return value>	String List	List of records in the format: Provider Name ^ Provider IEN

Returns a list of PCC+ providers.

### 20.11.9 RPC: BGOICDPR QUERY

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Category IEN [1] ^ Provider IEN [2] ^ Clinic IEN [3] ^ Provider Class [4] ^ Hospital Location [5] ^ Start Date [6] ^ End Date [7] ^ Max Hits [8]
<return value>	String	Failure: -n^error text Success: null

Execute a query to update frequencies in a pick list.

### 20.11.10RPC: BGOICDPR SETCAT

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Category IEN [1] ^ ICD IEN [2] ^ Display Text [3] ^ Delete [4] ^ ICD Code [5] ^ Frequency [6] ^ Allow Dups [7] ^ Item IEN [8]
<return value>	String	Failure: -n^error text Success: IEN of category



Set field values for the specified pick list.

### 20.11.11 RPC: BGOICDPR SETFREQ

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Category IEN [1] ^ ICD IEN [2] ^ Increment [3] ^ Frequency [4]
<return value>	String	Failure: -n^error text Success: null

Update frequency for a pick list item.

### 20.11.12 RPC: BGOICDPR SETITEM

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Category IEN [1] ^ ICD IEN [2] ^ Display Text [3] ^ Delete [4] ^ ICD Code [5] ^ Frequency [6] ^ Allow Dups [7] ^ Item IEN [8]
<return value>	String	Failure: -n^error text Success: null

Set field values for a pick list item.

### 20.11.13 RPC: BGOICDPR SETMGR

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Category IEN [1] ^ Manager IEN [2] ^ Add [3]
<return value>	String	Failure: -n^error text Success: null

Add or remove a manager from a pick list.

### 20.11.14 RPC: BGOICDPR SETNAME

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Category IEN [1] ^ Item IEN [2] ^ Display Name [3]
<return value>	String	Failure: -n^error text Success: null

Set display name for a pick list item.

## 20.12 External Relations

Entity	Name	Description
File	VEN EHP ICD PREFERENCES (#19707.1)	Read access

## 20.13 Internal Relations

None.

## 20.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 20.15 Components

This component supports the following properties and methods:

### 20.15.1 Properties

The properties are described in the following table.

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
REQUIRECACKKEY	Boolean	RW	If true, the BGOZ CAC security key is required to edit pick lists.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 21.0 Immunizations

### 21.1 Introduction

The screenshot shows the 'Immunization Record' window. It has a title bar with a syringe icon and an information icon. The window is divided into several sections:

- Forecast:** A table with two rows: 'Td-ADULT due' and 'HEP A ADLT past due'.
- Contraindications:** A table with one row: 'CHOLERA Anaphylaxis 12-Jun-2007'. There are 'Add' and 'Delete' buttons.
- Vaccinations:** A section with buttons for 'Print Record', 'Due Letter', 'Profile', and 'Case Data'. To the right are 'Add', 'Edit', and 'Delete' buttons.
- Vaccination Table:** A table with columns: Vaccine, Visit Date, Age@Visit, Location, Reaction, Volume, Inj. Site, Lot, VIS Date, and Administered By. It contains four rows of data:
 

Vaccine	Visit Date	Age@Visit	Location	Reaction	Volume	Inj. Site	Lot	VIS Date	Administered By
Td-ADULT	09/25/1996	61 yrs	DEMO HOSPITAL						
HEP A	03/17/1995	59 yrs	Home						
HEP A	03/30/1995	59 yrs	Home						
HEP A	04/20/1995	60 yrs	Home						

Figure 21-1: Sample Immunization

The Immunizations component permits the documentation of immunizations, immunization refusals, and contraindications. It also provides an interface to the immunization forecaster.

### 21.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	IHSBGOIMMUNIZATION.BGOIMM
Version	1.1.0.579
Class Identifier	{631C61F1-FF11-467D-9391-EFA4694BA425}
Image File	IhsBgoImmunitization.ocx
Property Initializations	none
Serializable Properties	HIDEBUTTONS=BOOL
Required Files	IhsBgoImmunitization.chm
Security Keys	none
Multiple Instances Allowed	yes
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BGO*1.1*3

There are no specific implementation or maintenance tasks associated with this component.

### 21.3 Routine Descriptions

This component has been assigned the namespace designation of “BGOVIM”. The following routines are distributed:

Routine	Description
BGOVIMM BGOVIMM2	Immunization support

## 21.4 File List

None.

## 21.5 Cross References

None.

## 21.6 Exported Options

Option	Type	Description
BGOIMM MAIN	menu	Immunization configuration main menu.
BGO DISABLE IMM EDITING	action	Disable editing of immunizations.
BGO IMM STOP ADDING CPT CODES	action	Stop immunizations from adding CPT codes.
BGO IMM STOP ADDING ICD CODES	action	Stop immunizations from adding ICD codes.

## 21.7 Exported Security Keys

None.

## 21.8 Exported Protocols

None.

## 21.9 Exported Parameters

Parameter	Instance Type	Value Type	Precedence	Description
BGO DISABLE IMM EDITING		Boolean	User, Class	Disable editing of immunizations.
BGO IMM STOP ADDING CPT CODES		Boolean	User, Division, Package	Stop immunizations from adding CPT codes.
BGO IMM STOP ADDING ICD CODES		Boolean	User, Division, Package	Stop immunizations from adding ICD codes.

## 21.10 Exported Mail Groups

None.

## 21.11 Callable Routines

This section describes supported entry points for routines exported with this component.

### 21.11.1 RPC: BGOVIMM DEL

Scope: private.

Parameter	Datatype	Description
VIMM	Pointer (#9000010.11)	V IMMUNIZATION IEN
<return value>	String	Failure: -n^error text Success: null

Delete a V IMMUNIZATION entry.

### 21.11.2 RPC: BGOVIMM DELCONT

Scope: private.

Parameter	Datatype	Description
IEN	Pointer (#9002084.11)	BI PATIENT CONTRAINDICATIONS IEN
<return value>	String	Failure: -n^error text Success: null

Delete an immunization contraindication from the BI PATIENT CONTRAINDICATIONS (#9002084.11) file.

### 21.11.3 RPC: BGOVIMM GET

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#200)	Patient IEN

Parameter	Datatype	Description
<return value>	String List	<p>Returned as a list of records in one of the following formats:</p> <p>For an immunization:  I ^ Imm Name [2] ^ Visit Date [3] ^ V File IEN [4] ^  Other Location [5] ^ Group [6] ^ Imm IEN [7] ^ Lot [8] ^  Reaction [9] ^ VIS Date [10] ^ Age [11] ^ Visit Date [12] ^  Provider IEN~Name [13] ^ Inj Site [14] ^ Volume [15] ^  Visit IEN [16] ^ Visit Category [17] ^ Full Name [18] ^  Location IEN~Name [19] ^ Visit Locked [20]</p> <p>For an immunization forecast:  F ^ Imm Name [2] ^ Status [3]</p> <p>For a contraindication:  C ^ Contra IEN [2] ^ Imm Name [3] ^ Reason [4] ^ Date [5]</p> <p>For a refusal:  R ^ Refusal IEN [2] ^ Type IEN [3] ^ Type Name [4] ^  Item IEN [5] ^ Item Name [6] ^ Provider IEN [7] ^  Provider Name [8] ^ Date [9] ^ Locked [10] ^ Reason [11] ^</p>

Get immunization history for the specified patient. Returns immunizations, forecast information, contraindications, and refusals.

#### 21.11.4 RPC: BGOVIMM GETCASE

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#200)	Patient IEN
<return value>	String List	List of records containing requested case data.

Retrieves case data from the BI PATIENT (9002084) file.

#### 21.11.5 RPC: BGOVIMM GETCONT

Scope: private.

Parameter	Datatype	Description
<return value>	String List	List of records of the format: IEN ^ Name

Retrieves immunization contraindication reasons from the BI TABLE CONTRA REASON (#9002084.81) file.

## 21.11.6 RPC: BGOVIMM LOADIMM

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Patient IEN ^ Immunization Type IEN
<return value>	String List	Failure: -n^error text Success: First record: Default Lot # [1] ^ Default Volume [2] ^ Default VIS Date [3]  Subsequent records: Contraindication IEN [1] ^ Contraindication Text [2] ^ Date Noted [3]

Retrieve the patient's immunization defaults and contraindications.

## 21.11.7 RPC: BGOVIMM LOT

Scope: private.

Parameter	Datatype	Description
IMM	Pointer (#9999999.14)	IMMUNIZATION file IEN
<return value>	String List	Returned as a list of records in the format: Lot IEN ^ Name ^ Manufacturer

Retrieve lot numbers associated with a vaccine.

## 21.11.8 RPC: BGOVIMM PRINT

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Patient IEN [1] ^ Letter IEN [2] ^ Text of Date/Location Line [3] ^ Forecast Date [4]
<return value>	String List	Text of the immunization letter.

Return the specified immunization letter with all imbedded references fully resolved.

## 21.11.9 RPC: BGOVIMM PRIPRV

Scope: private.

Parameter	Datatype	Description
IEN	Pointer (#9000010.11)	V IMMUNIZATION IEN
<return value>	String	Returned as: Provider IEN ^ Provider Name ^ V Provider IEN

Returns the primary provider associated with the specified V IMMUNIZATION entry.

## 21.11.10 RPC: BGOVIMM PROFILE

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient IEN
<return value>	String List	Failure: -n^error text Success: Text of report

Returns the patient's immunization profile.

## 21.11.11 RPC: BGOVIMM SET

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Visit IEN [1] ^ Historical [2] ^ Patient IEN [3] ^ Imm IEN [4] ^ V File IEN [5] ^ Provider IEN [6] ^ Location [7] ^ Other Location [8] ^ Imm Date [9] ^ Lot # [10] ^ Reaction [11] ^ VIS Date [12] ^ Dose Override [13] ^ Inj Site [14] ^ Volume [15]
<return value>	String	Failure: -n^error text Success: V File IEN

Add/edit a V IMMUNIZATION entry.

## 21.11.12 RPC: BGOVIMM SETCONT

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Patient IEN [1] ^ Vaccine IEN [2] ^ Contraindication IEN [3] ^ Visit Date [4]
<return value>	String	Failure: -n^error text Success: Contraindication IEN

Add a contraindication.

## 21.11.13 RPC: BGOVIMM SETREG

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Patient IEN [1] ^ Case Manager IEN [2] ^ Parent [3] ^ Other Info [4] ^ Activate Flag [5] ^ Inactive Date [6] ^ Inactive Reason [7] ^ Tx Location [8] ^ Forecast Inf/Pneu [9] ^ Mother HBSAg Status [10]



Parameter	Datatype	Description
<return value>	String	Failure: -n^error text Success: null

Add/edit patient immunization registry entry.

## 21.12 External Relations

Entity	Name	Description
File	IMMUNIZATION (#9999999.14)	Read access
File	V IMMUNIZATION (#9000010.11)	Read and write access
File	BI PATIENT (#9002084)	Read access
File	BI SITE (#9002084.02)	Read access
File	BI PATIENT CONTRAINDICATIONS (#9002084.11)	Read and write access
File	BI TABLE CONTRA REASON (#9002084.81)	Read access
File	PATIENT REFUSALS FOR SERVICE/NMI (#9000022)	Read and write access
Package	IMMUNIZATION 8.1	Uses the following supported API calls: VER^BILOGO IMMFORC^BIRPC CONTRAS^BIRPC5 IMMHX^BIRPC ADDPAT^BIPATE IMMPROF^BIRPC BUILD^BILETPR1

## 21.13 Internal Relations

Entity	Name	Description
Component	V CPT	Supported API calls.
Component	V POV	Supported API calls.

## 21.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 21.15 Components

This component supports the following properties and methods:

### 21.15.1 Properties

The following table describes the properties.

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HIDEBUTTONS	Boolean	RW	If true, hides the button controls on the component.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
LOOKUPACTIVELOTSONLY	Boolean	RW	If true, limits selection of lot numbers to active lots only.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 22.0 Superbill

### 22.1 Introduction

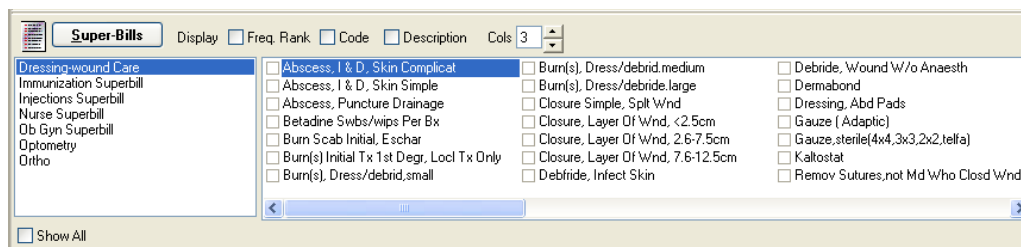


Figure 22-1: Sample Super-Bills

The SuperBill component permits documenting services delivered in the course of a patient encounter using configurable pick lists.

### 22.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Entity	Value
Programmatic Identifier	IHSBGOITEMS.BGOITEMS
Version	1.1.0.327
Class Identifier	{497CFBDB-7CCF-47CA-BB24-6B477C74EAC0}
Image File	IhsBgoItems.ocx
Property Initializations	none
Serializable Properties	RequireCACKey=BOOL
Required Files	IhsBgoItems.chm
Security Keys	none
Multiple Instances Allowed	yes
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BGO*1.1*3

There are no specific implementation or maintenance tasks associated with this component.

### 22.3 Routine Descriptions

This component has been assigned the namespace designation of “BGOCP.” The following routines are distributed:

Routine	Description
BGOCP2PR	CPT4 pick list support.
BGOCP2P2	

## 22.4 File List

### 22.4.1 BGO CPT PREFERENCES (#90362.31)

This file contains CPT4 pick list definitions.

Field Name	#	Datatype	Indexes	Description
NAME	.01	String	B – Standard	Display name for the pick list.
HOSPITAL LOCATION	.02	Pointer (#44)	AH – Standard AHP – M	Hospital location associated with this pick list.
CLINIC	.03	Pointer (#40.7)	AC – Standard ACP – M	Clinic stop associated with this pick list.
PROVIDER	.04	Pointer (#200)	AP – Standard AHPTOO – M ACPTOO - M	Provider associated with this pick list.
OWNER	.05	Pointer (#200)		Owner of this pick list.
DISCIPLINE	.06	Pointer (#7)	AD - Standard	Provider class associated with this pick list.
CPT	1	Subfile (#90362.312)		CPT4 codes associated with this pick list
MANAGERS	2	Subfile (#90362.313)		Managers associated with this pick list.
TYPE	7	Set		Type of pick list. One of: 0 = Visit services 1 = Historical services

#### 22.4.1.1 CPT subfile (#90362.312)

Field Name	#	Datatype	Indexes	Description
CPT	.01	Pointer (#81)	B – Standard	CPT4 code associated with this pick list.
DISPLAY TEXT	.02	String		Display text for this code.
FREQUENCY	.03	Integer	AC – Standard	Frequency count for this code.
MODIFIER 1ST	.04	Pointer (#9002274.07)		First billing modifier for this code.
MODIFIER 2ND	.05	Pointer (#9002274.07)		Second billing modifier for this code.
TRANSACTION CODE	.06	Pointer (#90092.02)		Transaction code associated with this item.
ASSOCIATIONS	1	Subfile (#90362.3121)		Additional associations with this pick list item.

**22.4.1.2 ASSOCIATIONS subfile (#90362.3121)**

Field Name	#	Datatype	Indexes	Description
ASSOCIATIONS	.01	Variable Pointer	B – Standard	Subitem associated with this pick list item. Can be from one of the following files: CPT (#81) CPT MODIFIER (#9999999.88) ICD DIAGNOSIS (#80) HEALTH FACTOR (#9999999.64) EDUCATION TOPIC (#9999999.09) EXAM (#9999999.15) IMMUNIZATION (#9999999.14) SKIN TEST (#9999999.28) TRANSACTION (#90092.02) ICD PROCEDURE (#80.1)
AUTOMATICALLY ADD	.02	Boolean		If true, automatically add this subitem when parent item is added.
DEFAULT TO ADD	.03	Boolean		If true, the default selected action will be to add the subitem.
PROHIBIT DUPLICATION	.04	Boolean		If true, this item should not be added if same type is already associated with the visit.
ASK CHARGE AMOUNT	.05	Boolean		Not current used.
ASK QUANTITY	.06	Boolean		Not currently used.
DEFAULT QUANTITY VALUE	.07	Integer		Default value to be used for quantity.

**22.4.1.3 MANAGERS subfile (#90362.313)**

Field Name	#	Datatype	Indexes	Description
MANAGERS	.01	Pointer (#200)	B – Standard	Managers associated with this pick list.

**22.5 Cross References**

Cross references are described in the preceding section.

**22.6 Exported Options**

None.

**22.7 Exported Security Keys**

None.

**22.8 Exported Protocols**

None.

## 22.9 Exported Parameters

None.

## 22.10 Exported Mail Groups

None.

## 22.11 Callable Routines

This section describes supported entry points for routines exported with this component.

### 22.11.1 RPC: BGOCTPR CLONE

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Pref IEN (from) ^ Pref IEN (to)
<return value>	String	Failure: -n^error text Success: null

Clones a pick list.

### 22.11.2 RPC: BGOCTPR CLONEOTH

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: CPT Category IEN ^ Preference Category IEN
<return value>	String	Failure: -n^error text Success: null

Converts a CPT category into a pick list.

### 22.11.3 RPC: BGOCTPR DELASSOC

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Category IEN ^ Item IEN ^ Element IEN
<return value>	String	Failure: -n^error text Success: null

Deletes an association.

## 22.11.4 RPC: BGOCTPR GETASSOC

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Category IEN ^ Item IEN
<return value>	String List	Returned as a list of records in the format: Item IEN [1] ^ Item Name [2] ^ Type [3] ^ Auto Add [4] ^ Auto Default [5] ^ No Dups [6] ^ Amount [7] ^ Association IEN [8] ^ Quantity [9] ^ Code [10] ^ Type ID [11]

Returns a list of associations for the specified pick list and item.

## 22.11.4.1 RPC: BGOCTPR GETCATS

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Category IEN [1] ^ Hospital Location IEN [2] ^ Provider IEN [3] ^ Manager IEN [4] ^ Show All [5] ^ Historical Flag [6]
<return value>	String List	Returns a list of records in the format: Category Name [1] ^ Category IEN [2] ^ Hosp Loc Name [3] ^ Hosp Loc IEN [4] ^ Clinic Stop Name [5] ^ Clinic Stop IEN [6] ^ Provider Name [7] ^ Provider IEN [8] ^ Owner Name [9] ^ Owner IEN [10] ^ Provider Class Name [11] ^ Provider Class IEN [12]

Returns a list of pick lists matching the specified criteria.

## 22.11.5 RPC: BGOCTPR GETITEMS

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as:  Category IEN [1] ^ Group [2] ^ Visit IEN [3] ^ Display Freq Order [4]
<return value>	String List	Failure: -n^error text Success: List of records in the format CPT IEN [1] ^ CPT Code [2] ^ CPT Text [3] ^ Short Text [4] ^ Freq [5] ^ VCPT IEN [6] ^ Fee [7] ^ Rank [8] ^ Pref IEN [9] ^

Returns a list of items for the specified pick list.

## 22.11.6 RPC: BGOCTPR GETLNAME

Scope: private.

Parameter	Datatype	Description
IEN	Pointer (#81)	IEN of CPT9 code
<return value>	String	Long narrative text.

Returns the long name for the specified CPT9 code.

## 22.11.7 RPC: BGOCTPR GETMGRS

Scope: private.

Parameter	Datatype	Description
CAT	Pointer (#90362,31)	IEN of pick list
<return value>	String List	List of records in the format: Provider Name ^ Provider IEN

Returns a list of managers associated with the specified pick list.

## 22.11.8 RPC: BGOCTPR OTHCATS

Scope: private.

Parameter	Datatype	Description
<return value>	String List	List of records in the format: Category Name [1] ^ Category IEN [2] ^ Class [3] where Class is one of: Med, Surg, Anesth, Rad, Lab

Returns a list of categories from the CPT file.

## 22.11.9 RPC: BGOCTPR QUERY

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Category IEN [1] ^ Provider IEN [2] ^ Clinic IEN [3] ^ Provider Class [4] ^ Hospital Location [5] ^ Start Date [6] ^ End Date [7] ^ Max Hits [8] ^ Med [9] ^ Surg [10] ^ Anest [11] ^ Lab [9] ^ Rad [12] ^ Supply [13] ^ 3rd Party Billing [14] ^ V CPT [15] ^ CHS [16]
<return value>	String	Failure: -n^error text Success: null

Executes a query to update pick list item frequencies.

## 22.11.10RPC: BGOCTPR SETACHK

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: CPT Preference IEN [1] ^ CPT Subfile IEN [2] ^ Associations Subfile IEN [3] ^ Column Index [4] ^ Value [5]
<return value>	String	Failure: -n^error text Success: null

Modifies a field for an association.



## 22.11.11 RPC: BGOCTPR SETASSOC

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: CPT Preference IEN [1] ^ CPT Subfile IEN [2] ^ Type [3] ^ Value [4] ^ Association [5] ^ Auto Add [6] ^ Default to Add [7] ^ No Dups [8] ^ Amount [9] ^ Quantity [10]
<return value>	String	Failure: -n^error text Success: IEN of association

Sets an association.

## 22.11.12 RPC: BGOCTPR SETCAT

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Name [1] ^ Hosp Loc [2] ^ Clinic [3] ^ Provider [4] ^ User [5] ^ Category IEN [6] ^ Delete [7] ^ Discipline [8]
<return value>	String	Failure: -n^error text Success: IEN of pick list

Sets field values for a pick list.

## 22.11.13 RPC: BGOCTPR SETFREQ

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Category IEN [1] ^ Item Value (defaults to all) [2] ^ Increment [3] ^ Frequency [4]
<return value>	String	Failure: -n^error text Success: null

Sets frequency for a pick list item.

## 22.11.14 RPC: BGOCTPR SETITEM

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Category IEN [1] ^ CPT IEN [2] ^ Display Text [3] ^ Delete [4] ^ CPT Code [5] ^ Frequency [6] ^ Allow Dups [7] ^ Item IEN [8]
<return value>	String	Failure: -n^error text Success: null

Sets field values for a pick list item.

## 22.11.15RPC: BGOCTPR SETMGR

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Category IEN [1] ^ Manager IEN [2] ^ Add [3]
<return value>	String	Failure: -n^error text Success: null

Adds or removes a manager from a pick list.

## 22.11.16RPC: BGOCTPR SETNAME

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Category IEN [1] ^ Item IEN [2] ^ Display Name [3]
<return value>	String	Failure: -n^error text Success: null

Sets the display name for a pick list item.

## 22.11.17RPC: BGOCTPR VSTASSOC

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Category IEN ^ Item IEN ^ Visit IEN
<return value>	String List	Returns a list of records in the format: Type ID [1] ^ Type Name [2] ^ Item IEN [3] ^ Item Text [4] ^ V File IEN [5] ^ V File # [6]

Returns all V file entries for a given visit that correspond to all associated entries for the given pick list.

## 22.12 External Relations

Entity	Name	Description
File	CPT (#81)	Read access
File	CPT MODIFIER (#9999999.88)	Read access
File	ICD DIAGNOSIS (#80)	Read access
File	HEALTH FACTOR (#9999999.64)	Read access
File	EDUCATION TOPIC (#9999999.09)	Read access

Entity	Name	Description
File	EXAM (#9999999.15)	Read access
File	IMMUNIZATION (#9999999.14)	Read access
File	SKIN TEST (#9999999.28)	Read access
File	TRANSACTION (#90092.02)	Read access
File	ICD PROCEDURE (#80.1)	Read access

## 22.13 Internal Relations

Entity	Name	Description
Component	VCPT	Calls supported APIs
Component	Visit Diagnosis	Calls supported APIs
Component	Health Factors	Calls supported APIs
Component	Patient Education	Calls supported APIs
Component	Exams	Calls supported APIs
Component	Immunization	Calls supported APIs
Component	Skin Test	Calls supported APIs

## 22.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 22.15 Components

This component supports the following properties and methods:

### 22.15.1 Properties

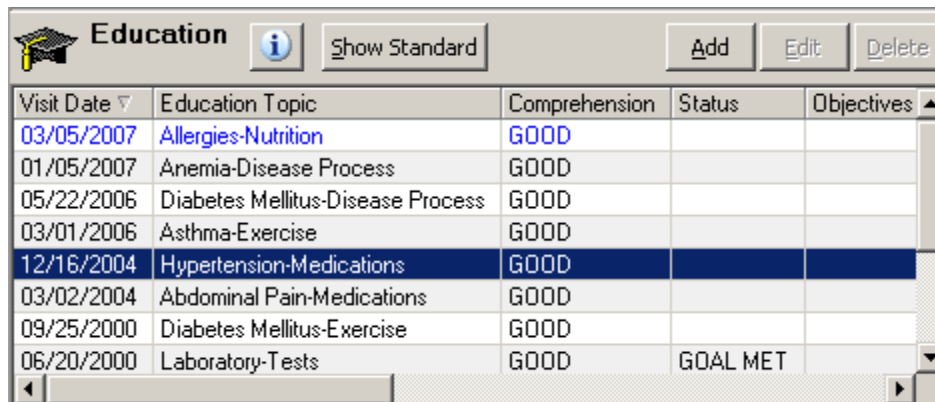
The following table describes the properties.

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent

Property	Datatype	Access	Description
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
REQUIRECACKEY	Boolean	RW	If true, the BGOZ CAC security key is required to edit pick lists.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 23.0 Patient Education

### 23.1 Introduction



The screenshot shows a software window titled "Education" with a graduation cap icon. It contains a table with columns: Visit Date, Education Topic, Comprehension, Status, and Objectives. The table lists several education sessions, with the entry for 12/16/2004 on Hypertension-Medications highlighted in blue. Buttons for "Add", "Edit", and "Delete" are visible at the top right, along with a "Show Standard" button and an information icon.

Visit Date	Education Topic	Comprehension	Status	Objectives
03/05/2007	Allergies-Nutrition	GOOD		
01/05/2007	Anemia-Disease Process	GOOD		
05/22/2006	Diabetes Mellitus-Disease Process	GOOD		
03/01/2006	Asthma-Exercise	GOOD		
12/16/2004	Hypertension-Medications	GOOD		
03/02/2004	Abdominal Pain-Medications	GOOD		
09/25/2000	Diabetes Mellitus-Exercise	GOOD		
06/20/2000	Laboratory-Tests	GOOD	GOAL MET	

Figure 23-1: Sample Education Component

The Patient Education component permits capturing patient education activities conducted during a patient encounter.

### 23.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	IHSBGOPATIENTED.BGOPATED
Version	1.1.0.447
Class Identifier	{CD0B03AE-F9AC-465C-A9FC-FAF42C8A5FB2}
Image File	IhsBgoPatientED.ocx
Property Initializations	none
Serializable Properties	HIDEBUTTONS=BOOL
Required Files	IhsBgoPatientEd.chm
Security Keys	none
Multiple Instances Allowed	yes
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BGO*1.1*3

There are no specific implementation or maintenance tasks associated with this component.

### 23.3 Routine Descriptions

This component has been assigned the namespace designations of “BGOVPED” and “BGOEDT.” The following routines are distributed:

<b>Routine</b>	<b>Description</b>
BGOVPED	Support for patient education.
BGOEDTPR	Support for patient education pick lists.
BGOEDTP2	

## 23.4 File List

The following files are distributed:

### 23.4.1 BGO ED TOPIC PREFERENCES (#90362.36)

This file contains education topic pick list definitions.

<b>Field Name</b>	<b>#</b>	<b>Datatype</b>	<b>Indexes</b>	<b>Description</b>
NAME	.01	Text	B – Standard	Display name for the pick list.
HOSPITAL LOCATION	.02	Pointer (#44)	AH – Standard AHP – M	Hospital location associated with this pick list.
CLINIC	.03	Pointer (#40.7)	AC – Standard ACP – M	Clinic stop associated with this pick list.
PROVIDER	.04	Pointer (#200)	AP – Standard AHPTOO – M ACPTOO – M	Provider associated with this pick list.
OWNER	.05	Pointer (#200)		Owner of this pick list.
DISCIPLINE	.06	Pointer (#7)	AD – Standard	Provider class associated with this pick list.
EDUCATION TOPIC	1	Subfile (#90362.361)		Education topics associated with this pick list
MANAGERS	2	Subfile (#90362.362)		Managers associated with this pick list.

#### 23.4.1.1 ICD DIAGNOSIS subfile (#90362.361)

<b>Field Name</b>	<b>#</b>	<b>Datatype</b>	<b>Indexes</b>	<b>Description</b>
EDUCATION TOPIC	.01	Pointer (#9999999.09)	B – Standard	Education topic associated with this pick list.
DISPLAY TEXT	.02	Text		Display text for this code.
FREQUENCY	.03	Integer	AC – Standard	Frequency count for this code.

#### 23.4.1.2 MANAGERS subfile (#90362.362)

<b>Field Name</b>	<b>#</b>	<b>Datatype</b>	<b>Indexes</b>	<b>Description</b>
MANAGERS	.01	Pointer (#200)	B – Standard	Managers associated with this pick list.

## 23.5 Cross References

Cross references are described in the preceding section.

## 23.6 Exported Options

Option	Type	Description
BGOPTED MAIN	menu	Patient education configuration menu.
BGO DISABLE PAT EDU EDITING	action	Disable patient education editing.

## 23.7 Exported Security Keys

None.

## 23.8 Exported Protocols

None.

## 23.9 Exported Parameters

Parameter	Instance Type	Value Type	Precedence	Description
BGO DISABLE PAT EDU EDITING		Boolean	User, Class	Disable patient education editing.

## 23.10 Exported Mail Groups

None.

## 23.11 Callable Routines

This section describes supported entry points for routines exported with this component.

### 23.11.1 RPC: BGOEDTPR CLONE

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Pref IEN (from) ^ Pref IEN (to)
<return value>	String	Failure: -n^error text Success: null

Clones a pick list.

### 23.11.2 RPC: BGOEDTPR GETCATS

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Category IEN [1] ^ Hospital Location IEN [2] ^ Provider IEN [3] ^ Manager IEN [4] ^ Show All [5]
<return value>	String List	Returns a list of records in the format: Category Name [1] ^ Category IEN [2] ^ Hosp Loc Name [3] ^ Hosp Loc IEN [4] ^ Clinic Stop Name [5] ^ Clinic Stop IEN [6] ^ Provider Name [7] ^ Provider IEN [8] ^ Owner Name [9] ^ Owner IEN [10] ^ Provider Class Name [11] ^ Provider Class IEN [12]

Returns a list of pick lists matching the specified criteria.

### 23.11.3 RPC: BGOEDTPR GETITEMS

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Category IEN [1] ^ Group [2] ^ Visit IEN [3] ^ Display Freq Order [4]
<return value>	String List	Returns a list of records in the format: Topic IEN [1] ^ Topic Text [2] ^ Freq [3] ^ VPED IEN [4] ^ Rank [5] ^ Item IEN [6]

Returns a list of education topics for the specified pick list.

### 23.11.4 RPC: BGOEDTPR GETLNAME

Scope: private.

Parameter	Datatype	Description
IEN	Pointer (#9999999.09)	Education topic IEN
<return value>	String	Long name for topic.

Returns a long name for an education topic.

### 23.11.5 RPC: BGOEDTPR GETMGRS

Scope: private.

Parameter	Datatype	Description
CAT	Pointer (#90362.36)	Pick list IEN
<return value>	String List	List of records in the format: Provider Name ^ Provider IEN

Returns a list of managers for the specified pick list.

### 23.11.6 RPC: BGOEDTPR QUERY

Scope: private.



Parameter	Datatype	Description
INP	String	Specified as: Category IEN [1] ^ Provider IEN [2] ^ Clinic IEN [3] ^ Provider Class [4] ^ Hospital Location [5] ^ Start Date [6] ^ End Date [7] ^ Max Hits [8]
<return value>	String	Failure: -n^error text Success: null

Executes a query to update frequencies for pick list items.

### 23.11.7 RPC: BGOEDTPR SETCAT

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Name [1] ^ Hosp Loc [2] ^ Clinic [3] ^ Provider [4] ^ User [5] ^ Category IEN [6] ^ Delete [7] ^ Discipline [8]
<return value>	String	Failure: -n^error text Success: IEN of category

Sets values for pick list fields.

### 23.11.8 RPC: BGOEDTPR SETFREQ

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Category IEN [1] ^ Item Value [2] (defaults to all) ^ Increment [3] ^ Frequency [4]
<return value>	String	Failure: -n^error text Success: null

Sets the frequency for a pick list item.

### 23.11.9 RPC: BGOEDTPR SETITEM

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Category IEN [1] ^ Education Topic IEN [2] ^ Display Text [3] ^ Delete [4] ^ Mnemonic [5] ^ Frequency [6] ^ Allow Dups [7] ^ Item IEN [8]
<return value>	String	Failure: -n^error text Success: null

Sets field values for a pick list item.

## 23.11.10 RPC: BGOEDTPR SETMGR

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Category IEN [1] ^ Manager IEN [2] ^ Add [3]
<return value>	String	Failure: -n^error text Success: null

Adds or removes a manager from a pick list.

## 23.11.11 RPC: BGOEDTPR SETNAME

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Category IEN [1] ^ Item IEN [2] ^ Display Name [3]
<return value>	String	Failure: -n^error text Success: null

Sets the display name for a pick list item.

## 23.11.12 RPC: BGOVPED DEL

Scope: private.

Parameter	Datatype	Description
VPED	Pointer (#9000010.16)	IEN of V PATIENT ED entry
<return value>	String	Failure: -n^error text Success: null

Deletes a patient education entry.

## 23.11.13 RPC: BGOVPED GET

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Patient IEN ^ Visit IEN If Visit IEN is not specified, all V File entries for the patient are returned. Otherwise, only entries for the specified visit are

Parameter	Datatype	Description
<return value>	String List	Returned as list of records in format: Topic Name [1] ^ Visit Date [2] ^ Level [3] ^ Provider Name [4] ^ Group/Individual [5] ^ Length [6] ^ CPT [7] ^ Comment [8] ^ Topic Category [9] ^ Behavior [10] ^ Objective Met [11] ^ Visit Locked [12] ^ Location Name [13] ^ VFile IEN [14] ^ Visit IEN [15] ^ Topic IEN [16] ^ Location IEN [17] ^ Provider IEN [18] ^ Visit Category [19] ^ ICD9 [20] ^

Gets patient education records associated with the specified visit or for all visits.

### 23.11.14 RPC: BGOVPED GETNAME

Scope: private.

Parameter	Datatype	Description
EDT	Pointer (#9999999.09)	IEN of EDUCATION TOPICS entry.
<return value>	String	Descriptive text. This will be the name of the associated major topic, ICD9 code, or CPT4 code, whichever is found first.

Gets descriptive text associated with education topic.

### 23.11.15 RPC: BGOVPED GETOS

Scope: private.

Parameter	Datatype	Description
EDT	Pointer (#9999999.09)	IEN of EDUCATION TOPICS entry.
<return value>	String List	Report text conveying outcome and standard guidelines.

Returns outcome and standard guidelines for an education topic.

### 23.11.16 RPC: BGOVPED GETTOPIC

Scope: private.

Parameter	Datatype	Description
<return value>	String List	Returned as a list of records in the format: IEN of PCC Education Topic ^ Name of PCC Education Topic

Returns a list of PCC education topics from the PCC DATA ENTRY EDUC TOPICS (#9001002.5) file.

### 23.11.17 RPC: BGOVPED GETTYPES

Scope: private.

Parameter	Datatype	Description
INP	Integer	One of: 0 = Category, 1 = Diagnosis, 2 = Non-diagnosis
<return value>	String List	Returned as a list of records in the format: Name [1] ^ Category Name [2] ^ ICD IEN [3] ^ ICD Name [4] ^ Education Topic IEN [5] ^ Type [6]

Returns a list of education topics from the EDUCATION TOPICS (#9999999.09) file.

### 23.11.18 RPC: BGOVPED PRIPRV

Scope: private.

Parameter	Datatype	Description
VPED	Pointer (#9000010.16)	IEN of V PATIENT ED entry
<return value>	String	Returned as: Provider IEN ^ Provider Name ^ V Provider IEN

Returns the primary provider associated with the specified V PATIENT ED entry.

### 23.11.19 RPC: BGOVPED SET

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: VFile IEN [1] ^ Topic [2] ^ Patient IEN [3] ^ Visit IEN [4] ^ Provider IEN [5] ^ Level Understanding [6] ^ Individual/Group [7] ^ Length [8] ^ CPT [9] ^ Comment [10] ^ Behavior Code [11] ^ Objectives [12] ^ Event Date [13] ^ Location IEN [14] ^ Other Location [15] ^ Historical [16] ^ Readiness [17]
<return value>	String	Failure: -n^error text Success: V File IEN

Adds or edits a V PATIENT ED entry.

### 23.11.20 RPC: BGOVPED SETDXTOP

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: ICD9 IEN ^ EDC IEN
<return value>	String	Failure: -n^error text Success: Education Topic IEN^Education Topic Name

Sets/returns diagnostic-based education topic.

### 23.11.21 RPC: BGOVPED SETPXTOP

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: CPT4 IEN ^ EDC IEN
<return value>	String	Failure: -n^error text Success: Education Topic IEN^Education Topic Name

Sets/returns procedure-based education topic.

## 23.12 External Relations

Entity	Name	Description
File	PCC DATA ENTRY EDUC TOPICS (#9001002.5)	Read access
File	EDUCATION TOPICS (#9999999.09)	Read and write access
File	V PATIENT ED (#9000010.16)	Read and write access

## 23.13 Internal Relations

Entity	Name	Description
Package	BGO COMPONENTS	Uses supported API calls.

## 23.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 23.15 Components

This component supports the following properties and methods:

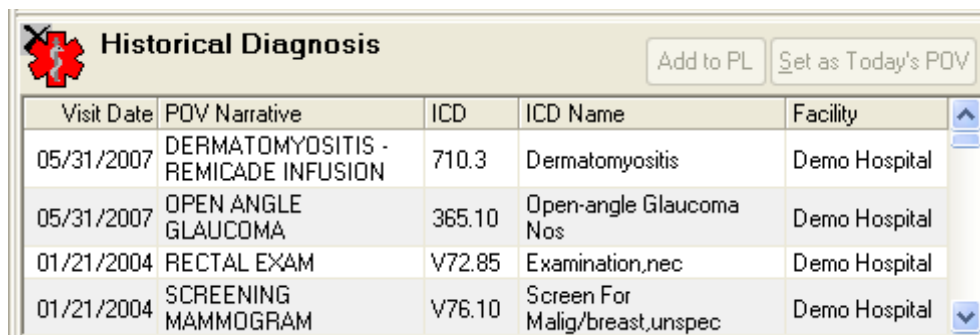
### 23.15.1 Properties

The properties are described in the following table.

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HIDEBUTTONS	Boolean	RW	If true, hides the button controls on the component.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
REQUIRECACKEY	Boolean	RW	If true, the BGOZ CAC security key is required to edit pick lists.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 24.0 POV History

### 24.1 Introduction



The screenshot shows a window titled "Historical Diagnosis" with a red cross icon. It contains a table with columns: Visit Date, POV Narrative, ICD, ICD Name, and Facility. There are also buttons for "Add to PL" and "Set as Today's POV".

Visit Date	POV Narrative	ICD	ICD Name	Facility
05/31/2007	DERMATOMYOSITIS - REMICADE INFUSION	710.3	Dermatomyositis	Demo Hospital
05/31/2007	OPEN ANGLE GLAUCOMA	365.10	Open-angle Glaucoma Nos	Demo Hospital
01/21/2004	RECTAL EXAM	V72.85	Examination,nec	Demo Hospital
01/21/2004	SCREENING MAMMOGRAM	V76.10	Screen For Malign/breast,unspec	Demo Hospital

Figure 24-1: Sample Historical Diagnosis

The POV History component permits the viewing of past purpose of visit entries.

### 24.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	IHSBGOPOVHISTORY.BGOPOVHISTORY
Version	1.1.0.194
Class Identifier	{AE82C827-D8CE-4B89-A1B7-6D93A2B5137D}
Image File	IhsBgoPovHistory.ocx
Property Initializations	none
Serializable Properties	none
Required Files	IhsBgoPovHistory.chm
Security Keys	none
Multiple Instances Allowed	yes
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BGO*1.1*3

There are no specific implementation or maintenance tasks associated with this component.

### 24.3 Routine Descriptions

This component shares routines with the Visit Diagnosis component. See that component for a description of routines.

### 24.4 File List

None.

## 24.5 Cross References

None.

## 24.6 Exported Options

Option	Type	Description
BGO POV HIST MAX ENTRIES	action	Set maximum entries shown in POV history.

## 24.7 Exported Security Keys

None.

## 24.8 Exported Protocols

None.

## 24.9 Exported Parameters

Parameter	Instance Type	Value Type	Precedence	Description
BGO POV HIST MAX ENTRIES		Integer	User, Class, Division, Package	The maximum number of entries (25-250) to display in POV History

## 24.10 Exported Mail Groups

None.

## 24.11 Callable Routines

This component shares routines with the Visit Diagnosis component. See that component for a description of callable routines.

## 24.12 External Relations

See Visit Diagnosis component.

## 24.13 Internal Relations

Entity	Name	Description
Component	Visit Diagnosis	Uses supported APIs.

## 24.14 Archiving and Purging

There are no archiving or purging requirements within this software.



## 24.15 Components

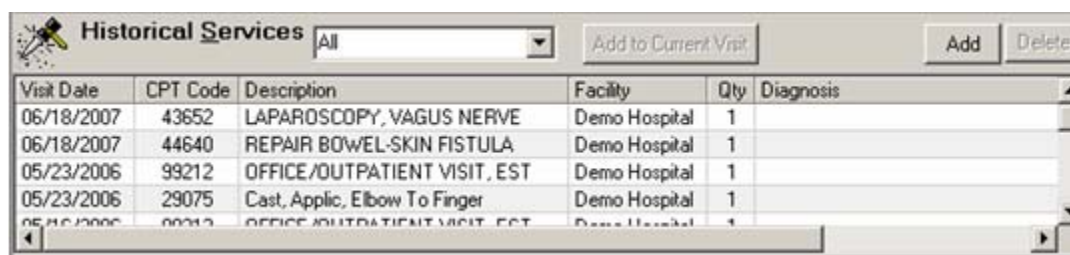
This component supports the following properties and methods:

### 24.15.1 Properties

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 25.0 Procedure Viewer

### 25.1 Introduction



The screenshot shows a window titled "Historical Services" with a dropdown menu set to "All". Below the menu are "Add to Current Visit", "Add", and "Delete" buttons. The main area contains a table with the following data:

Visit Date	CPT Code	Description	Facility	Qty	Diagnosis
06/18/2007	43652	LAPAROSCOPY, VAGUS NERVE	Demo Hospital	1	
06/18/2007	44640	REPAIR BOWEL-SKIN FISTULA	Demo Hospital	1	
05/23/2006	99212	OFFICE/OUTPATIENT VISIT, EST	Demo Hospital	1	
05/23/2006	29075	Cast, Applic, Elbow To Finger	Demo Hospital	1	
05/16/2006	99212	OFFICE/OUTPATIENT VISIT, EST	Demo Hospital	1	

Figure 25-1: Sample Historical Services

The Procedure Viewer component permits the viewing of past procedures recorded for a patient.

### 25.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	IHSBGPROCEDUREVIEWER.BGOPROCVIEW
Version	1.1.0.230
Class Identifier	{CB3C869B-09B3-417A-AA3C-F031A825DE6D}
Image File	IhsBgoProceduresViewer.ocx
Property Initializations	none
Serializable Properties	HIDEBUTTONS=BOOL
Required Files	IhsBgoProceduresViewer.chm
Security Keys	none
Multiple Instances Allowed	yes
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BGO*1.1*3

There are no specific implementation or maintenance tasks associated with this component.

### 25.3 Routine Descriptions

This component shares routines with the VCPT component. See that component for a description of routines.

### 25.4 File List

None.

## 25.5 Cross References

None.

## 25.6 Exported Options

None.

## 25.7 Exported Security Keys

None.

## 25.8 Exported Protocols

None.

## 25.9 Exported Parameters

None.

## 25.10 Exported Mail Groups

None.

## 25.11 Callable Routines

This component shares routines with the VCPT component. See that component for a description of callable routines.

## 25.12 External Relations

Entity	Name	Description
File	CPT (#81)	Read access
File	V CPT (#9000010.18)	Read access

## 25.13 Internal Relations

Entity	Name	Description
Component	VCPT	Uses supported API calls.

## 25.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 25.15 Components

This component supports the following properties and methods:

### 25.15.1 Properties

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HIDEBUTTONS	Boolean	RW	If true, hides the button controls on the component.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 26.0 Personal Health History

### 26.1 Introduction

Personal Health	
<b>Asthma Status</b>	06/14/2007: Severity=Mild Intermittent; Mgt. Plan=Yes
<b>Refusal</b>	06/14/2007: MAMMOGRAM BILAT (Radiology Exam)
<b>Reproductive</b>	GRAVIDA=1; PARA=0; LIVING CHILDREN=0; SPONT AB=0; TX AB=0; Pregnant: Est. Delivery=09/14/2007; Determined by=Clinical Parameters
<b>Tx Contract</b>	06/14/2007: Initiated=Jun 14, 2007; Type=Mental Health; Provider=User,Power

Figure 26-1: Sample Personal History

The Personal Health History component permits the documentation of several facets of a patient's personal health history including asthma status, functional status, refusals, reproductive history, infant feeding, and treatment contracts.

### 26.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	IHSBGOEPHISTORY.IHSBGOEPHISTCTRL
Version	1.1.0.408
Class Identifier	{9631FAD0-1ECF-4D7E-B61F-CF1E775D7203}
Image File	IhsBgoRepHist.ocx
Property Initializations	none
Serializable Properties	none
Required Files	IhsBgoRepHist.chm
Security Keys	none
Multiple Instances Allowed	yes
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BGO*1.1*3

There are no specific implementation or maintenance tasks associated with this component.

### 26.3 Routine Descriptions

The following routines are distributed:

Routine	Description
BGOBMSR	Birth measurements management
BGOREF	Refusal management

<b>Routine</b>	<b>Description</b>
BGOREP	Reproductive factors management
BGOVAST	Asthma management
BGOVELD	Elder care management
BGOVER	Emergency room visit tracking
BGOVIF	Infant feeding management
BGOVTXC	Treatment contract management

## 26.4 File List

None.

## 26.5 Cross References

None.

## 26.6 Exported Options

<b>Option</b>	<b>Type</b>	<b>Description</b>
BGOPHHX MAIN	menu	Personal health history configuration menu.
BGO DISABLE REP HIST EDITING	action	Disable reproductive history editing.

## 26.7 Exported Security Keys

<b>Key</b>	<b>Description</b>
BGOZ ASTHMA EDIT	Enable editing of V ASTHMA entries.
BGOZ ELDER CARE EDIT	Enable editing of V ELDER CARE entries.
BGOZ ER EDIT	Enable editing of V EMERGENCY VISIT RECORD entries.
BGOZ PEDIATRIC EDIT	Enable editing of BIRTH MEASUREMENT entries.
BGOZ REP HIST EDIT	Enable editing of REPRODUCTIVE FACTORS entries.
BGOZ TX CONTRACT EDIT	Enable editing of V TREATMENT CONTRACT entries.

## 26.8 Exported Protocols

None.

## 26.9 Exported Parameters

Parameter	Instance Type	Value Type	Precedence	Description
BGO DISABLE REP HIST EDITING		Boolean	User, Class	Disable editing of reproductive factors.

## 26.10 Exported Mail Groups

None.

## 26.11 Callable Routines

This section describes supported entry points for routines exported with this component.

### 26.11.1 RPC: BGOBMSR DEL

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's IEN.
<return value>	String	Failure: -n^error text Success: null

Deletes a patient's birth measurement record.

### 26.11.2 RPC: BGOBMSR GET

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's IEN.
<return value>	String	Returned as: Birth Weight Lbs [1] ^ Birth Weight Oz [2] ^ Birth Weight Kg [3] ^ Birth Weight Gms [4] ^ Apgar 1m [5] ^ Apgar 5m [6] ^ Gest Age Wks [7] ^ Delivery Type [8] ^ Complications [9] ^ Birth Order [10] ^ Formula Started [11] ^ Breast Stopped [12] ^ Solids Started [13] ^ Mother Name   IEN [14]

Returns birth measurement entries for patient.

### 26.11.3 RPC: BGOBMSR SET

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Patient IEN [1] ^ Weight [2] ^ Order [3] ^ Formula [4] ^ Breast [5] ^ Solids [6] ^ Mother [7]
<return value>	String	Failure: -n^error text Success: null

Sets birth measurement field values.

#### 26.11.4 RPC: BGOREF DEL

Scope: private.

Parameter	Datatype	Description
REFIEN	Pointer (#9000022)	IEN of refusal.
<return value>	String	Failure: -n^error text Success: null

Deletes a refusal.

#### 26.11.5 RPC: BGOREF GET

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Patient IEN ^ Refusal IEN (optional)
<return value>	String List	List of records in the format: R ^ Refusal IEN [2] ^ Type IEN [3] ^ Type Name [4] ^ Item IEN [5] ^ Item Name [6] ^ Provider IEN [7] ^ Provider Name [8] ^ Date [9] ^ Locked [10] ^ Reason [11] ^ Comment [12]

Gets refusal data for a specific refusal or for all refusals for a patient.

#### 26.11.6 RPC: BGOREF REFLIST

Scope: private.

Parameter	Datatype	Description
INP	String	One of: PAP SMEAR, MAMMOGRAM, or EKG.
<return value>	String	Failure: -n^error text Success: IEN of specified entity

Looks up the specified entry to return the appropriate IEN for storage in the refusal file.

#### 26.11.7 RPC: BGOREF SET

Scope: private.



Parameter	Datatype	Description
INP	String	Specified as: Refusal IEN [1] ^ Refusal Type [2] ^ Item IEN [3] ^ Patient IEN [4] ^ Refusal Date [6] ^ Comment [7] ^ Provider IEN [8] ^ Reason [9]
<return value>	String	Failure: -n^error text Success: null

Adds or edits a refusal.

### 26.11.8 RPC: BGOREP DEL

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's IEN.
<return value>	String	Failure: -n^error text Success: null

Deletes a patient's reproductive record.

### 26.11.9 RPC: BGOREP GET

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Patient IEN [1] ^ Date Obtained (opt) [2] ^ Expand History (opt) [3]
<return value>	String	Returned as: History Text [1] ^ LMP Date [2] ^ Contraception Method [3] ^ Contraception Begun [4] ^ How EDC Determined [5] ^ EDC Date [6]

Gets reproductive factors for a patient.

### 26.11.10 RPC: BGOREP SET

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Patient IEN [1] ^ Repro Hx [2] ^ LMP Date [3] ^ Contraceptive Method [4] ^ Contraception Begun [5] ^ Pregnant [6] ^ Delivery Date [7] ^ Delivery Method [8]
<return value>	String	Failure: -n^error text Success: null

Adds or edits reproductive factors.

### 26.11.11 RPC: BGOVAST DEL

Scope: private.

Parameter	Datatype	Description
VFIEN	Pointer (#9000010.41)	IEN of V ASTHMA entry.
<return value>	String	Failure: -n^error text Success: null

Deletes a V ASTHMA entry.

### 26.11.12 RPC: BGOVAST GET

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Patient IEN [1] ^ V File IEN [2] ^ Visit IEN [3]
<return value>	String List	List of records in the format: IEN [1] ^ Visit Locked [2] ^ Visit [3] ^ Severity [4] ^ FEV1 [5] ^ FEF25-75 [6] ^ PEF [7] ^ ETS [8] ^ Particulate Matter [9] ^ Dust Mite [10] ^ Management Plan [11] ^ Event Date [12] ^ Encounter Provider [13] where each field except IEN and Visit Locked has the form: External Value   Internal Value

Gets V ASTHMA entries by individual entry, visit, or patient.

### 26.11.13 RPC: BGOVAST GETNOTE

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's IEN.
<return value>	String List	Failure: -n^error text Success: Text of note

Fetches asthma registry note.

### 26.11.14 RPC: BGOVAST GETREG

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's IEN.
<return value>	String	Returned as: IEN [1] ^ Status [2] ^ Last Asthma Visit Date [3] ^ Calculated Date Due [4] ^ Next Scheduled Appt [5] ^ Case Manager [6] where each field except IEN has the form: External Value   Internal Value

Gets asthma registry entry.

## 26.11.15RPC: BGOVAST SET

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: V File IEN [1] ^ Visit IEN [2] ^ Severity [3] ^ FEV [4] ^ FEF [5] ^ PEF [6] ^ ETS [7] ^ Matter [8] ^ Mites [9] ^ Plan [10]
<return value>	String	Failure: -n^error text Success: V File IEN

Adds/edits V ASTHMA entry.

## 26.11.16 RPC: BGOVAST SETREG

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Patient IEN [1] ^ Status [2] ^ Last Visit Date [3] ^ Date Due [4] ^ Next Appt Date [5] ^ Case Manager [6] ^ Note [7]
<return value>	String	Failure: -n^error text Success: null

Adds/edits asthma registry entry.

## 26.11.17 RPC: BGOVELD DEL

Scope: private.

Parameter	Datatype	Description
VFIEN	Pointer (#9000010.35)	V File IEN
<return value>	String	Failure: -n^error text Success: null

Deletes a V ELDER CARE entry.

## 26.11.18 RPC: BGOVELD GET

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Patient IEN [1] ^ V File IEN [2] ^ Visit IEN [3]

Parameter	Datatype	Description
<return value>	String List	List of records in the format: IEN [1] ^ Visit Locked [2] ^ Visit [3] ^ Toileting [4] ^ Bathing [5] ^ Dressing [6] ^ Transfers [7] ^ Feeding [8] ^ Continence [9] ^ Finances [10] ^ Cooking [11] ^ Shopping [12] ^ Chores [13] ^ Medications [14] ^ Transportation [15] ^ Func Status Change [16] ^ Caregiver? [17] ^ Event Date [18] ^ Encounter Provider [19] where each field except IEN and Visit Locked has the form:

Gets elder care entries by individual entry, visit, or patient.

### 26.11.19 RPC: BGOVELD SET

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: V File IEN [1] ^ Visit IEN [2] ^ Toileting [3] ^ Bathing [4] ^ Dressing [5] ^ Transfers [6] ^ Feeding [7] ^ Continence [8] ^ Finances [9] ^ Cooking [10] ^ Shopping [11] ^ Chores [12] ^ Medications [13] ^ Transportation [14] ^ Func Status Change [15] ^ Caregiver [16]
<return value>	String	Failure: -n^error text Success: V File IEN

Adds or edits a V ELDER CARE entry.

### 26.11.20 RPC: BGOVIF DEL

Scope: private.

Parameter	Datatype	Description
VFIEN	Pointer (#9000010.44)	IEN of V INFANT FEEDING entry.
<return value>	String	Failure: -n^error text Success: null

Deletes a V INFANT FEEDING CHOICES entry.

### 26.11.21 RPC: BGOVIF GET

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Patient IEN [1] ^ V File IEN [2] ^ Visit IEN [3]
<return value>	String List	List of records in the format: IEN [1] ^ Visit Locked [2] ^ Visit [3] ^ Feeding Choice [4] ^ Event Date [5] ^ Encounter Provider [6] where each field except IEN and Visit Locked has the form: External Value   Internal Value

Gets a V INFANT FEEDING CHOICES entry.

### 26.11.22 RPC: BGOVIF SET

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: V File IEN ^ Visit IEN ^ Feeding Choice
<return value>	String	Failure: -n^error text Success: V File IEN

Adds or edits a V INFANT FEEDING CHOICES entry.

### 26.11.23 RPC: BGOVTXC DEL

Scope: private.

Parameter	Datatype	Description
VFIEN	Pointer (#9000010.39)	V File IEN
<return value>	String	Failure: -n^error text Success: null

Deletes a V TREATMENT CONTRACT entry.

### 26.11.24 RPC: BGOVTXC GET

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Patient IEN [1] ^ V File IEN [2] ^ Visit IEN [3]
<return value>	String List	List of records in the format: IEN [1] ^ Visit Locked [2] ^ Visit [3] ^ Contract Type [4] ^ Date Initiated [5] ^ Provider [6] ^ Event Date [7] ^ Encounter Provider [8] where each field except IEN and Visit Locked has the form: External Value   Internal Value

Returns treatment contract entries by individual entry, visit, or patient.

### 26.11.25 RPC: BGOVTXC SET

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: V File IEN [1] ^ Visit IEN [2] ^ Type [3] ^ Date [4] ^ Provider IEN [5]
<return value>	String	Failure: -n^error text Success: V File IEN

Adds or edits a V TREATMENT CONTRACT entry.

## 26.12 External Relations

<b>Entity</b>	<b>Name</b>	<b>Description</b>
File	V ASTHMA (#9000010.41)	Read and write access
File	V ELDER CARE (#9000010.35)	Read and write access
File	V INFANT FEEDING CHOICES (#9000010.44)	Read and write access
File	BIRTH MEASUREMENT (#9000024)	Read and write access
File	REPRODUCTIVE FACTORS #9000017)	Read and write access
File	V TREATMENT CONTRACT (#9000010.39)	Read and write access
File	PATIENT REFUSALS FOR SERVICE/NMI (#9000022)	Read and write access

## 26.13 Internal Relations

None.

## 26.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 26.15 Components

This component supports the following properties and methods:

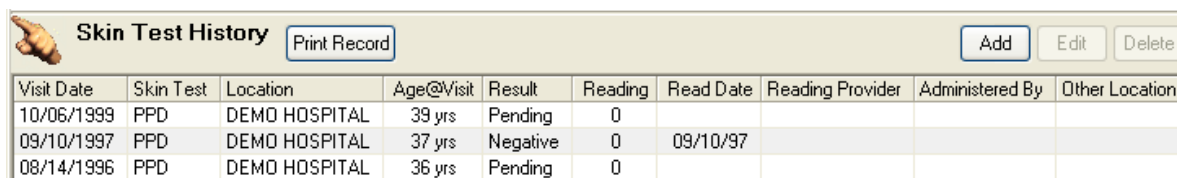
### 26.15.1 Properties

The properties are described in the following table.

<b>Property</b>	<b>Datatype</b>	<b>Access</b>	<b>Description</b>
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 27.0 Skin Tests

### 27.1 Introduction



Visit Date	Skin Test	Location	Age@Visit	Result	Reading	Read Date	Reading Provider	Administered By	Other Location
10/06/1999	PPD	DEMO HOSPITAL	39 yrs	Pending	0				
09/10/1997	PPD	DEMO HOSPITAL	37 yrs	Negative	0	09/10/97			
08/14/1996	PPD	DEMO HOSPITAL	36 yrs	Pending	0				

Figure 27-1: Sample Skin Test

The Skin Tests component facilitates documentation of the application and results of skin tests.

### 27.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	IHSBGOSKINTEST.IHSBGOSK
Version	1.1.0.604
Class Identifier	{9F9B1E7D-65C1-486B-A9BC-D806240B4ECA}
Image File	IhsBgoSkinTest.ocx
Property Initializations	none
Serializable Properties	HIDEBUTTONS=BOOL
Required Files	IhsBgoSkinTest.chm
Security Keys	none
Multiple Instances Allowed	yes
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BGO*1.1*3

There are no specific implementation or maintenance tasks associated with this component.

### 27.3 Routine Descriptions

This component has been assigned the namespace designation of “BGOVSK”. The following routines are distributed:

Routine	Description
BGOVSK	Skin test management

### 27.4 File List

None.



## 27.5 Cross References

None.

## 27.6 Exported Options

None.

## 27.7 Exported Security Keys

None.

## 27.8 Exported Protocols

None.

## 27.9 Exported Parameters

Parameter	Instance Type	Value Type	Precedence	Description
BGO DISABLE SK EDITING		Boolean	User, Class	Disable skin test editing.

## 27.10 Exported Mail Groups

None.

## 27.11 Callable Routines

This section describes supported entry points for routines exported with this component.

### 27.11.1 BGOVSK DEL

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: VSK ien ^ refusal ien
<return value>	String	Failure: -n^error text Success: null

Deletes a skin test or skin test refusal.

### 27.11.2 BGOVSK GET

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient IEN
<return value>	String List	List of records with one of two formats: For skin tests: S [1] ^ Visit Date [2] ^ VFile IEN [3] ^ Other Location [4] ^ Result [5] ^ Reading [6] ^ Date Read [7] ^ Test Name [8] ^ Test IEN [9] ^ Age [10] ^ Provider IEN~Name [11] ^ Reader IEN~Name [12] ^ Visit IEN [13] ^ Service Category [14] ^ Location IEN~Name [15] ^ Visit Locked [16] For refusals: R ^ Refusal IEN [2] ^ Type IEN [3] ^ Type Name [4] ^ Item IEN [5] ^ Item Name [6] ^ Provider IEN [7] ^ Provider Name [8] ^ Date [9] ^ Locked [10] ^ Reason [11] ^ Comment [12]

Retrieves skin tests and skin test refusals.

### 27.11.3 BGOVSK SET

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Visit IEN [1] ^ Historical [2] ^ Patient IEN [3] ^ Test IEN [4] ^ V File IEN [5] ^ Date Applied [6] ^ Location [7] ^ Other Location [8] ^ Result [9] ^ Reading [10] ^ Date Read [11] ^ Reader [12] ^ Provider [13]
<return value>	String	Failure: -n^error text Success: V File IEN

Adds or edits a V SKIN TEST entry.

## 27.12 External Relations

Entity	Name	Description
File	SKIN TEST (#9999999.28)	Read access
File	V SKIN TEST (#9000010.12)	Read and write access
Package	Immunization 8.1	Uses the following APIs: IMMHX^BIRPC

## 27.13 Internal Relations

Entity	Name	Description
Component	Immunization	Uses supported APIs.

## 27.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 27.15 Components

This component supports the following properties and methods:

## 27.15.1 Properties

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HIDEBUTTONS	Boolean	RW	If true, hides the button controls on the component.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 28.0 VCPT

### 28.1 Introduction

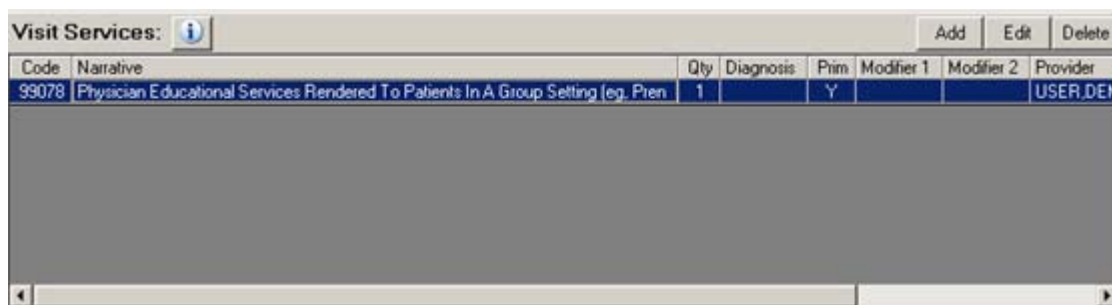


Figure 28-1: Sample Visit Services

The VCPT component permits recording services rendered during a specific patient encounter.

### 28.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	IHSBGOVCPT.BGOVCPT
Version	1.1.0.282
Class Identifier	{05B446C8-E08A-4227-8B23-0F14D2D7DB9D}
Image File	IhsBgoVCPT.ocx
Property Initializations	none
Serializable Properties	none
Required Files	IhsBgoVCPT.chm
Security Keys	none
Multiple Instances Allowed	no
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BGO*1.1*3

There are no specific implementation or maintenance tasks associated with this component.

### 28.3 Routine Descriptions

This component has been assigned the namespace designation of “BGOVCPT”. The following routines are distributed:

Routine	Description
BGOVCPT	V CPT maintenance
BGOVCPT2	

## 28.4 File List

None.

## 28.5 Cross References

None.

## 28.6 Exported Options

Option	Type	Description
BGOPROC MAIN	menu	Procedure configuration menu.
BGO DISABLE CPT EDITING	action	Disable CPT4 code editing.
BGO ENABLE CHARGEMASTER ENTRY	action	Enable support for ChargeMaster entry.
BGO ENABLE ICD PROCEDURE ENTRY	action	Enable support for ICD procedure entry

## 28.7 Exported Security Keys

Key	Description
BGOZ VCPT EDIT	Enable editing of V CPT entries.

## 28.8 Exported Protocols

None.

## 28.9 Exported Parameters

Parameter	Instance Type	Value Type	Precedence	Description
BGO DISABLE CPT EDITING		Boolean	User, Class	Disable CPT4 code editing.
BGO ENABLE CHARGEMASTER ENTRY		Boolean	Division, Package	Enable ChargeMaster data entry.
BGO ENABLE ICD PROCEDURE ENTRY		Boolean	Division, Package	Enable ICD procedure entry.

## 28.10 Exported Mail Groups

None.

## 28.11 Callable Routines

This section describes supported entry points for routines exported with this component.

## 28.11.1 RPC: BGOVCPT CPTLKUP

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Lookup Text [1] ^ Use Lexicon [2] ^ Date [3] ^ Exclude Med [4] ^ Exclude Surg [5] ^ Exclude HCPCS [6] ^ Exclude E&M [7] ^ Exclude Rad [8] ^ Exclude Lab [9] ^ Exclude Anesth [10] ^ Exclude Home [11]
<return value>	String List	List of CPT4 codes matching selection criteria in format: Description ^ CPT IEN ^ CPT Code ^ Narrative

0Looks up a CPT entry.

## 28.11.2 RPC: BGOVCPT DEL

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: V File IEN ^ Type where Type is one of: TRN = V Transaction Code PRC = V Procedure CPT = V CPT
<return value>	String	Failure: -n^error text Success: null

Deletes a V CPT, V Procedure, or V Transaction Code entry.

## 28.11.3 RPC: BGOVCPT GET

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Patient IEN [1] ^ Max [2] ^ Visit IEN [3] ^ Type [4] ^ Format [5] where Format is one of: 0 = Detailed 1 = TIU List
<return value>	String List	Returned as a list of records in the format: Visit Date [1] ^ Fac IEN [2] ^ Fac Name [3] ^ CPT [4] ^ CPT name [5] ^ Narrative [6] ^ Dx [7] ^ Prim [8] ^ Mod1 [9] ^ Mod2 [10] ^ VCPT IEN [11] ^  Visit IEN [12] ^ CPT IEN [13] ^ Quantity [14] ^ Provider Name [15] ^ Tran Code IEN [16] ^ ICD0 IEN [17] ^ Visit Locked [18]

Returns VCPT entries by patient or by visit.

## 28.11.4 RPC: BGOVCPT GETIEN

Scope: private.

Parameter	Datatype	Description
CPT	String	CPT code to lookup.
<return value>	Pointer (#81)	IEN of CPT code or null if not found.

Returns the IEN of a CPT4 code.

### 28.11.5 RPC: BGOVCPT IMMCK

Scope: private.

Parameter	Datatype	Description
CPTIEN	Pointer (#81)	IEN of CPT4 code.
<return value>	String	Returned as: Vaccine IEN ^ Short Name

Returns vaccine IEN associated with specified CPT IEN.

### 28.11.6 RPC: BGOVCPT MODLKUP

Scope: private.

Parameter	Datatype	Description
INP	String	CPT modifier to lookup.
<return value>	String List	Returns a list of matching records in the format: Modifier IEN [1] ^ Modifier Code [2] ^ Description [3]

Looks up a CPT modifier.

### 28.11.7 RPC: BGOVCPT SET

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Visit IEN [1] ^ CPT IEN [2] ^ Patient IEN [3] ^ Event Date [4] ^ Quantity [5] ^ Diagnosis [6] ^ Modifier #1 [7] ^ Provider IEN [8] ^ Principal [9] ^ V File IEN [10] ^ Narrative [11] ^ Modifier #2 [12] ^ Location IEN [13] ^ Outside Location [14] ^ Historical [15] ^ ICD Procedure Flag [16] ^ No Dups [17]
<return value>	String	Failure: -n^error text Success: VCPT IEN

Adds or edits a V CPT entry.

### 28.11.8 RPC: BGOVCPT SETDX

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: VCPT IEN ^ Diagnosis IEN

<return value>	String	Failure: -n^error text Success: null
----------------	--------	---

Sets a V CPT diagnosis.

### 28.11.9 RPC: BGOVCPT SETQTY

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: VCPT IEN ^ Quantity
<return value>	String	Failure: -n^error text Success: null

Sets a V CPT quantity.

## 28.12 External Relations

Entity	Name	Description
File	CPT (#81)	Read access
File	V CPT (#9000010.18)	Read and write access
File	CPT MODIFIER (#9999999.88)	Read access

## 28.13 Internal Relations

None.

## 28.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 28.15 Components

This component supports the following properties and methods:

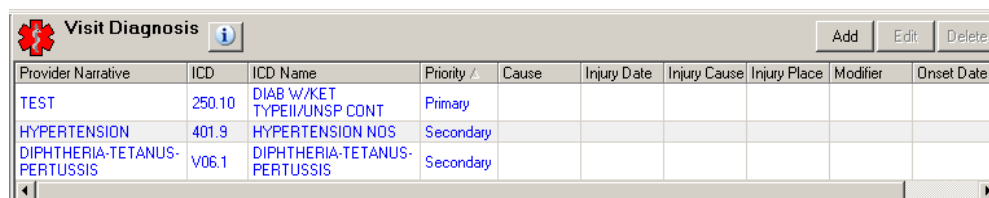


## 28.15.1 Properties

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HIDEBUTTONS	Boolean	RW	If true, hides the button controls on the component.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 29.0 Visit Diagnosis (VPOV)

### 29.1 Introduction



Provider Narrative	ICD	ICD Name	Priority	Cause	Injury Date	Injury Cause	Injury Place	Modifier	Onset Date
TEST	250.10	DIAB W/KET TYPEII/UNSP CONT	Primary						
HYPERTENSION	401.9	HYPERTENSION NOS	Secondary						
DIPHTHERIA-TETANUS- PERTUSSIS	V06.1	DIPHTHERIA-TETANUS- PERTUSSIS	Secondary						

Figure 29-1: Sample Visit Diagnosis

The Visit Diagnosis component permits the documentation of diagnoses pertaining to the current patient encounter.

### 29.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	IHSBGOVPOV.BGOVPOV
Version	1.1.0.279
Class Identifier	{629A3FFD-EA95-46D0-A73C-7334A947132F}
Image File	IhsBgoVPOV.ocx
Property Initializations	
Serializable Properties	HideButtons=BOOL, UseLexicon=BOOL
Required Files	IhsBgoVPOV.chm
Security Keys	none
Multiple Instances Allowed	no
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BGO*1.1*3

There are no specific implementation or maintenance tasks associated with this component.

### 29.3 Routine Descriptions

This component has been assigned the namespace designation of “BGOVPOV”. The following routines are distributed:

Routine	Description
BGOVPOV	V POV support
BGOVPOV1	

### 29.4 File List

None.

## 29.5 Cross References

None.

## 29.6 Exported Options

Option	Type	Description
BGOPOV MAIN	menu	POV configuration menu.
BGO DISABLE POV EDITING	action	Disable POV editing.

## 29.7 Exported Security Keys

Key	Description
BGOZ VPOV EDIT	Enable POV editing.

## 29.8 Exported Protocols

None.

## 29.9 Exported Parameters

Parameter	Instance Type	Value Type	Precedence	Description
BGO DISABLE POV EDITING		Boolean	User, Class	Disable POV editing.

## 29.10 Exported Mail Groups

None.

## 29.11 Callable Routines

This section describes supported entry points for routines exported with this component.

### 29.11.1 RPC: BGOVPOV CHECK

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: ICD IEN ^ Patient IEN ^ Visit Date
<return value>	String	Failure: -n^error text Success: null

Checks validity of an ICD9 code.

### 29.11.2 RPC: BGOVPOV CHRTREVV

Scope: private.

Parameter	Datatype	Description
VIEN	Pointer (#9000010)	Visit IEN
<return value>	String	Failure: -n^error text Success: V File IEN

Stores appropriate V code for a chart review visit.

### 29.11.3 RPC: BGOVPOV CKSIGNBY

Scope: private.

Parameter	Datatype	Description
VIEN	Pointer (#9000010)	Visit IEN
<return value>	String	Failure: -n^error text Success: IEN of Document

Checks for note signed by provider and associated with specified visit.

### 29.11.4 RPC: BGOVPOV DEL

Scope: private.

Parameter	Datatype	Description
VPOV	Pointer (#9000010.07)	IEN of V POV entry
<return value>	String	Failure: -n^error text Success: null

Deletes a V POV entry.

### 29.11.5 RPC: BGOVPOV GET

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Visit IEN [1] ^ VPOV IEN (optional) [2] ^ Format [3] where Format is one of:

Parameter	Datatype	Description
<return value>	String List	List of records in the format:  Format = 0 (detailed): V File IEN [1] ^ Visit Date (ext) [2] ^ Facility Code [3] ^ Facility Name [4] ^ ICD Code [5] ^ ICD Name [6] ^ Provider Narrative [7] ^ Modifier [8] ^ Onset Date (ext) [9] ^ Stage [10] ^ Revisit [11] ^ Cause [12] ^ Injury Date (ext) [13] ^ Cause E-code [14] ^ Place of Accident [15] ^ Primary [16] ^ ICD IEN [17] ^ Provider Name [18] ^ Visit IEN [19] ^ Visit Locked [20]  Format = 1 (single string): Concatenated string of POV's.  Format = 2 (multi-line): Provider narrative only in format:

Gets V POV entries associated with the specified visit.

### 29.11.6 RPC: BGOVPOV GETCODE

Scope: private.

Parameter	Datatype	Description
ICDIEN	Pointer (#80)	IEN of ICD9 code
<return value>	String	Returned as: ICD9 Code ^ Narrative

Returns ICD9 code information given its IEN.

### 29.11.7 RPC: BGOVPOV GETICD

Scope: private.

Parameter	Datatype	Description
ICD	String	Text to lookup
<return value>	String	Returns null if not found or ICD IEN ^ ICD Text

Looks up ICD9 code given text input.

### 29.11.8 RPC: BGOVPOV RECENT

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Patient IEN ^ Max Records ^ Visit IEN
<return value>	String List	List of records in the format: Visit Date [1] ^ Facility ID [2] ^ Facility Name [3] ^ ICD Code [4] ^ ICD Text [5] ^ Provider Narrative [6] ^ V POV IEN [7] ^ Visit IEN [8] ^ ICD IEN [9] ^ Visit Locked [10]

Returns recent POV's by patient or by visit.

## 29.11.9 RPC: BGOVPOV SET

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: VPOV IEN [1] ^ Visit IEN [2] ^ ICD Code IEN [3] ^ Patient IEN [4] ^ Narrative [5] ^ Stage [6] ^ Modifier [7] ^ Cause Dx [8] ^ First/Revisit [9] ^ Injury E-Code [10] ^ Injury Place [11] ^ Primary/Secondary [12] ^ Injury Date [13] ^ Onset Date [14] ^ Provider IEN [15]
<return value>	String	Failure: -n^error text Success: V File IEN

Adds or edits V POV data.

## 29.11.10 RPC: BGOVPOV SETPRI

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: VPOV IEN ^ Primary/Secondary (P/S)
<return value>	String	Failure: -n^error text Success: V File IEN

Sets primary/secondary status for a POV.

## 29.11.11 RPC: BGOVPOV TELEPHON

Scope: private.

Parameter	Datatype	Description
VIEN	Pointer (#9000010)	Visit IEN
<return value>	String	Failure: -n^error text Success: V File IEN

Stores appropriate V code for a telephonic visit.

## 29.12 External Relations

Entity	Name	Description
File	ICD DIAGNOSIS (#80)	Read access
File	V POV (#9000010.07)	Read and write access

## 29.13 Internal Relations

None.

## 29.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 29.15 Components

This component supports the following properties and methods:

### 29.15.1 Properties

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HIDEBUTTONS	Boolean	RW	If true, hides the button controls on the component.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
USELEXICON	Boolean	RW	If true, uses Lexicon Utility by default for lookups.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 30.0 Problem Management

### 30.1 Introduction

Problem List								
All Problems		Set as Today's POV			Add Edit Delete			
Provider Narrative	Status	Modified	Priority	Notes	Onset	ICD	ICD Name	
3 TYPE 2 DIABETES MELLITUS	Active	03/11/2000			03/11/2000	250.00	DIABETES II/UNSPEC NOT UNCONTR	
1 HYPERTENSION	Active	02/04/2000			01/19/1999	401.9	HYPERTENSION NOS	
2 TOBACCO USE	Active	02/04/2000			02/04/2000	305.1	TOBACCO USE DISORDER	

Figure 30-1: Sample Problem List

This Problem Management component facilitates the management of the patient's problem list.

### 30.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	IHSBGOPROBLEM.BGOPROBLEM
Version	1.1.0.296
Class Identifier	{8BAD7D20-FE2D-47D0-999D-F9996A17166B}
Image File	IhsBgoProblem.ocx
Property Initializations	none
Serializable Properties	HIDEBUTTONS=BOOL, USELEXICON=BOOL
Required Files	IhsBgoProblem.chm
Security Keys	none
Multiple Instances Allowed	yes
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BGO*1.1*3

There are no specific implementation or maintenance tasks associated with this component.

### 30.3 Routine Descriptions

This component has been assigned the namespace designation of "BGOPR". The following routines are distributed:

Routine	Description
BGOPROB	Problem list maintenance.
BGOPRBN	Problem list note maintenance.

### 30.4 File List

The following files are distributed:



### 30.4.1 BGO PROBLEM PRIORITY (#90362.22)

This file contains priority settings for problem list entries.

Field Name	#	Datatype	Indexes	Description
PROBLEM	.01	Pointer (#9000011)	B – Standard	Problem list entry associated with this priority.
PRIORITY	.02	Integer		Priority setting (0-5).

## 30.5 Cross References

Cross references are described in the preceding section.

## 30.6 Exported Options

Option	Type	Description
BGOPL MAIN	menu	Problem list configuration menu.
BGO DISABLE PROB LIST EDITING	action	Disable problem list editing.
BGO PL DEFAULT FILTER	action	Default filter for problem list.
BGO PL INCLUDE PERS HIST W ACT	action	Include personal history problem under active.

## 30.7 Exported Security Keys

Key	Description
BGOZ PROBLEM LIST EDIT	Enable problem list editing.

## 30.8 Exported Protocols

None.

## 30.9 Exported Parameters

Parameter	Instance Type	Value Type	Precedence	Description
BGO DISABLE PROB LIST EDITING		Boolean	User, Class	Disable problem list editing.
BGO PL DEFAULT FILTER		Set	User, Class, Division, System	Specifies which problem list filter should be the initial default. One of: 0 = All 1 = Active 2 = Inactive 3 = Personal History 4 = Family History
BGO PL INCLUDE PERS HIST W ACT		Boolean	User, Class, Division, Package	If true, problems marked as personal history are included under the active filter.

## 30.10 Exported Mail Groups

None.

## 30.11 Callable Routines

This section describes supported entry points for routines exported with this component.

### 30.11.1 BGOPRBN DEL

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Problem IEN [1] ^ Location IEN [2] ^ Note IEN [3]
<return value>	String	Failure: -n^error text Success: null

Deletes a problem note.

### 30.11.2 BGOPRBN GET

Scope: private.

Parameter	Datatype	Description
PRIEN	Pointer (#9000011)	IEN of PROBLEM entry
<return value>	String List	List of records in the format: Location IEN [1] ^ Note IEN [2] ^ Note # [3] ^ Narrative [4] ^ Status [5] ^ Date Added [6] ^ Author Name [7]

Returns all notes for a problem entry.

### 30.11.3 BGOPRBN SET

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Problem IEN [1] ^ Note IEN [2] ^ Location IEN [3] ^ Note # [4] ^ Narrative [5] ^ Status [6]
<return value>	String	Failure: -n^error text Success: Problem IEN [1] ^ Note IEN [2] ^ Location IEN [3] ^ Note # [4] ^ Narrative [5] ^ Status [6] ^ Date Entered [7] ^ Author Name [8] ^ Note ID [9]

Adds or edits a problem note.

### 30.11.4 BGOPROB CKID

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Patient IEN ^ Problem ID ^ Site IEN ^ Problem IEN (optional)
<return value>	String	Failure: -n^error text Success: If problem IEN is passed, returns 1 if id is same, 0 if different. If no problem IEN is passed, return value is null.

Checks for existence of problem id.

### 30.11.5 BGOPROB DEL

Scope: private.

Parameter	Datatype	Description
PRIEN	Pointer (#9000011)	IEN of PROBLEM entry
<return value>	String	Failure: -n^error text Success: null

Deletes a problem list entry.

### 30.11.6 BGOPROB GET

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient IEN
<return value>	String	List of records in the format: Number Code [1] ^ Patient IEN [2] ^ ICD Code [3] ^ Modify Date [4] ^ Class [5] ^ Provider Narrative [6] ^ Date Entered [7] ^ Status [8] ^ Date Onset [9] ^ Problem IEN [10] ^ Notes [11] ^ ICD9 IEN [12] ^ ICD9 Short Name [13] ^ Provider [14] ^ Facility IEN [15] ^ Priority [16]

Gets problem list entries for the specified patient.

### 30.11.7 BGOPROB NEXTID

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient IEN
<return value>	String	Next available problem ID.

Returns the next available problem id for the specified patient.

### 30.11.8 BGOPROB SET

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: ICD IEN or Code [1] ^ Narrative [2] ^ Location IEN [3] ^ Date of Onset [4] ^ Class [5] ^ Status [6] ^ Patient IEN [7] ^ Problem IEN [8] ^ Problem # [9] ^ Priority [10]
<return value>	String	Failure: -n^error text Success: Problem IEN

Adds or edits a problem list entry.

### 30.11.9 BGOPROB SETPRI

Scope: private.

Parameter	Datatype	Description
INP	String	Specified as: Problem IEN ^ Problem Priority
<return value>	String	Failure: -n^error text Success: BGO Problem Priority IEN

Sets priority for a problem list entry.

## 30.12 External Relations

Entity	Name	Description
File	PROBLEM (#9000011)	Read and write access
File	V POV (#9000010.07)	Read and write access

### 30.13 Internal Relations

None.

### 30.14 Archiving and Purging

There are no archiving or purging requirements within this software.

### 30.15 Components

This component supports the following properties and methods:

#### 30.15.1 Properties

The properties are described in the following table.

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ALLOWADDDPOV	Boolean	RW	If true, the user is permitted to add problem list entries to the V POV file.
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HIDEBUTTONS	Boolean	RW	If true, hides the button controls on the component.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
RUNASYNC	Boolean	RW	If true, problem list entries are fetched asynchronously. If false, entries are fetched synchronously.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
USELEXICON	Boolean	RW	If true, uses Lexicon Utility by default for lookups.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 31.0 Problems (CPRS)

### 31.1 Introduction

Stat/Ver	Description	Onset Date	Last Updated	Location
A	SINUSITIS		Sep 28 1998	
A	ALLERGY - TALWIN/CODIENE		Feb 16 2001	
A	RIGHT NASAL OBSTRUCTION		May 23 2001	
A	ATROPHIC RHINITIS		May 23 2001	
A	HYPERTENSION	Apr 27 2007	Apr 27 2007	
A	TEST	Apr 27 2007	Apr 27 2007	
A	OPEN ANGLE GLAUCOMA	May 31 2007	May 31 2007	
A	DIVERTICULOSIS	May 31 2007	May 31 2007	
A	HYPERTENSION	May 31 2007	May 31 2007	
A	SINUSITIS	May 31 2007	May 31 2007	
A	DERMATITIS	May 31 2007	May 31 2007	

Figure 31-1: Sample Problems

The Problems (CPRS) component facilitates the management of problem list entries. This component is provided for backward compatibility only. The Problem Management component provides more features and will eventually replace this component entirely.

### 31.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHPROBLEMS.PROBLEMS
Version	20.1.0.4
Class Identifier	{CB7F089E-AD83-4CE4-99FD-364B387C8266}
Image File	BEHProblems.ocx
Property Initializations	
Serializable Properties	AUTOSIZE=BOOL, BORDERSTYLE=ENUM, CAPTION=TEXT, CAPTIONCOLOR1=COLOR, CAPTIONCOLOR2=COLOR, CAPTIONSTYLE=ENUM, COLOR=COLOR
Required Files	BEHProblems.chm
Security Keys	none
Multiple Instances Allowed	no
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BEHO*1.1*036001

There are no specific implementation or maintenance tasks associated with this component.

### 31.3 Routine Descriptions

None.

### 31.4 File List

None.

### 31.5 Cross References

None.

### 31.6 Exported Options

None.

### 31.7 Exported Security Keys

None.

### 31.8 Exported Protocols

None.

### 31.9 Exported Parameters

None.

### 31.10 Exported Mail Groups

None.

### 31.11 Callable Routines

None.

### 31.12 External Relations

<b>Entity</b>	<b>Name</b>	<b>Description</b>
File	PROBLEM (#9000011)	Read and write access

## 31.13 Internal Relations

Entity	Name	Description
Component	CPRS Support Library	Uses supported APIs

## 31.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 31.15 Components

This component supports the following properties and methods:

### 31.15.1 Properties

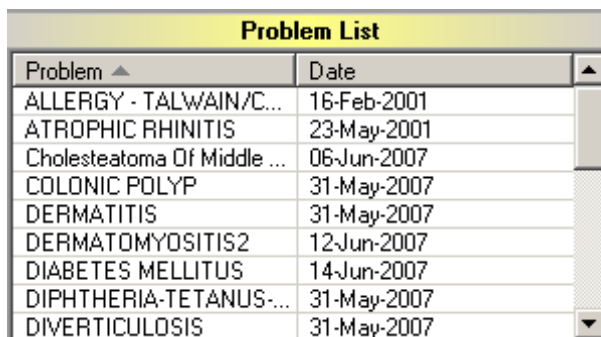
Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
AUTOSIZE	Boolean	RW	If true, the component automatically resizes itself to accommodate its contents.
BORDERSTYLE	Enum	RW	Sets the style of the border surrounding the component. Can be one of: 0 = None 1 = Single 2 = Sunken 3 = Raised
CAPTION	String	RW	Sets the text displayed in the title bar. To justify portions of the caption text, use the “\” character to delimit the left-, center-, and right-justified portions of the caption text.



Property	Datatype	Access	Description
CAPTIONCOLOR1 CAPTIONCOLOR2	Color	RW	Colors to apply to the title bar. If the two colors differ and a gradient style is set, a gradient effect is created. For a standard title bar style, only the first color is applied.
CAPTIONSTYLE	Enum	RW	Sets the caption style. Can be one of: 0 = None – No caption (hides title bar) 1 = Title – Standard title bar 2 = Frame – Framed title bar (group box style) 3 = Left – Left gradient title bar 4 = Right – Right gradient title bar 5 = Center – Center gradient title bar
COLOR	Color	RW	Sets the background color of the component.
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPPFILE	String	RW	Sets the name of the help file associated with the component.
LAYOUT	String	RW	Property representing the internal layout of the form.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 32.0 Problem List

### 32.1 Introduction



Problem List	
Problem ▲	Date ▲
ALLERGY - TALWAIN/C...	16-Feb-2001
ATROPHIC RHINITIS	23-May-2001
Cholesteatoma Of Middle ...	06-Jun-2007
COLONIC POLYP	31-May-2007
DERMATITIS	31-May-2007
DERMATOMYOSITIS2	12-Jun-2007
DIABETES MELLITUS	14-Jun-2007
DIPHThERIA-TETANUS...	31-May-2007
DIVERTICULOSIS	31-May-2007

Figure 32-1: Sample Problem List

The Problem List component is designed primarily for use on the cover sheet and provides a quick overview of active problems.

### 32.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHPROBLEMLIST.PROBLEMLIST
Version	4.2.0.7
Class Identifier	{3CE38F64-227D-46ED-A109-27073F032C64}
Image File	BEHProblemList.ocx
Property Initializations	none
Serializable Properties	ALLOWPRINT=BOOL, CAPTION=TEXT, CAPTIONCOLOR1=COLOR, CAPTIONCOLOR2=COLOR, CAPTIONSTYLE=ENUM, DEFERUPDATE=BOOL, DETAILPANE=BOOL, ORIENTATION=ENUM, LAYOUT=HIDDEN
Required Files	BEHProblemList.chm
Security Keys	none
Multiple Instances Allowed	yes
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BEHO*1.1*034001

There are no specific implementation or maintenance tasks associated with this component.

## 32.3 Routine Descriptions

This component has been assigned the namespace designation of “BEHOPL”. The following routines are distributed:

Routine	Description
BEHOPLCV	Problem list display support.

None.

## 32.4 Cross References

None.

## 32.5 Exported Options

None.

## 32.6 Exported Security Keys

None.

## 32.7 Exported Protocols

None.

## 32.8 Exported Parameters

None.

## 32.9 Exported Mail Groups

None.

## 32.10 Callable Routines

This section describes supported entry points for routines exported with this component.

### 32.10.1 RPC: BEHOPLCV DETAIL

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient IEN
IEN	Pointer (#9000011)	Problem IEN
<return value>	String List	Detail text

Returns detailed information about the specified problem entry.

## 32.10.2 RPC: BEHOPLCV LIST

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient IEN
STATUS	String	Status filter
<return value>	String List	List of problem records.

Returns a list of problem list entries filtered by the specified status.

## 32.11 External Relations

Entity	Name	Description
File	PROBLEM (#9000011)	Read access
Package	PROBLEM LIST 2.0	Uses the following API calls: LIST^GMPLUTL2 DETAIL^GMPLUTL2

## 32.12 Internal Relations

None.

## 32.13 Archiving and Purging

There are no archiving or purging requirements within this software.

## 32.14 Components

This component supports the following properties and methods:

### 32.14.1 Properties

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ALLOWPRINT	Boolean	RW	If true, a print button will appear on the detail dialog allowing printing of the contents.

Property	Datatype	Access	Description
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
AUTOSIZE	Boolean	RW	If true, the component automatically resizes itself to accommodate its contents.
BORDERSTYLE	Enum	RW	Sets the style of the border surrounding the component. Can be one of: 0 = None 1 = Single 2 = Sunken 3 = Raised
CAPTION	String	RW	Sets the text displayed in the title bar. To justify portions of the caption text, use the “\” character to delimit the left-, center-, and right-justified portions of the caption text.
CAPTIONCOLOR1 CAPTIONCOLOR2	Color	RW	Colors to apply to the title bar. If the two colors differ and a gradient style is set, a gradient effect is created. For a standard title bar style, only the first color is applied.
CAPTIONSTYLE	Enum	RW	Sets the caption style. Can be one of: 0 = None – No caption (hides title bar) 1 = Title – Standard title bar 2 = Frame – Framed title bar (group box style) 3 = Left – Left gradient title bar 4 = Right – Right gradient title bar 5 = Center – Center gradient title bar
COLOR	Color	RW	Sets the background color of the component.
DEFERUPDATE	Boolean	RW	If true, data refresh is deferred until the component becomes visible. If false, data refresh happens immediately.
DETAILPANE	Boolean	RW	If true, a detail pane appears next to the list view that displays detail text of the selected entry. If false, detail text appears in a popup dialog when an entry is clicked.
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPPFILE	String	RW	Sets the name of the help file associated with the component.
LAYOUT	String	RW	Property representing the internal layout of the form.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.

<b>Property</b>	<b>Datatype</b>	<b>Access</b>	<b>Description</b>
ORIENTATION	Enum	RW	Sets the orientation of the detail pane. Can be one of: 0 = Horizontal 1 = Vertical
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 33.0 Consults (CPRS)

### 33.1 Introduction

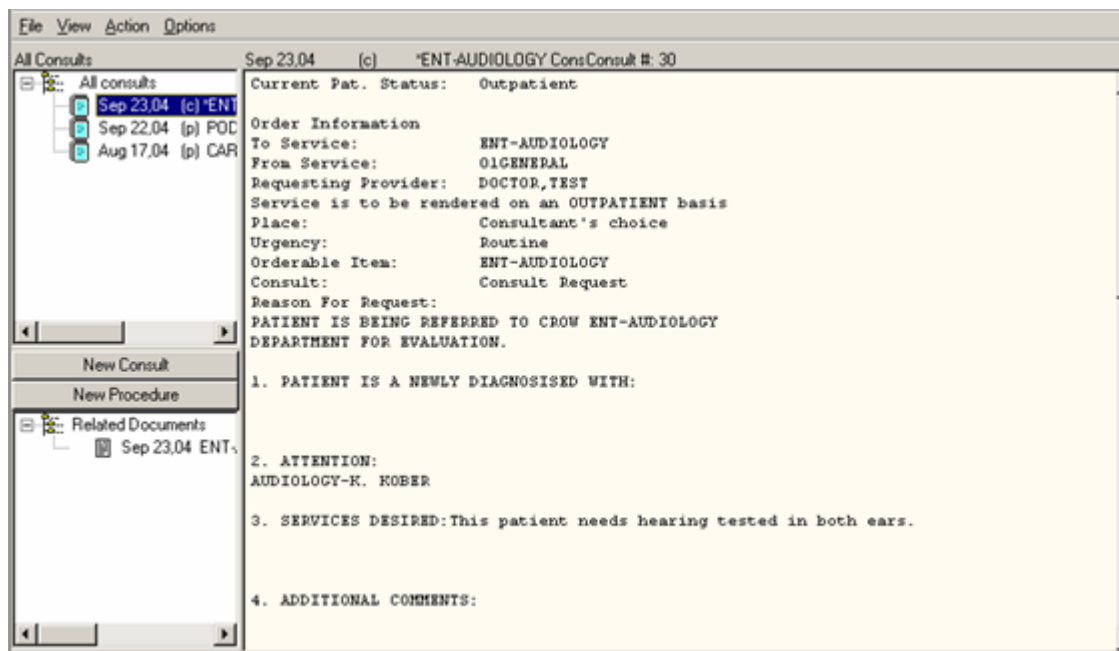


Figure 33-1: Sample Consults

The Consults (CPRS) component is a front end to the Consult Tracking package.

### 33.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHCONSULTS.CONSULTS
Version	20.1.0.18
Class Identifier	{48467F12-A353-43C8-A438-649925548B1A}
Image File	BEHConsults.ocx
Property Initializations	
Serializable Properties	AUTOSIZE=BOOL, BORDERSTYLE=ENUM, CAPTION=TEXT, CAPTIONCOLOR1=COLOR, CAPTIONCOLOR2=COLOR, CAPTIONSTYLE=ENUM, COLOR=COLOR
Required Files	BEHConsults.chm
Security Keys	none
Multiple Instances Allowed	no
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BEHO*1.1*017001

There are no specific implementation or maintenance tasks associated with this component.

### 33.3 Routine Descriptions

None.

### 33.4 File List

None.

### 33.5 Cross References

None.

### 33.6 Exported Options

Option	Type	Description
BEHOCN CLONE PROSTHETICS	action	This option allows the copying of one of the nationally exported Prosthetics services. These services have specific data in certain fields that are critical to the interface between Consult/Request Tracking and Prosthetics.
BEHOCN COMPLETION STATISTICS	action	Collect and display (via List Manager) the mean number of days that consults/requests take to be filled, from the time that they are accepted by the service to the time that they are entered as Complete in the computer. Also lists the Standard Deviation and number of consults that were found for a date range.
BEHOCN DUPLICATE SUB-SERVICE	action	This check if any Consult/Request Tracking REQUEST SERVICES (file #123.5) are Sub-Services of more than one Service.
BEHOCN IFC BKG PARAM MON	action	This option allows the viewing of the two parameters involved in the scheduling and running of the IFC background job. The parameters and their current setting is listed along with any information the system can tell from the settings. From this display the start parameter can be edited to a date/time in the future which will have the effect of delaying the next run of the background job until the time defined in this parameter.
BEHOCN IFC INC RPT	action	This option will print all incomplete IFC transactions sorted by Consult number.
BEHOCN IFC INC TRANS	action	This option is used to list and/or retransmit any and all Inter-facility Consult activities that did not complete successfully.



Option	Type	Description
BEHOCN IFC MGMT	menu	This menu contains a collection of options used to implement and manage the Inter-facility Consults interface.
BEHOCN IFC PARAMETER EDIT	action	This option allows the editing of parameters related to the processing of Inter-facility Consult requests.
BEHOCN IFC PRINT RPT NUMBERED	action	Sends the Inter-facility Consult Requests report to a device without going through the List Manager display of the report and waiting for the list to be built.
BEHOCN IFC REMOTE NUMBER	action	This option allows the look-up of a local consult/request when only the remote site and remote consult number are known. Upon look-up, the option includes both a brief and detailed display of the request.
BEHOCN IFC RPT CONSULTS	action	This option provides detailed information regarding inter-facility consults. If not specified, all status' are utilized. "Routing Facility" data: If selecting requesting site information, this refers to the facility the consult is sent to; if selecting consulting site information, this refers to the facility the consult is sent from. "Days Diff" data: This refers to the (whole number) of days difference between the COMPLETE activity date/time and one of the following activity date/time (if one doesn't exist, retrieve the next): RECEIVED, ENTERED IN CPRS, SERVICE ENTERED, FILE ENTRY DATE. All date/time data must be in the range selected. Field REMOTE ACTIVITY TIME ZONE is NOT figured into the calculations. "Rec Date" data: For IFC consulting site report requests only, the option checks for DATE/TIME OF ACTION ENTRY for activity of REMOTE REQUEST RECEIVED, and if that doesn't exist it checks for DATE/TIME OF ACTION ENTRY for the activity of RECEIVED. Only date is listed in the report. "Mean Days Completed To Service <service> @ <facility>" data: For only those entries in which Days Diff exists, this calculation for each service at each routing facility is: Days Diff total divided by Number of Days Diff values. "Mean Days Completed To Service <service>" data: For only those entries in which Days Diff exists, this calculation for each service is: Days Diff total divided by Number of Days Diff values.
BEHOCN IFC RPT CONSULTS BY PT	action	This option provides actions which a service can use to track inter-facility consults and requests. The processing flow is similar to the OE/RR review screen. Most of the actions have the same mnemonics as OE/RR for clinician familiarity.
BEHOCN IFC RPT CONSULTS BY REM	action	This option provides detailed information regarding inter-facility consults by remote ordering provider for consulting sites to utilize.

Option	Type	Description
BEHOCN IFC TEST PT MPI	action	WARNING: This option is for use in test accounts ONLY! This option will not function in a production VistA environment. This option will use the social security number of the patient and create a pseudo-ICN for the patient selected. If the patient name and social security number are the same at the local facility and the IFC partnering facility, the selected patient can be used for IFC testing after this option is executed on both systems.
BEHOCN IFC TEST SETUP	action	This option can be used to test the IFC setup for a given procedure or consult service. After selection of the service or procedure, the remote site will be contacted and the procedure or service name will be confirmed as correct. Any errors encountered will be listed.
BEHOCN IFC TRANS	action	This option is used to list any or all Inter-facility Consult entries from the IFC MESSAGE LOG file (#123.6). NOTE: The parameter GMRC RETAIN IFC ACTIVITY DAYS controls the number of days that completed Inter-facility Consult transactions are maintained in the IFC MESSAGE LOG (#123.6) file.
BEHOCN LIST HIERARCHY	action	This option will allow the Consult Service hierarchy to be printed. The listing will include all disabled and tracking only services. Any services in the REQUEST SERVICES (#123.5) file that are not part of the hierarchy will be listed at the end.
BEHOCN MAIN	menu	Contains utilities which aid in the management of consultation requests site setup and maintenance.
BEHOCN NOTIFICATION	action	Determine Notification Recipients for a Service
BEHOCN PHARMACY TPN	action	Pharmacy TPN Consults
BEHOCN PRINT BY SEARCH	action	This option will compile a list of consult requests based on Provider, Location, or Procedure search criteria. Date range, status, and output formats are selectable parameters. There are additional prompts for selection of local, remote, or both local and remote Provider(s) and Location(s). For the purposes of this option, "local" refers to non-Inter-facility requests and Inter-facility requests originating locally; "remote" only refers to Inter-facility requests originating at another site.
BEHOCN PRINT COMPLETION STAT	action	Sends the Completion Time Statistics report to a device without going through the List Manager display of the report and waiting for the list to be built.
BEHOCN PRINT RPT NUMBERED	action	Sends the Service Consults By Status report to a device without going through the List Manager display of the report and waiting for the list to be built.

Option	Type	Description
BEHOCN PRINT TEST PAGE	action	This option allows a user to select a printer for reviewing print parameters. A report is generated which includes the current device print parameters and a ruler to verify the length of the current print page. A description of how to use the ruler is included in the report.
BEHOCN PROCEDURE	action	Setup Procedures
BEHOCN REPORTS	menu	Consult Tracking Reports
BEHOCN RPT COMPLETE	action	Service Consults Completed
BEHOCN RPT COMPLETE/	action	Service Consults Completed or Pending Resolution
BEHOCN RPT CONSULTS BY	action	Service Consults By Status
BEHOCN RPT NUMBERED	action	Service Consults with Consults Numbers
BEHOCN RPT PENDING	action	Service Consults Pending Resolution
BEHOCN SERVICE TRACKING	action	This option provides actions which a service can use to track consults and requests. The processing flow is similiar to the OE/RR review screen. Most of the actions have the same mnemonics as OE/RR for clinician familiarity.
BEHOCN SERVICE USER MGMT	edit	This option is used to identify individuals or teams which should be notified when a Consult/Request is being sent to their receiving service. It is also used to identify individuals who will not be notified when consults are being sent to the service, but DO have update/tracking capabilities for the service. Individuals or teams can also be notified of a new consult based on the Patient's Location. When a Consult/Request is sent to a service, the Patient Location will be checked. If the receiving service has broken down their service notification by Hospital location, the notification will be sent to the individual and/or a team defined in this option.
BEHOCN SETUP REQUEST SERVICES	action	This option is used to setup the Hospital hierarchy of services and specialties. Each service/specialty defined in this file can be setup to have a consult print at its own service printer when the consult is entered and signed using the "Add orders" menus in OE/RR. Teams of clinicians can also be associated with each Service/Specialty. The team members will be notified when a new consult is ordered from their service/specialty.

Option	Type	Description
BEHOCN STSU	action	This option will allow a search of the REQUEST/CONSULTATION (#123) file for requests to a particular service within a date range. The status of the entries meeting the selected criteria can be updated to COMPLETE or DISCONTINUED. A comment of up to 256 characters can be entered once and attached to each of the entries. A report of those records to be updated can be printed without performing the status updates. The option also allows printing of the records along with the status updates. The status of the corresponding order in CPRS will be updated to match that in file 123. Only the user entered in the SPECIAL UPDATES INDIVIDUAL field #123.5 of the REQUEST SERVICES (#123.5) file will be allowed to perform the updates for the chosen service.
BEHOCN TEST DEFAULT REASON	action	This option will allow the Clinical Coordinator or IRM to test the default reason for request for a given service or procedure without placing an order. This option prompts for the selection of a procedure or a service from the consult hierarchy and a patient name. The default reason will be presented based on the selected patient in a list manager format. This format allows searching, printing etc.
BEHOCN UPDATE AUTHORITY	action	This option will determine a selected user's update authority for a selected service in the consult hierarchy.
BEHOCN USER NOTIFICATION	action	This option will determine if a user would be a notification recipient for a selected service.

### 33.7 Exported Security Keys

None.

### 33.8 Exported Protocols

None.

### 33.9 Exported Parameters

None.

### 33.10 Exported Mail Groups

None.

### 33.11 Callable Routines

None.

## 33.12 External Relations

Entity	Name	Description
Package	CONSULT/REQUEST TRACKING 3.0	Uses supported APIs.

## 33.13 Internal Relations

Entity	Name	Description
Component	CPRS Support Library	Uses supported APIs.

## 33.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 33.15 Components

This component supports the following properties and methods:

### 33.15.1 Properties

The properties are described in the following table.

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
AUTOSIZE	Boolean	RW	If true, the component automatically resizes itself to accommodate its contents.
BORDERSTYLE	Enum	RW	Sets the style of the border surrounding the component. Can be one of: 0 = None 1 = Single 2 = Sunken 3 = Raised

Property	Datatype	Access	Description
CAPTION	String	RW	Sets the text displayed in the title bar. To justify portions of the caption text, use the “\” character to delimit the left-, center-, and right-justified portions of the caption text.
CAPTIONCOLOR1 CAPTIONCOLOR2	Color	RW	Colors to apply to the title bar. If the two colors differ and a gradient style is set, a gradient effect is created. For a standard title bar style, only the first color is applied.
CAPTIONSTYLE	Enum	RW	Sets the caption style. Can be one of: 0 = None – No caption (hides title bar) 1 = Title – Standard title bar 2 = Frame – Framed title bar (group box style) 3 = Left – Left gradient title bar 4 = Right – Right gradient title bar 5 = Center – Center gradient title bar
COLOR	Color	RW	Sets the background color of the component.
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPPFILE	String	RW	Sets the name of the help file associated with the
LAYOUT	String	RW	Property representing the internal layout of the form.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 34.0 Discharge Summary (CPRS)

### 34.1 Introduction

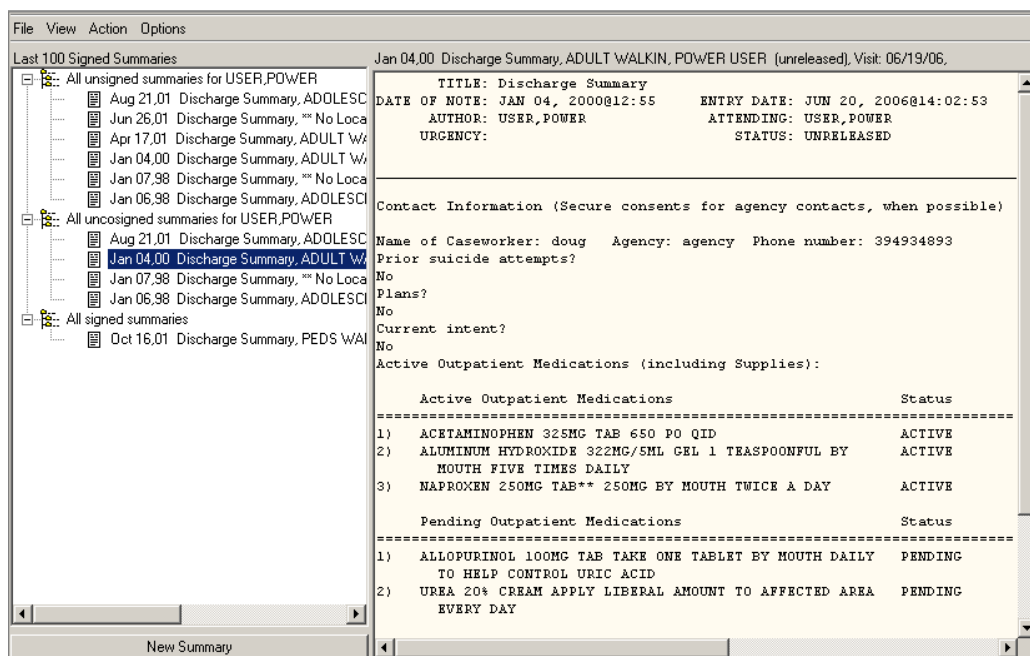


Figure 34-1: Sample Discharge Summary

The Discharge Summary (CPRS) facilitates authoring and management of discharge summaries.

### 34.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHDCSUMM.DCSUMM
Version	20.1.0.13
Class Identifier	{7D11897E-AC36-4CEF-B338-CA559B7206DE}
Image File	BEHDCSumm.ocx
Property Initializations	
Serializable Properties	AUTOSIZE=BOOL, BORDERSTYLE=ENUM, CAPTION=TEXT, CAPTIONCOLOR1=COLOR, CAPTIONCOLOR2=COLOR, CAPTIONSTYLE=ENUM, COLOR=COLOR
Required Files	BEHDCSumm.chm
Security Keys	none
Multiple Instances Allowed	no
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BEHO*1.1*018001

There are no specific implementation or maintenance tasks associated with this component.

### 34.3 Routine Descriptions

None.

### 34.4 File List

None.

### 34.5 Cross References

None.

### 34.6 Exported Options

None.

#### 34.6.1 Exported Security Keys

None.

### 34.7 Exported Protocols

None.

### 34.8 Exported Parameters

None.

### 34.9 Exported Mail Groups

None.

### 34.10 Callable Routines

None.



## 34.11 External Relations

Entity	Name	Description
Package	TIU 1.0	Uses supported APIs.

## 34.12 Internal Relations

Entity	Name	Description
Component	CPRS Support Library	Uses supported APIs.

## 34.13 Archiving and Purging

There are no archiving or purging requirements within this software.

### 34.13.1 Components

This component supports the following properties and methods:

### 34.13.2 Properties

The properties are described in the following table.

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
AUTOSIZE	Boolean	RW	If true, the component automatically resizes itself to accommodate its contents.
BORDERSTYLE	Enum	RW	Sets the style of the border surrounding the component. Can be one of: 0 = None 1 = Single 2 = Sunken 3 = Raised
CAPTION	String	RW	Sets the text displayed in the title bar. To justify portions of the caption text, use the “\” character to delimit the left-, center-, and right-justified portions of the caption text.
CAPTIONCOLOR1 CAPTIONCOLOR2	Color	RW	Colors to apply to the title bar. If the two colors differ and a gradient style is set, a gradient effect is created. For a standard title bar style, only the first color is applied.
CAPTIONSTYLE	Enum	RW	Sets the caption style. Can be one of: 0 = None – No caption (hides title bar) 1 = Title – Standard title bar 2 = Frame – Framed title bar (group box style) 3 = Left – Left gradient title bar 4 = Right – Right gradient title bar 5 = Center – Center gradient title bar
COLOR	Color	RW	Sets the background color of the component.
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPPFILE	String	RW	Sets the name of the help file associated with the component.
LAYOUT	String	RW	Property representing the internal layout of the form.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 35.0 Progress Notes

### 35.1 Introduction

The screenshot shows a software window titled "File View Action Options". On the left, there is a tree view labeled "Last 100 Signed Notes" with a list of notes including "Jun 18,07 ADULT CARE HIGH RISK SCREENING INTAKE FORM" and "Jun 18,07 NOTE WITH BOILEF". The main area on the right displays the details for the selected note: "Visit: 06/15/07 ADULT CARE HIGH RISK SCREENING INTAKE FORM, ADULT WALKIN, POWER USER (Jun 18,07@09:26)". The details include:

- Metadata:** TITLE: ADULT CARE HIGH RISK SCREENING INTAKE FORM; DATE OF NOTE: JUN 18, 2007@09:26; ENTRY DATE: JUN 18, 2007@09:26:36; AUTHOR: USER,POWER; EXP COSIGNER: ; URGENCY: ; STATUS: COMPLETED
- Demographics:** Name: DEMO,FEMALE; SSN: 300-07-2101; HR#: 21-71-83; Birthdate: APR 5,1935; Age: 72; RR 6938 Box 9362 TUBA CITY, AZ 71420; 524-843-7889 (home); Emergency Contact: YAZZIE,WILLIE
- Admission:** Admission Date: ; Attending: ; Ward/Room: ; Admitting Diagnosis: ??; Intake Date & Assigned To: ; Discharge Date: ;
- Insurance Information:** A list of checkboxes for various conditions such as "65 years or older", "Repeated Admissions", "Transportation Problem", etc.
- Vital Measurements:** TMP, HT, WT

At the bottom, there are buttons for "Templates" and "New Note".

Figure 35-1: Sample Progress Notes

The Progress Notes component facilitates the authoring and management of progress notes.

### 35.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Entity	Value
Programmatic Identifier	BEHNOTES.PROGRESSNOTES
Version	20.1.0.28
Class Identifier	{AA1670AC-315C-4352-9F78-A4BE5515D986}
Image File	BEHNotes.ocx
Property Initializations	none
Serializable Properties	AUTOSIZE=BOOL, BORDERSTYLE=ENUM, CAPTION=TEXT, CAPTIONCOLOR1=COLOR, CAPTIONCOLOR2=COLOR, CAPTIONSTYLE=ENUM, COLOR=COLOR
Required Files	BEHNotes.chm

Entity	Value
Security Keys	none
Multiple Instances Allowed	no
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BEHO*1.1*015001

There are no specific implementation or maintenance tasks associated with this component.

### 35.3 Routine Descriptions

None.

### 35.4 File List

None.

### 35.5 Cross References

None.

### 35.6 Exported Options

Option	Type	Description
BEHOTI MAIN	menu	TIU configuration menu.
BEHOTIPA AUTOSAVE NOTE	action	This parameter determines how many seconds should elapse between each auto-save of a note that is being edited in the graphical interface.
BEHOTIPA DEFAULT TEMPLATES	action	Default Template for Document Type
BEHOTIPA FIELD EDITOR CLASSES	action	Template Field Editor User Classes
BEHOTIPA MAIN	menu	TIU Parameters

Option	Type	Description
BEHOTIPA PERSONAL TEMPL ACCESS	action	Personal Template Access
BEHOTIPA TEMPL ACCESS BY CLASS	action	Personal Template Access by User Class
BEHOTIPA TEMPL PERSONAL OBJ	action	Allowed Personal Template Objects
BEHOTIPA TEMPL REM DIALOGS	action	Reminder Dialogs Allowed as Templates
BEHOTIPA TEMPL USER AUTO DEL	action	Auto Cleanup Upon User Termination
BEHOTIPA VERIFY NOTE TITLE	action	If this parameter is set to YES, the window that allows the user to change a note title will appear whenever the user starts to enter a new note, even if they have a default title. If the parameter is set to NO, - and- the user has a default title, that title will be automatically loaded when a new note is entered.

### 35.7 Exported Security Keys

None.

### 35.8 Exported Protocols

None.

### 35.9 Exported Parameters

None.

### 35.10 Exported Mail Groups

None.

### 35.11 Callable Routines

None.

## 35.12 External Relations

Entity	Name	Description
Package	TIU 1.0	Uses supported APIs.

## 35.13 Internal Relations

Entity	Name	Description
Entity	Name	Description
Component	CPRS Support Library	Uses supported APIs.

## 35.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 35.15 Components

This component supports the following properties and methods:

### 35.15.1 Properties

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
AUTOSIZE	Boolean	RW	If true, the component automatically resizes itself to accommodate its contents.

Property	Datatype	Access	Description
BORDERSTYLE	Enum	RW	Sets the style of the border surrounding the component. Can be one of: 0 = None 1 = Single 2 = Sunken 3 = Raised
CAPTION	String	RW	Sets the text displayed in the title bar. To justify portions of the caption text, use the “\” character to delimit the left-, center-, and right-justified portions of the caption text.
CAPTIONCOLOR1 CAPTIONCOLOR2	Color	RW	Colors to apply to the title bar. If the two colors differ and a gradient style is set, a gradient effect is created. For a standard title bar style, only the first color is applied.
CAPTIONSTYLE	Enum	RW	Sets the caption style. Can be one of: 0 = None – No caption (hides title bar) 1 = Title – Standard title bar 2 = Frame – Framed title bar (group box style) 3 = Left – Left gradient title bar 4 = Right – Right gradient title bar 5 = Center – Center gradient title bar
COLOR	Color	RW	Sets the background color of the component.
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPPFILE	String	RW	Sets the name of the help file associated with the component.
LAYOUT	String	RW	Property representing the internal layout of the form.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 36.0 Dictate Notes

### 36.1 Introduction



Figure 36-1: Dictate Note Button

The Dictate Note component supports the use of transcription services.

### 36.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHDICTATE.DICTATE
Version	1.2.0.17
Class Identifier	{9615A38C-663E-4368-952D-590796DA713B}
Image File	BEHDictate.ocx
Property Initializations	XMLFILE=C:\DICTAPHONE\STUDY.XML, HL7FILEMASK=C:\DICTAPHONE\%s.TXT CAPTION=TEXT, COLOR=COLOR
Serializable Properties	
Required Files	
Security Keys	none
Multiple Instances Allowed	no
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BEHO*1.1*040001

There are no specific implementation or maintenance tasks associated with this component.

### 36.3 Routine Descriptions

This component has been assigned the namespace designation of “BEHODC.” The following routines are distributed:

Routine	Description
BEHODC	Dictation support
BEHODC6	
BEHODC7	
BEHODC8	
BEHODCH	Dictation boilerplate header
BEHODCIN	Installation support
BEHODCP	Progress note lookup logic
BEHODCS	Discharge summary lookup logic.



## 36.4 File List

None.

## 36.5 Cross References

None.

## 36.6 Exported Options

Option	Type	Description
BEHODC BATCH NOTE UPLOAD	run routine	Batch note upload.

## 36.7 Exported Security Keys

None.

## 36.8 Exported Protocols

None.

## 36.9 Exported Parameters

Parameter	Instance Type	Value Type	Precedence	Description
BEHODC DIALOG TEXT	Text	Word Processing	Division, System	Holds dialog message text for the vcDictate component.
BEHODC PROBLEM FOLDER		Text	System	Network path for problem folder.
BEHODC MAXIMUM LINES		Numeric	System	This parameter holds a value that will be checked during the importing of a host file. If the number of lines stored in the TIU UPLOAD BUFFER file exceeds this value, the process will be halted, the buffer entry removed and the file moved to the network folder stored in the BEHODC PROBLEM FOLDER parameter.
BEHODC DICTATION NOTE TITLES	Numeric (Sequence #)	Pointer (#8925.1)	User, Class, Service, Division, System	This parameter holds a list of TIU document titles that are used with the Dictate component.
BEHODC ARCHIVE FOLDER		Text	System	Network path for archive folder.
BEHODC SOURCE FOLDER		Text	System	Network path for source folder.

## 36.10 Exported Mail Groups

Mail Group	Description
BEHODC PROBLEM FILE	Used by the Dictated Note Upload utility to notify members of a document that has exceeded the maximum number of lines.

## 36.11 Callable Routines

This section describes supported entry points for routines exported with this component.

### 36.11.1 BATCH^BEHODC

Scope: private.

Used by the tasked background processor to loop thru files in a system directory.

### 36.11.2 GETFILE^BEHODC

Scope: private.

Parameter	Datatype	Description
FILE	String	Name of input file containing document.

Used by background processor to file a document.

### 36.11.3 EN^BEHODC6

Scope: private.

Processes message from PACS system for notes and stores contents into TIU. The format is an HL7 message with each line of the note as a separate OBX segment. The note IEN is sent in the message.

### 36.11.4 PID^BEHODC6

Scope: private.

Checks the PID segment.

### 36.11.5 KIL^BEHODC6

Scope: private.

Cleans up the environment.

### 36.11.6 PROCESS^BEHODC7

Scope: private.

Main processing routine for message that was received through HL7 and is to be stored in TIU.

### 36.11.7 OBX^BEHODC7

Scope: private.

Process an OBX segment.

### 36.11.8 \$\$AGTEXT^BEHODC7

Scope: private.

Returns agency-specific text.

### 36.11.9 BOTH^BEHODC8

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient IEN
EVNDT	FM Date/Time	Event date/time
ERRTEXT	String	Text of error message.

Sends error text as a message acknowledgement and a local bulletin.

### 36.12 GENACK^BEHODC8

Scope: private.

Generates an HL7 ACK message.

### 36.13 External Relations

Entity	Name	Description
Package	TIU 1.0	Uses supported APIs.
Package	HL7 1.6	Uses supported APIs.

### 36.14 Internal Relations

None.

## 36.15 Archiving and Purging

There are no archiving or purging requirements within this software.

## 36.16 Components

This component supports the following properties and methods:

### 36.16.1 Properties

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
AUTOSIZE	Boolean	RW	If true, the component automatically resizes itself to accommodate its contents.
BORDERSTYLE	Enum	RW	Sets the style of the border surrounding the component. Can be one of: 0 = None 1 = Single 2 = Sunken 3 = Raised
CAPTION	String	RW	Sets the text displayed in the button caption.
COLOR	Color	RW	Sets the background color of the component.
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
GLYPH	String	RW	Name of bitmap file containing glyphs to be displayed on button surface.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPPFILE	String	RW	Sets the name of the help file associated with the component.

<b>Property</b>	<b>Datatype</b>	<b>Access</b>	<b>Description</b>
HL7FILEMASK	String	RW	Must contain the path to the PACSBRIDGE server and the user must have access to place files on that system.
LAYOUT	String	RW	Property representing the internal layout of the form.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
NUMGLYPHS	Integer	RW	Number of glyphs contained in glyph file.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.
XMLFILE	String	RW	Must point to a local directory to which the user has access on the client system.

## 37.0 Medication Management

### 37.1 Introduction

Action	Chronic	Outpatient Medications	Status	Issued	Last Filled	Expires	Refills Remaining	Rx #	Provider
	✓	ATENOLOL 25MG TAB** Qty: 45 for 90 days Sig: TAKE ONE HALF TABLET MOUTH EVERY DAY THC	Active	22-May-2006	22-May-2006	20-Aug-2006	0	157	HAGER,MARY G
	✓	GLYBURIDE 5MG TAB** Qty: 90 for 90 days Sig: TAKE 5MG MOUTH EVERY DAY TO HELP CONTROL	Active	22-May-2006	22-May-2006	23-May-2007	3	155	HAGER,MARY G
	✓	HYDROCHLOROTHIAZIDE 25MG TAB** Qty: 30 for 30 days Sig: TAKE ONE HALF TABLET MOUTH EVERY MORNING THC BLOOD PRESSURE	Active	22-May-2006	22-May-2006	23-May-2007	11	158	HAGER,MARY G
	✓	METFORMIN 500MG TAB** Qty: 180 for 90 days Sig: TAKE ONE TABLET MOUTH TWICE A DAY THC	Active	22-May-2006	22-May-2006	23-May-2007	3	154	HAGER,MARY G
		ASPIRIN 81MG TAB** Qty: 90 for 90 days Sig: TAKE ONE TABLET MOUTH EVERY DAY	Active	22-May-2006	22-May-2006	23-May-2007	3	159	HAGER,MARY G
		LOSARTAN 25MG TAB** Qty: 90 for 90 days Sig: TAKE ONE TABLET MOUTH EVERY DAY THC HIGH	Active	22-May-2006	22-May-2006	20-Aug-2006	0	156	HAGER,MARY G

Action	Inpatient Medications	Status	Stop Date

Figure 37-1: Sample Medication Window

The Medication Management component facilitates ordering and viewing of a patient's medications.

### 37.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHMEDS.MEDMANAGEMENT
Version	20.2.0.21
Class Identifier	{1FA8437A-C30C-496B-BE5A-ABA3E0C82EDE}
Image File	BEHMeds.ocx
Property Initializations	none
Serializable Properties	AUTOSIZE=BOOL, BORDERSTYLE=ENUM, CAPTION=TEXT, CAPTIONCOLOR1=COLOR, CAPTIONCOLOR2=COLOR, CAPTIONSTYLE=ENUM, COLOR=COLOR
Required Files	BEHMeds.chm
Security Keys	none
Multiple Instances Allowed	no
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BEHO*1.1*009002

There are no specific implementation or maintenance tasks associated with this component.

### 37.3 Routine Descriptions

This component has been assigned the namespace designation of “BEHORX.” The following routines are distributed:

<b>Routine</b>	<b>Description</b>
BEHORXFN	Medication management support.
BEHORXIN	Installation support.

### 37.4 File List

None.

### 37.5 Cross References

None.

### 37.6 Exported Options

<b>Option</b>	<b>Type</b>	<b>Description</b>
BEHORX COLLATION ORDER	action	Sets default collation order.
BEHORX DAYS ACTIVE	action	Sets days of medication activity.
BEHORX MAIN	menu	Medication management configuration menu.
BEHORX RX EXPIRED MAX	action	Sets renewal limit for expired meds.

### 37.7 Exported Security Keys

None.

### 37.8 Exported Protocols

None.

### 37.9 Exported Parameters

The exported parameters are described in the following table.

Parameter	Instance Type	Value Type	Precedence	Description
BEHORX COLLATION ORDER		Text	User, Division, System	Controls the default collation order for the medication list. Can be one or more of the following values: C = Chronic med status E = Expiration date F = Last fill date I = Issue date M = Medication name N = Rx # P = Provider name R = Refills remaining S = Status  To reverse the collation order, use the lowercase equivalent.
BEHORX DAYS ACTIVE		Numeric	User, Division, System	Limits the medication display to only those medications that were active within the last number of days specified.

## 37.10 Exported Mail Groups

None.

## 37.11 Callable Routines

This section describes supported entry points for routines exported with this component.

### 37.11.1 \$\$GETCMF1^BEHORXFN

Scope: private.

Parameter	Datatype	Description
ORIFN	String	Order filler number for prescription.
<return value>	Boolean	True if prescription is marked as chronic.

Returns chronic medication status for a specified order id.

### 37.11.2 RPC: BEHORXFN GETRXS

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient IEN
DAYS	Integer	# of days to include in search (default = 365).



Parameter	Datatype	Description
<return value>	String List	Returned as a list of records in the format: ~Type^PharmID^Drug^InfRate^StopDt^RefRem^TotDose^UnitDose^OrderID ^Status^LastFill^Chronic^Issued^Rx #^Provider^ Status Reason^DEA Handling  <"\" or " "><Instruction Text> where "\" indicates a new line

Returns a list of medications for populating the display grid.

### 37.11.3 RPC: BEHORXFN SETCMF

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient IEN
RXS	String or String List	Single order id or list of order ids.
CMF	Boolean	New value for chronic medication flag.
<return value>	String List	Returned as a list of errors (if any) in the format: OrderID^Error Text

Sets the chronic medication status for a single order or multiple orders.

## 37.12 External Relations

Entity	Name	Description
Package	ORDER ENTRY/RESULTS REPORTING 3.0	Uses supported API calls.
Package	OUTPATIENT PHARMACY 7.0	Uses supported API calls.

## 37.13 Internal Relations

Entity	Name	Description
Component	CPRS Support Library	Uses supported API calls.

## 37.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 37.15 Components

This component supports the following properties and methods:

### 37.15.1 Properties

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
AUTOSIZE	Boolean	RW	If true, the component automatically resizes itself to accommodate its contents.
BORDERSTYLE	Enum	RW	Sets the style of the border surrounding the component. Can be one of: 0 = None 1 = Single 2 = Sunken 3 = Raised
CAPTION	String	RW	Sets the text displayed in the title bar. To justify portions of the caption text, use the “\” character to delimit the left-, center-, and right-justified portions of the caption text.
CAPTIONCOLOR1 CAPTIONCOLOR2	Color	RW	Colors to apply to the title bar. If the two colors differ and a gradient style is set, a gradient effect is created. For a standard title bar style, only the first color is applied.
CAPTIONSTYLE	Enum	RW	Sets the caption style. Can be one of: 0 = None – No caption (hides title bar) 1 = Title – Standard title bar 2 = Frame – Framed title bar (group box style) 3 = Left – Left gradient title bar 4 = Right – Right gradient title bar 5 = Center – Center gradient title bar
COLOR	Color	RW	Sets the background color of the component.
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPPFILE	String	RW	Sets the name of the help file associated with the component.
LAYOUT	String	RW	Property representing the internal layout of the form.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 38.0 Orders

### 38.1 Introduction

Service	Order	Duration	Provider	Nurse	Clerk	Chart	Status
Lab	MICROALBUMIN DEMO URINE SP ONCE Indication: TYPE 2 DIABETES MELLITUS LB #34	Start: -1	Hager,M				active
Out. Meds	METFORMIN TAB,ORAL 500MG TAKE ONE TABLET MOUTH TWICE A DAY THE BLOOD SUGAR - TWF Quantity: 180 Refills: 3	Start: 05/22/06 Stop: 05/23/07	Hager,M				active
Out. Meds	HYDROCHLOROTHIAZIDE TAB 25MG TAKE ONE HALF TABLET MOUTH EVERY MORNING THE BLOOD PRESSURE Quantity: 30 Refills: 11	Start: 05/22/06 Stop: 05/23/07	Hager,M				active
Out. Meds	ASPIRIN TAB,EC 81MG TAKE ONE TABLET MOUTH EVERY DAY Quantity: 90 Refills: 3	Start: 05/22/06 Stop: 05/23/07	Hager,M				active
Out. Meds	*GLYBURIDE 5MG TAB** TAKE 5MG MOUTH EVERY DAY TO HELP CONTROL BLOOD SUGARS Quantity: 90 Refills: 3	Start: 05/22/06 Stop: 05/23/07	Hager,M				active
Allergy	Reaction to PENICILLINS	Start: 12/17/04 11.02	Hager,M				active

Figure 38-1: Sample Orders

The Orders (CPRS) component facilitates the placement and management of orders.

### 38.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHORDERS.ORDERENTRY
Version	20.1.0.27
Class Identifier	{A11D0650-58C6-42D8-AFB1-C26616DFECA2}
Image File	BEHOrders.ocx
Property Initializations	
Serializable Properties	AUTOSIZE=BOOL, BORDERSTYLE=ENUM, CAPTION=TEXT, CAPTIONCOLOR1=COLOR, CAPTIONCOLOR2=COLOR, CAPTIONSTYLE=ENUM, COLOR=COLOR
Required Files	BEHOrders.chm
Security Keys	none
Multiple Instances Allowed	no
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BEHO*1.1*011001

There are no specific implementation or maintenance tasks associated with this component.

### 38.3 Routine Descriptions

None.

## 38.4 File List

None.

## 38.5 Cross References

None.

## 38.6 Exported Options

Option	Type	Description
BEHOOR MAIN	menu	Order entry configuration menu
BEHOORCM ACTIONS	action	This option lets you create or change actions that can be placed on Add Orders menus.
BEHOORCM ASSIGN PRIMARY	action	This option is used to assign a primary order menu to users. A primary order menu must be assigned to each user in order to add orders using OE/RR. Examples of menus assigned to users are: ORZ ADD MENU CLINICIAN, ORZ ADD MENU NURSE, ORZ ADD MENU WARD CLERK
BEHOORCM DISABLE	action	This option lets you disable order dialogs so that execution will be prevented if selected; dialogs can also be re-enabled in this option.
BEHOORCM LIST ORDER MENUS	action	This option will print a list of users, locations, divisions etc. that have a specified order menu assigned for use in the List Manager interface.
BEHOORCM MAIN	menu	This option lets you create or change the Add Orders menus for your site.
BEHOORCM MENU	action	This option lets you create or change the Add Orders menus for your site.
BEHOORCM NEW CONSULT	action	New Consult Dialog Default
BEHOORCM NEW MED	action	New Med Dialog Default
BEHOORCM NEW PROCEDURE	action	New Procedure Dialog Default
BEHOORCM ORDER MENU STYLE	action	Order Menu Style
BEHOORCM ORDER SETS	action	This option lets you create or modify order sets, by combining a group of related quick orders into a single item.
BEHOORCM ORDERABLES	action	This option lets you create or change the things that are orderable via generic text orders at your site.

<b>Option</b>	<b>Type</b>	<b>Description</b>
BEHOORCM ORDERS	action	This option lets you create or change generic text orders that can be placed on Add Orders menus; limited access to some clinical service ordering dialogs is also available through this option.
BEHOORCM PARAMETERS	menu	Menu parameters menu.
BEHOORCM PROMPTS	action	This option lets you create or change prompts for generic order dialogs.
BEHOORCM PROTOCOLS	action	This option lets you convert any protocols that didn't get processed automatically or successfully by the CPRS order menu conversion, for use on Add Orders menus.
BEHOORCM QO ITEMS	action	This option allows a site to mark orderable items for use within pre- defined quick orders only; these items will not be selectable when ordering via standard package order dialogs.
BEHOORCM QUICK ORDERS	action	This option lets you create or change quick orders. The dialog for creating a quick order is similar to a dialog for entering an order. The responses you enter will become the pre-defined responses for this quick order. End users will have the opportunity to edit these pre-defined responses, if desired.
BEHOORCM SEARCH/REPLACE	action	This option lets you search for specific components on menus and replace or delete one or more instances of these components. Components include: quick orders, order sets, prompts, protocols, actions, etc.
BEHOORCM SEARCH/REPLACE ORD	action	This option lets you search for specific orderable items saved as responses within quick orders; a new orderable item can be selected to automatically replace it, as well.
BEHOORCM WRITE ORDERS IN	action	Write Orders List (Inpatient)
BEHOORCM WRITE ORDERS OUT	action	Write Orders List (Outpatient)
BEHOORCX CLINICAL DANGER LEVEL	action	Enter the code indicating the clinical danger level of the order check.
BEHOORCX CONTRAST MEDIA CREAT	action	Use this option to set the number of days back in time for the patient's most recent lab serum creatinine results when Imaging procedures using contrast media are ordered. This value is used in the Biochem Abnormality for Contrast Media order check.
BEHOORCX CT LIMIT HT	action	This option checks to determine if a patient is too tall to be examined by the CAT Scanner. Enter the maximum height (in inches) of a patient.

Option	Type	Description
BEHOORCX CT LIMIT WT	action	This option checks to determine if a patient weighs too much to be safely examined by the CAT Scanner. Enter the maximum weight (in pounds) of a patient.
BEHOORCX DEBUG ENABLE/ DISABLE	action	Determines if order check debug messages will be logged to ^XTMP("ORKLOG". 'E' or 'Enabled' enables the logging of order checking debug information. 'D' or 'Disabled' disables logging debug messages and cleans out ^XTMP("ORKLOG"). ^XTMP("ORKLOG" is also killed when more than 5,000 entries exist.
BEHOORCX DUP ORDER RANGE LAB	action	The number of hours back in time to check for duplicate lab orders.
BEHOORCX DUP ORDER RANGE OI	action	The number of hours back in time to look for duplicate orders.
BEHOORCX DUP ORDER RANGE RAD	action	The number of hours back in time to check for duplicate radiology orders.
BEHOORCX EDITABLE BY USER	action	This option enters values for the parameter ORK EDITABLE BY USER. IRM and Clinical Coordinators can use this option to set one or more order checks to be uneditable by end users. An order check with an Editable By User value of "No" prevents end users from changing or setting that order check's enabled/disabled parameter value. If the order check's enabled/disabled value is "enabled," in combination with an Editable By User value of "No," that order check is effectively "mandatory." (In the GUI option to enable/disable order checks, the order check will appear with a "Mandatory" flag.) Editable By User values only apply to end users. An order check with a "No" value does not prevent IRM and Clinical Coordinators from setting or changing the order check's enabled/disabled values. The order check can still be edited by users (presumably IRM and Clinical Coordinators) with access to the Order Checking Mgmt menu option "Enable/Disable an Order Check." An order check with no entry for this option/parameter will be treated as "editable by user" (as if it had a "Yes" value.) In other words, the default for this option/parameter is "Yes." This option is used by the List Manager option "Enable/Disable My Order Checks" [ORK REC PROCESSING FLAG]. In the GUI, these settings can be accessed from the Tools Options menu then selecting the Order Checks tab.
BEHOORCX EXPERT SYSTEM INQUIRE	action	Expert System Inquiry

Option	Type	Description
BEHOORCX GLUCOPHAGE CREATININE	action	Use this option to set the number of hours back in time to search for the most recent creatinine lab results. This value is used in the order check Glucophage-Lab Results.
BEHOORCX LOCAL TERM EDIT	action	Edit Local Terms.
BEHOORCX MAIN	menu	Order check configuration menu.
BEHOORCX MRI LIMIT HT	action	This option checks to determine if a patient is too tall to be safely examined by the MRI Scanner. Enter the maximum height (in inches) of a patient.
BEHOORCX MRI LIMIT WT	action	This option checks to determine if a patient weighs too much to be safely examined by the MRI scanner. Enter the maximum weight (in pounds) for the patient.
BEHOORCX ORDER CHK MGMT MENU	menu	Order check parameters menu.
BEHOORCX POLYPHARMACY	action	Use this option to set the number medications for determining polypharmacy.
BEHOORCX PROCESSING FLAG	action	Enter the code indicating the processing flag for the entity and order check.
BEHOORCX RECIP ORDER CHECKS	action	This option prompts for a user/recipient then processes each order check to determine if and why the user will receive the order check message during the ordering process.
BEHOORCX RULE ACTIVATE	action	Activate/InactivateRules
BEHOORCX RUN COMPILER	action	Compile Rules
BEHOORCX SYSTEM ENABLE/DISABLE	action	Determines if any order checking will occur. 'E' or 'Enable' indicates order checking is enabled and running. 'D' or 'Disabled' indicates order checking is disabled and not running. Can be set at the institution, System, or Package level.
BEHOORDO DELAYED ORDERS	action	Release/Cancel Delayed Orders
BEHOORDO DELAYED ORDERS EDITOR	action	This option invokes the auto-dc rules/release events editor. Use this option to define and maintain your auto-discontinue rules as well as your delayed order release events. Whether or not orders auto-discontinue will depend on how you have your rules defined. In addition, orders that are being written for delayed release can only be delayed to the release events that you've defined.
BEHOORDO ENABLE DELAYED ORDERS	action	Enable Event-Delayed Orders

<b>Option</b>	<b>Type</b>	<b>Description</b>
BEHOORDO EVENT PARAMETERS	action	This option will allow you to edit the parameters related to event delayed orders.
BEHOORDO MAIN	menu	Delayed orders configuration menu.
BEHOORDO PATIENT EVENT INQUIRY	action	This option gives a FileMan inquiry of an entry from the patient event file. This file holds information related to orders placed on delay. Included in this information are orders released and auto-dcd due to an event occurring as well as an audit trail of MAS actions taken related to this delayed event.
BEHOORKY KEY ALLOCATION	action	This option is to assist the OE/RR Clinical Coordinator when allocating Security keys to users of the OE/RR system.
BEHOORKY KEY CHECK	action	This option will identify users that have more than one OR key. Users must only have one OR key to correctly use the software. Any users identified need to have their keys edited so that only one of the OR keys remain (ORES, OREMAS, ORELSE).
BEHOORKY MAIN	menu	Key management menu.
BEHOORPA DISABLE HOLD ORDERS	action	Disable Hold/Unhold Actions in EHR
BEHOORPA DISABLE ORDERING	action	Disable Ordering in EHR
BEHOORPA ENABLE VERIFY	action	Enable/Disable Order Verify Actions
BEHOORPA MAIN	menu	Order parameters menu.
BEHOORPA ORDER MISC	action	This option is for editing miscellaneous hospital wide OE/RR parameters.
BEHOORPA ORDER REASON	action	This option allows access to the Order Reason file to enter or edit reasons for discontinuing an order.
BEHOORPA UNSIGNED ORDERS VIEW	action	This option can be used to set the default view of unsigned orders that ORES key holders will see when exiting a patient's chart; possible views are only those orders entered during the current session, all of the current user's unsigned orders, or all unsigned orders for the patient.
BEHOORPR CHART COPY	action	This option is for editing hospital wide Chart Copy parameters.
BEHOORPR MAIN	menu	This menu is for editing print parameters. It should be available to the clinical coordinator and IRM Staff.



Option	Type	Description
BEHOORPR PRINTS (HOSP)	action	This option is for editing hospital wide print parameters for OE/RR.
BEHOORPR PRINTS (LOC)	action	This option is for editing print parameters for each ward/ clinic location.
BEHOORPR REQ/LABEL	action	This option is for editing requisition and label site parameters.
BEHOORPR SERVICE COPY	action	This option is for editing Service Copy site parameters.
BEHOORPR SUMMARY REPORTS	action	This option is for editing Summary Report site parameters.
BEHOORPR WORK COPY	action	This option is for editing Work Copy site Parameters.
BEHOORRP MAIN	menu	Order Reports
BEHOORRP NATURE/STATUS ORDERS	action	This option will allow the user to search orders for a specific NATURE OF ORDER or order STATUS. There are two formats available. One is a detailed display that is printed in real time as the order number that meets the search criteria is encountered. There is no further sort capability for this format. The second format is the columnar format which will allow the sort by ENTERING person or by SERVICE/ SECTION. This format works best if sending the output to a 132 column compressed printer or to the BROWSER device.
BEHOORRP PERFORMANCE MONITOR	action	This option produces a report showing the number of orders entered by provider. The report includes a detailed listing as well as a summary. The detail listing includes pertinent information about each order associated with the listed provider. The summary report shows the number of orders entered for a provider, the number of orders entered by an ORES key holder, the percentage of orders entered by an ORES key holder for the given provider and a break down of the orders by nature of order for those that were entered by a non-ORES key holder.

Option	Type	Description
BEHOORRP UNSIGNED ORDERS	action	This option will allow the user to search for either RELEASED but UNSIGNED orders or just UNSIGNED orders and sort them by Service/section, Provider, Patient, or Location. A start date entry will allow the site to ignore unsigned orders that fall within their allowed grace period. For example: the site allows the clinician 48 hours to sign unsigned orders, you would enter T-2 for a start date. A stop date entry will allow the site to ignore orders older than the date entered. An example of this would be, if you wanted to ignore unsigned orders that were placed prior to CPRS and you installed CPRS on Jan 1, 1999, you would enter 010199.

### 38.7 Exported Security Keys

None.

### 38.8 Exported Protocols

None.

### 38.9 Exported Parameters

None.

### 38.10 Exported Mail Groups

None.

### 38.11 Callable Routines

None.

### 38.12 External Relations

Entity	Name	Description
Package	ORDER ENTRY/RESULTS REPORTING 3.0	Uses supported APIs.

### 38.13 Internal Relations

Entity	Name	Description
Component	CPRS Support Library	Uses supported APIs.

### 38.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 38.15 Components

This component supports the following properties and methods:

### 38.15.1 Properties

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
AUTOSIZE	Boolean	RW	If true, the component automatically resizes itself to accommodate its contents.
BORDERSTYLE	Enum	RW	Sets the style of the border surrounding the component. Can be one of: 0 = None 1 = Single 2 = Sunken 3 = Raised
CAPTION	String	RW	Sets the text displayed in the title bar. To justify portions of the caption text, use the “\” character to delimit the left-, center-, and right-justified portions of the caption text.
CAPTIONCOLOR1 CAPTIONCOLOR2	Color	RW	Colors to apply to the title bar. If the two colors differ and a gradient style is set, a gradient effect is created. For a standard title bar style, only the first color is applied.
CAPTIONSTYLE	Enum	RW	Sets the caption style. Can be one of: 0 = None – No caption (hides title bar) 1 = Title – Standard title bar 2 = Frame – Framed title bar (group box style) 3 = Left – Left gradient title bar 4 = Right – Right gradient title bar 5 = Center – Center gradient title bar
COLOR	Color	RW	Sets the background color of the component.

<b>Property</b>	<b>Datatype</b>	<b>Access</b>	<b>Description</b>
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPPFILE	String	RW	Sets the name of the help file associated with the component.
LAYOUT	String	RW	Property representing the internal layout of the form.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 39.0 Quick Order Wizard

### 39.1 Introduction

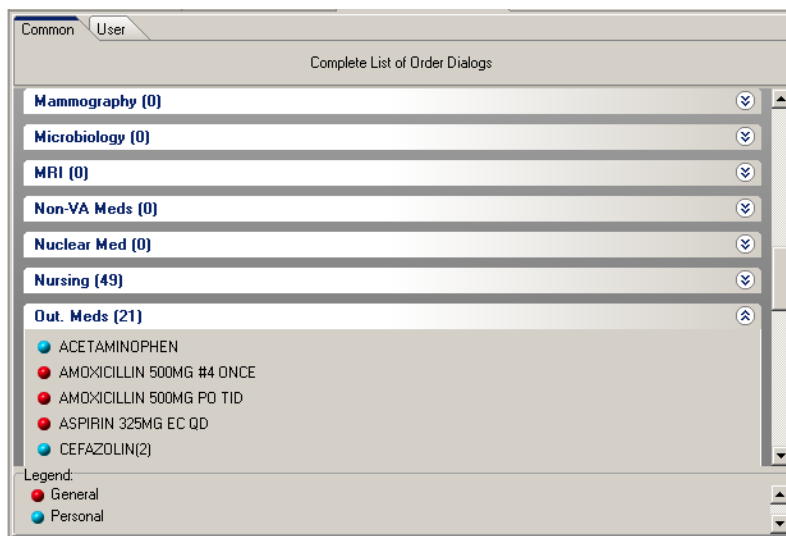


Figure 39-1: Sample Quick Order Wizard

The Quick Order Wizard facilitates the creation and maintenance of quick orders.

### 39.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHQOWIZARD.QOWIZARD
Version	1.1.0.17
Class Identifier	{9B97DAAB-AA13-4C2B-B27D-014C897DC457}
Image File	BEHQOWizard.ocx
Property Initializations	none
Serializable Properties	CAPTION=TEXT, CAPTIONCOLOR1=COLOR, CAPTIONCOLOR2=COLOR, CAPTIONSTYLE=ENUM
Required Files	none
Security Keys	none
Multiple Instances Allowed	no
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BEHO*1.1*039001

There are no specific implementation or maintenance tasks associated with this component.

### 39.3 Routine Descriptions

This component has been assigned the namespace designation of “BEHOQOW.” The following routines are distributed:

<b>Routine</b>	<b>Description</b>
BEHOQOW	Quick Order Wizard support.

### 39.4 File List

None.

### 39.5 Cross References

None.

### 39.6 Exported Options

None.

### 39.7 Exported Security Keys

None.

### 39.8 Exported Protocols

None.

### 39.9 Exported Parameters

None.

### 39.10 Exported Mail Groups

None.

### 39.11 Callable Routines

This section describes supported entry points for routines exported with this component.

#### 39.11.1 BEHOQOW CANDEL

Scope: private.

Parameter	Datatype	Description
BEHOQO	Pointer (#101.41)	IEN of order dialog
USR	Pointer (#200)	User IEN. If >0, always return true.
<return value>	Boolean	Returns true if order dialog can be deleted.

Returns delete status of an order dialog.

### 39.11.2 BEHOQOW CLONE

Scope: private.

Parameter	Datatype	Description
FIEN	Pointer (#101.41)	IEN of source order dialog.
TIEN	Pointer (#101.41)	IEN of target order dialog.
<return value>	Boolean	Returns true if successful.

Clones a quick order dialog.

### 39.11.3 BEHOQOW DEFDISGP

Scope: private.

Parameter	Datatype	Description
GRP	Pointer (#100.98)	Display group IEN
<return value>	Pointer (#100.98)	The IEN of the main display group.

Given a display group IEN, returns the IEN of the main display group.

### 39.11.4 BEHOQOW DELETEQO

Scope: private.

Parameter	Datatype	Description
BEHOQO	Pointer (#101.41)	Order dialog IEN
USR	Pointer (#200)	User IEN
DGRP	Pointer (#100.98)	Display group IEN
DNM	String	Display name
<return value>	Boolean	Returns true if operation was successful.

Deletes the specified order dialog.

### 39.11.5 BEHOQOW DISGRP

Scope: private.

Parameter	Datatype	Description
GRPTYP	Integer	One of: 0 = common 1 = user
GRPUSR	Pointer (#200)	If GRPTYP=1, use this user to retrieve groups.
<return value>	String List	List of display groups.

Returns a list of display groups of the specified type.

### 39.11.6 BEHOQOW GETDISAB

Scope: private.

Parameter	Datatype	Description
BEHOQO	Pointer (#101.41)	Order dialog IEN
<return value>	Boolean	Returns true if the order dialog has been disabled.

Returns true if the order dialog has been disabled.

### 39.11.7 BEHOQOW GETPKG

Scope: private.

Parameter	Datatype	Description
DISGRP	Pointer (#100.98)	Display group IEN
<return value>	Pointer (#9.4)	Package IEN

Returns the IEN of the PACKAGE file entry corresponding to the specified display group.

### 39.11.8 BEHOQOW GRPDEFWD

Scope: private.

Parameter	Datatype	Description
GRP	Pointer (#100.98)	Display group IEN
<return value>	Numeric	Window form ID

Returns the WINDOW FORM ID associated with the specified display group.

### 39.11.9 BEHOQOW PROPERTY

Scope: private.

Parameter	Datatype	Description
BEHOQO	Pointer (#101.41)	Order dialog IEN
<return value>	String List	Returns a list of property values in display format.

Returns a list of property values for the specified quick order.



## 39.11.10 BEHOQOW QOFVAL

Scope: private.

Parameter	Datatype	Description
IEN	Pointer (#101.41)	Order dialog IEN
FLD (optional)	Number	Field number (defaults to .01)
FLG (optional)	Boolean	If true, returns in upper case. Defaults to false.
<return value>	String	Value of specified field in external format.

Returns the value of the specified field from the specified order dialog.

## 39.11.11 BEHOQOW QOITEMS

Scope: private.

Parameter	Datatype	Description
GRP (optional)	Pointer (#100.98)	Display group IEN (defaults to all groups)
USR (optional)	Pointer (#200)	User IEN (defaults to all users)
<return value>	String List	List of quick orders

Returns a list of quick orders matching the specified criteria.

## 39.11.12 BEHOQOW SETDISAB

Scope: private.

Parameter	Datatype	Description
BEHOQO	Pointer (#101.41)	Order dialog IEN
MSG	String	Disabled dialog message. Pass "@" to clear and enable dialog.
<return value>	String	Failure: error text Success: null

Sets or clears the disable text for the specified dialog.

## 39.11.13 BEHOQOW UPDRSP

Scope: private.

Parameter	Datatype	Description
IEN	Pointer (#101.41)	Order dialog IEN
DGRP	Pointer (#100.98)	Display group IEN
<return value>	Integer	Always returns 0.

Updates quick order responses.

## 39.12 External Relations

Entity	Name	Description
Package	ORDER ENTRY/RESULTS REPORTING 3.0	Uses supported APIs

## 39.13 Internal Relations

Entity	Name	Description
Component	CPRS Support Library	Uses supported APIs

## 39.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 39.15 Components

This component supports the following properties and methods:

### 39.15.1 Properties

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
AUTOSIZE	Boolean	RW	If true, the component automatically resizes itself to accommodate its contents.
BORDERSTYLE	Enum	RW	Sets the style of the border surrounding the component. Can be one of: None Single Sunken Raised
CAPTION	String	RW	Sets the text displayed in the title bar. To justify portions of the caption text, use the “\” character to delimit the left-, center-, and right-justified portions of the caption text.

Property	Datatype	Access	Description
CAPTIONCOLOR1 CAPTIONCOLOR2	Color	RW	Colors to apply to the title bar. If the two colors differ and a gradient style is set, a gradient effect is created. For a standard title bar style, only the first color is applied.
CAPTIONSTYLE	Enum	RW	Sets the caption style. Can be one of: None – No caption (hides title bar) Title – Standard title bar Frame – Framed title bar (group box style) Left – Left gradient title bar Right – Right gradient title bar Center – Center gradient title bar
COLOR	Color	RW	Sets the background color of the component.
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPPFILE	String	RW	Sets the name of the help file associated with the component.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
MINHEIGHT	Integer	RW	Sets the minimum height, in pixels, that the component can attain.
MINWIDTH	Integer	RW	Sets the minimum width, in pixels, that the component can attain.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 40.0 Consult Order History

### 40.1 Introduction

Consult Orders		
Service ▲	Date	Status
HOME OXYGEN REQUEST	19-Jun-2007 10:57	c
HOME OXYGEN REQUEST	26-Apr-2007 12:07	p
HOME OXYGEN REQUEST	06-Jul-2006 13:45	p
PROSTHETICS REQUEST	26-Apr-2007 12:07	c

Figure 40-1: Sample Consults

The Consult Order History component provides a quick overview of consult orders for display on the cover sheet.

### 40.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHCONSULTORDERS.CONSULTORDERS
Version	4.2.0.9
Class Identifier	{4C369EA9-584D-4638-AF26-F1E90CF53B0F}
Image File	BEHConsultOrders.ocx
Property Initializations	HELPPFILE=BEHCover.chm
Serializable Properties	ALLOWPRINT=BOOL, AUTOSIZE=BOOL, BORDERSTYLE=ENUM, CAPTION=TEXT, CAPTIONCOLOR1=COLOR, CAPTIONCOLOR2=COLOR, CAPTIONSTYLE=ENUM, COLOR=COLOR, DEFERUPDATE=BOOL, DETAILPANE=BOOL, ORIENTATION=ENUM, LAYOUT=HIDDEN
Required Files	
Security Keys	none
Multiple Instances Allowed	yes
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BEHO*1.1*028001

There are no specific implementation or maintenance tasks associated with this component.

### 40.3 Routine Descriptions

This component has been assigned the namespace designation of “BEHOCNCV.” The following routines are distributed:

<b>Routine</b>	<b>Description</b>
BEHOCNCV	Consult order history cover sheet support.

#### 40.4 File List

None.

#### 40.5 Cross References

None.

#### 40.6 Exported Options

None.

#### 40.7 Exported Security Keys

None.

#### 40.8 Exported Protocols

None.

#### 40.9 Exported Parameters

None.

#### 40.10 Exported Mail Groups

None.

#### 40.11 Callable Routines

This section describes supported entry points for routines exported with this component.

##### 40.11.1 RPC: BEHOCNCV DETAIL

Scope: private.

<b>Parameter</b>	<b>Datatype</b>	<b>Description</b>
DFN	Pointer (#2)	Patient IEN.
IEN	Pointer (#123)	Consult order IEN.
<return value>	String List	Detail text.

Returns detail text for the specified consult order.

## 40.11.2 RPC: BEHOCNCV LIST

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient IEN.
<return value>	String List	List of records in the format: IEN^STATUS^FORMATTED DATE^TYPE^FM DT

Returns a list of consult orders for the specified patient.

## 40.12 External Relations

Entity	Name	Description
Package	CONSULT/REQUEST TRACKING 3.0	Uses supported APIs.

## 40.13 Internal Relations

None.

## 40.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 40.15 Components

This component supports the following properties and methods:

### 40.15.1 Properties

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ALLOWPRINT	Boolean	RW	If true, a print button will appear on the detail dialog allowing printing of the contents.
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom

Property	Datatype	Access	Description
AUTOSIZE	Boolean	RW	If true, the component automatically resizes itself to accommodate its contents.
BORDERSTYLE	Enum	RW	Sets the style of the border surrounding the component. Can be one of: 0 = None 1 = Single 2 = Sunken 3 = Raised
CAPTION	String	RW	Sets the text displayed in the title bar. To justify portions of the caption text, use the “\” character to delimit the left-, center-, and right-justified portions of the caption text.
CAPTIONCOLOR1 CAPTIONCOLOR2	Color	RW	Colors to apply to the title bar. If the two colors differ and a gradient style is set, a gradient effect is created. For a standard title bar style, only the first color is applied.
CAPTIONSTYLE	Enum	RW	Sets the caption style. Can be one of: 0 = None – No caption (hides title bar) 1 = Title – Standard title bar 2 = Frame – Framed title bar (group box style) 3 = Left – Left gradient title bar 4 = Right – Right gradient title bar 5 = Center – Center gradient title bar
COLOR	Color	RW	Sets the background color of the component.
DEFERUPDATE	Boolean	RW	If true, data refresh is deferred until the component becomes visible. If false, data refresh happens immediately.
DETAILPANE	Boolean	RW	If true, a detail pane appears next to the list view that displays detail text of the selected entry. If false, detail text appears in a popup dialog when an entry is clicked.
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPPFILE	String	RW	Sets the name of the help file associated with the component.
LAYOUT	String	RW	Property representing the internal layout of the form.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
ORIENTATION	Enum	RW	Sets the orientation of the detail pane. Can be one of: 0 = Horizontal 1 = Vertical
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 41.0 Lab Orders

### 41.1 Introduction

Lab Orders		
Lab Order ▲	Status	Date
CBC BLOOD SP Indication: DIABETES MELLITUS LB #3	COMPLETE	21-Jun-2007 09:59

Figure 41-1: Sample Lab Orders

The Lab Orders component provides a quick overview of outstanding lab orders for display on the cover sheet.

### 41.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHLABORDERS.LABORDERS
Version	4.2.0.7
Class Identifier	{CE4EB1BE-71EF-427E-ABAF-1257685A4624}
Image File	BEHLabOrders.ocx
Property Initializations	none
Serializable Properties	ALLOWPRINT=BOOL, AUTOSIZE=BOOL, BORDERSTYLE=ENUM, CAPTION=TEXT, CAPTIONCOLOR1=COLOR, CAPTIONCOLOR2=COLOR, CAPTIONSTYLE=ENUM, COLOR=COLOR, DEFERUPDATE=BOOL, DETAILPANE=BOOL, ORIENTATION=ENUM, LAYOUT=HIDDEN
Required Files	BEHLabOrders.chm
Security Keys	none
Multiple Instances Allowed	yes
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BEHO*1.1*032001

There are no specific implementation or maintenance tasks associated with this component.

### 41.3 Routine Descriptions

This component has been assigned the namespace designation of “BEHOLRCV.” The following routines are distributed:



Routine	Description
BEHOLRCV	Lab order cover sheet support.

#### 41.4 File List

None.

#### 41.5 Cross References

None.

#### 41.6 Exported Options

Option	Type	Description
BEHOLR MAIN	menu	Lab configuration menu.
BEHOLRCV DATE RANGE	action	Days of lab results to retrieve.

#### 41.7 Exported Security Keys

None.

#### 41.8 Exported Protocols

None.

#### 41.9 Exported Parameters

Parameter	Instance Type	Value Type	Precedence	Description
BEHOLRCV DATE RANGE	Set (Inpatient or Outpatient)	Integer	User, Location, Service, Division, System	The number of days back in time to search for lab orders/results. If not indicated, the default period of 2 days will be used. The maximum number of days is 100,000 or about 220 years.

#### 41.10 Exported Mail Groups

None.

#### 41.11 Callable Routines

This section describes supported entry points for routines exported with this component.

##### 41.11.1 RPC: BEHOLRCV DETAIL

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient IEN.
ORID	String	Order ID.
ID	Pointer (#100)	Order IEN.
<return value>	String List	Detail text.

Returns detail text for the specified lab order.

#### 41.11.2 RPC: BEHOLRCV LIST

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient IEN.
<return value>	String List	List of lab orders.

Returns a list of lab orders for populating the list view.

### 41.12 External Relations

Entity	Name	Description
Package	ORDER ENTRY/RESULTS REPORTING 3.0	Uses the following APIs: LIST^ORQOR1 ORDERS^ORCXPND1

### 41.13 Internal Relations

None.

### 41.14 Archiving and Purging

There are no archiving or purging requirements within this software.

### 41.15 Components

This component supports the following properties and methods:

#### 41.15.1 Properties

The properties are described in the following table.

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ALLOWPRINT	Boolean	RW	If true, a print button will appear on the detail dialog allowing printing of the contents.
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
AUTOSIZE	Boolean	RW	If true, the component automatically resizes itself to accommodate its contents.
BORDERSTYLE	Enum	RW	Sets the style of the border surrounding the component. Can be one of: 0 = None 1 = Single 2 = Sunken 3 = Raised
CAPTION	String	RW	Sets the text displayed in the title bar. To justify portions of the caption text, use the “\” character to delimit the left-, center-, and right-justified portions of the caption text.
CAPTIONCOLOR1 CAPTIONCOLOR2	Color	RW	Colors to apply to the title bar. If the two colors differ and a gradient style is set, a gradient effect is created. For a standard title bar style, only the first color is applied.
CAPTIONSTYLE	Enum	RW	Sets the caption style. Can be one of: 0 = None – No caption (hides title bar) 1 = Title – Standard title bar 2 = Frame – Framed title bar (group box style) 3 = Left – Left gradient title bar 4 = Right – Right gradient title bar 5 = Center – Center gradient title bar
COLOR	Color	RW	Sets the background color of the component.
DEFERUPDATE	Boolean	RW	If true, data refresh is deferred until the component becomes visible. If false, data refresh happens immediately.
DETAILPANE	Boolean	RW	If true, a detail pane appears next to the list view that displays detail text of the selected entry. If false, detail text appears in a popup dialog when an entry is clicked.
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.

<b>Property</b>	<b>Datatype</b>	<b>Access</b>	<b>Description</b>
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPPFILE	String	RW	Sets the name of the help file associated with the component.
LAYOUT	String	RW	Property representing the internal layout of the form.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
ORIENTATION	Enum	RW	Sets the orientation of the detail pane. Can be one of: 0 = Horizontal 1 = Vertical
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 42.0 Medications

### 42.1 Introduction

Medications			
Medication	Status	Issue Date	
ACETAMINOPHEN 325MG TAB	ACTIVE	15-Jun-2007	
ACETAMINOPHEN 325MG TAB	ACTIVE	31-May-2007	
ACETAMINOPHEN 325MG TAB	ACTIVE	21-Aug-2006	
CIMETIDINE 300MG TAB	ACTIVE	21-Aug-2006	
METHYLDOPATE INJ	PENDING	29-Mar-2006 09:...	
ACETAMINOPHEN 325MG TAB	ACTIVE		
ALUMINUM HYDROXIDE 322MG/5ML GEL	DISCONTINUED		
ALUMINUM HYDROXIDE 322MG/5ML GEL	ACTIVE		
NAPROXEN 250MG TAB**	ACTIVE		

Status:  All  Active  
 Inpatient/Outpatient:  All  Out  In

Figure 42-1: Sample Medications

The Medications component provides an overview of a patient's medication profile for display on the cover sheet.

### 42.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHMEDLIST.MEDLIST
Version	4.2.0.11
Class Identifier	{F76FC2D9-4837-4F0A-A411-015D985AB1E6}
Image File	BEHMedList.ocx
Property Initializations	none
Serializable Properties	ALLOWPRINT=BOOL, AUTOSIZE=BOOL, BORDERSTYLE=ENUM, CAPTION=TEXT, CAPTIONCOLOR1=COLOR, CAPTIONCOLOR2=COLOR, CAPTIONSTYLE=ENUM, COLOR=COLOR, DEFERUPDATE=BOOL, DETAILPANE=BOOL, ORIENTATION=ENUM, LAYOUT=HIDDEN
Required Files	BEHMedList.chm
Security Keys	none
Multiple Instances Allowed	yes
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BEHO*1.1*033001

There are no specific implementation or maintenance tasks associated with this component.

## 42.3 Routine Descriptions

This component has been assigned the namespace designation of “BEHORXCV.” The following routines are distributed:

<b>Routine</b>	<b>Description</b>
BEHORXCV	Medication cover sheet support.

## 42.4 File List

None.

## 42.5 Cross References

None.

## 42.6 Exported Options

None.

## 42.7 Exported Security Keys

None.

## 42.8 Exported Protocols

None.

## 42.9 Exported Parameters

None.

## 42.10 Exported Mail Groups

None.

## 42.11 Callable Routines

This section describes supported entry points for routines exported with this component.

### 42.11.1 RPC: BEHORXCV DETAIL

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient IEN.
ID	String	Medication ID.
<return value>	String List	Detail text.

Returns detailed information about the specified medication.

#### 42.11.2 RPC: BEHORXCV LIST

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient IEN.
<return value>	String List	List of medications.

Returns a list of medications for populating the list view.

### 42.12 External Relations

Entity	Name	Description
Package	OUTPATIENT PHARMACY 7.0	Uses the following APIs: OCL^PSOORRL OEL^PSOORRL

### 42.13 Internal Relations

None.

### 42.14 Archiving and Purging

There are no archiving or purging requirements within this software.

### 42.15 Components

This component supports the following properties and methods:

#### 42.15.1 Properties

The properties are described in the following table.

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ALLOWPRINT	Boolean	RW	If true, a print button will appear on the detail dialog allowing printing of the contents.
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
AUTOSIZE	Boolean	RW	If true, the component automatically resizes itself to accommodate its contents.
BORDERSTYLE	Enum	RW	Sets the style of the border surrounding the component. Can be one of: 0 = None 1 = Single 2 = Sunken 3 = Raised
CAPTION	String	RW	Sets the text displayed in the title bar. To justify portions of the caption text, use the “\” character to delimit the left-, center-, and right-justified portions of the caption text.
CAPTIONCOLOR1 CAPTIONCOLOR2	Color	RW	Colors to apply to the title bar. If the two colors differ and a gradient style is set, a gradient effect is created. For a standard title bar style, only the first color is applied.
CAPTIONSTYLE	Enum	RW	Sets the caption style. Can be one of: 0 = None – No caption (hides title bar) 1 = Title – Standard title bar 2 = Frame – Framed title bar (group box style) 3 = Left – Left gradient title bar 4 = Right – Right gradient title bar 5 = Center – Center gradient title bar
COLOR	Color	RW	Sets the background color of the component.
DEFERUPDATE	Boolean	RW	If true, data refresh is deferred until the component becomes visible. If false, data refresh happens immediately.
DETAILPANE	Boolean	RW	If true, a detail pane appears next to the list view that displays detail text of the selected entry. If false, detail text appears in a popup dialog when an entry is clicked.
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.



<b>Property</b>	<b>Datatype</b>	<b>Access</b>	<b>Description</b>
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPPFILE	String	RW	Sets the name of the help file associated with the component.
LAYOUT	String	RW	Property representing the internal layout of the form.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
ORIENTATION	Enum	RW	Sets the orientation of the detail pane. Can be one of: 0 = Horizontal 1 = Vertical
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 43.0 Health Summary Report

### 43.1 Introduction



Figure 43-1: Sample Health Summary Report Button

The Health Summary Report component permits easy access to any of several predefined report types.

### 43.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHHSREPORT.HSREPORT
Version	1.0.0.58
Class Identifier	{C65320E0-A0F5-4511-9EA9-C76C50D9FEE7}
Image File	BEHHSReport.ocx
Property Initializations	
Serializable Properties	GLYPH=IMAGE, CAPTION=TEXT, LAYOUT=ENUM, ASYNCHRONOUS=BOOL, ENCOUNTERREQUIRED=BOOL, REFRESHOPTION=ENUM, REPORT=TEXT, TITLE=TEXT
Required Files	
Security Keys	none
Multiple Instances Allowed	yes
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BEHO*1.1*042001

There are no specific implementation or maintenance tasks associated with this component.

### 43.3 Routine Descriptions

None.

### 43.4 File List

None.

## 43.5 Cross References

None.

## 43.6 Exported Options

None.

## 43.7 Exported Security Keys

None.

## 43.8 Exported Protocols

None.

## 43.9 Exported Parameters

None.

## 43.10 Exported Mail Groups

None.

## 43.11 Callable Routines

None.

## 43.12 External Relations

Entity	Name	Description
Package	ORDER ENTRY/RESULTS REPORTING 3.0	Uses supported APIs.

## 43.13 Internal Relations

None.

## 43.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 43.15 Components

This component supports the following properties and methods:

## 43.15.1 Properties

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
ASYNCHRONOUS	Boolean	RW	If true, the report is generated asynchronously. If false, the user must wait until the report is complete.
AUTOSIZE	Boolean	RW	If true, the component automatically resizes itself to accommodate its contents.
BORDERSTYLE	Enum	RW	Sets the style of the border surrounding the component. Can be one of: 0 = None 1 = Single 2 = Sunken 3 = Raised
CAPTION	String	RW	Sets the text displayed in the button caption.
COLOR	Color	RW	Sets the background color of the component.
ENCOUNTER	Boolean	RW	If true, report requires an encounter context.
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
GLYPH	String	RW	Name of bitmap file containing glyphs to be displayed on button surface.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPPFILE	String	RW	Sets the name of the help file associated with the component.
LAYOUT	String	RW	Property representing the internal layout of the form.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
NUMGLYPHS	Integer	RW	Number of glyphs contained in glyph file.

Property	Datatype	Access	Description
REFRESHOPTION	Enum	RW	Controls behavior upon receipt of a refresh request. One of: 0 = Ignore refresh requests 1 = If a report is displayed on receipt of a refresh request, its contents will be refreshed. 2 = Always refresh report. If a report is not displayed, it will be displayed on receipt of a refresh request. Otherwise, the displayed report is refreshed. 3 = If a report is displayed on receipt of a refresh request, it is closed.
REPORT	String	RW	Report control parameters. Format is: <report id>^<HS type>^<date range>^<exam id>
TITLE	String	RW	Title for the report.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 44.0 Lab Results

### 44.1 Introduction

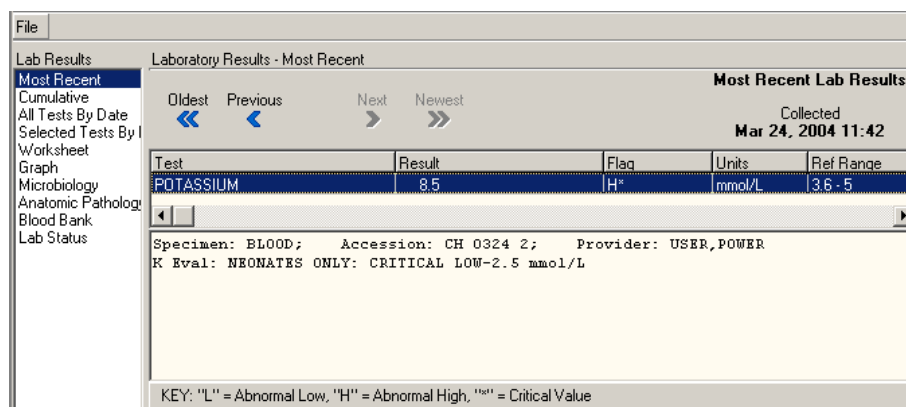


Figure 44-1: Sample Lab Results

The Lab Results component permits viewing of lab results in a variety of formats.

### 44.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHLAB.LABVIEW
Version	20.1.0.17
Class Identifier	{F9DD2047-8DDB-41A0-9EF6-7F2850B7F09A}
Image File	BEHLab.ocx
Property Initializations	
Serializable Properties	AUTOSIZE=BOOL, BORDERSTYLE=ENUM, CAPTION=TEXT, CAPTIONCOLOR1=COLOR, CAPTIONCOLOR2=COLOR, CAPTIONSTYLE=ENUM, COLOR=COLOR
Required Files	BEHLab.chm
Security Keys	none
Multiple Instances Allowed	yes
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BEHO*1.1*019001

There are no specific implementation or maintenance tasks associated with this component.

### 44.3 Routine Descriptions

None.

#### 44.4 File List

None.

#### 44.5 Cross References

None.

#### 44.6 Exported Options

None.

#### 44.7 Exported Security Keys

None.

#### 44.8 Exported Protocols

None.

#### 44.9 Exported Parameters

None.

#### 44.10 Exported Mail Groups

None.

#### 44.11 Callable Routines

None.

#### 44.12 External Relations

Entity	Name	Description
Package	ORDER ENTRY/RESULTS REPORTING 3.0	Uses supported APIs.
Package	LAB 5.2	Uses supported APIs.

#### 44.13 Internal Relations

Entity	Name	Description
Component	CPRS Support Library	Uses supported APIs.

#### 44.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 44.15 Components

This component supports the following properties and methods:

### 44.15.1 Properties

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
AUTOSIZE	Boolean	RW	If true, the component automatically resizes itself to accommodate its contents.
BORDERSTYLE	Enum	RW	Sets the style of the border surrounding the component. Can be one of: 0 = None 1 = Single 2 = Sunken 3 = Raised
CAPTION	String	RW	Sets the text displayed in the title bar. To justify portions of the caption text, use the “\” character to delimit the left-, center-, and right-justified portions of the caption text.
CAPTIONCOLOR1 CAPTIONCOLOR2	Color	RW	Colors to apply to the title bar. If the two colors differ and a gradient style is set, a gradient effect is created. For a standard title bar style, only the first color is applied.
CAPTIONSTYLE	Enum	RW	Sets the caption style. Can be one of: 0 = None – No caption (hides title bar) 1 = Title – Standard title bar 2 = Frame – Framed title bar (group box style) 3 = Left – Left gradient title bar 4 = Right – Right gradient title bar 5 = Center – Center gradient title bar
COLOR	Color	RW	Sets the background color of the component.
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPPFILE	String	RW	Sets the name of the help file associated with the component.



<b>Property</b>	<b>Datatype</b>	<b>Access</b>	<b>Description</b>
LAYOUT	String	RW	Property representing the internal layout of the form.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 45.0 Remote Data (CPRS)

### 45.1 Introduction



Figure 45-1: Sample Remote Data Button

The Remote Data (CPRS) component provides access to report from other sites where a patient can be receiving care.

### 45.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHREMOTEDATA.REMOTEDATA
Version	4.2.0.31
Class Identifier	{386D0EB0-4F01-4DDA-95A9-90129B2400DA}
Image File	BEHRemoteData.ocx
Property Initializations	none
Serializable Properties	CAPTION=TEXT, COLOR=COLOR
Required Files	none
Security Keys	none
Multiple Instances Allowed	no
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BEHO*1.1*020001

For information on how to configure remote data views, refer to the Computerized Patient Record System (CPRS) Technical Manual published by the Department of Veterans Affairs.

### 45.3 Routine Descriptions

None.

### 45.4 File List

None.

### 45.5 Cross References

None.

## 45.6 Exported Options

None.

## 45.7 Exported Security Keys

None.

## 45.8 Exported Protocols

None.

## 45.9 Exported Parameters

None.

## 45.10 Exported Mail Groups

None.

## 45.11 Callable Routines

None.

## 45.12 External Relations

Entity	Name	Description
Package	CLINICAL INFO RESOURCE NETWORK 1.0	Uses supported APIs.
Package	MASTER PATIENT INDEX VISTA 1.0	Uses supported APIs.

## 45.13 Internal Relations

Entity	Name	Description
Component	Remote Sites Service	Uses supported APIs.

## 45.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 45.15 Components

This component supports the following properties and methods:

### 45.15.1 Properties

The properties are described in the following table.

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
AUTOSIZE	Boolean	RW	If true, the component automatically resizes itself to accommodate its contents.
BORDERSTYLE	Enum	RW	Sets the style of the border surrounding the component. Can be one of: 0 = None 1 = Single 2 = Sunken 3 = Raised
CAPTION	String	RW	Sets the text displayed in the button caption.
COLOR	Color	RW	Sets the background color of the component.
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
GLYPH	String	RW	Name of bitmap file containing glyphs to be displayed on button surface.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPPFILE	String	RW	Sets the name of the help file associated with the component.
LAYOUT	String	RW	Property representing the internal layout of the form.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
NUMGLYPHS	Integer	RW	Number of glyphs contained in glyph file.
TITLE	String	RW	Title for the report.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 46.0 Remote Sites Service

### 46.1 Introduction

The Remote Sites Service provides access to patient data that can reside at locations other than the local facility.

### 46.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHREMOTEVIEWS.REMOTESITES
Version	4.2.0.98
Class Identifier	{CE0E8D76-4597-467D-B942-6D52838D65E4}
Image File	BEHRemoteViews.dll
Property Initializations	none
Serializable Properties	none
Required Files	none
Security Keys	none
Multiple Instances Allowed	no
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	yes
.Net Component	no
Associated Build	BEHO*1.1*020001

For information on how to configure remote data views, refer to the Computerized Patient Record System (CPRS) Technical Manual published by the Department of Veterans Affairs.

### 46.3 Routine Descriptions

This component has been assigned the namespace designation of “BEHORDV.” The following routines are distributed:

Routine	Description
BEHORDV	Remote sites support.

### 46.4 File List

None.

### 46.5 Cross References

None.

### 46.6 Exported Options

None.

## 46.7 Exported Security Keys

None.

## 46.8 Exported Protocols

None.

## 46.9 Exported Parameters

None.

## 46.10 Exported Mail Groups

None.

## 46.11 Callable Routines

This section describes supported entry points for routines exported with this component.

### 46.11.1 RPC: BEHORDV DIRECT

Scope: private.

Parameter	Datatype	Description
LOC	String	Destination institution name or IEN
RPC	String	RPC name
RPCVER	String	RPC version
P1...P10	Any	Up to 10 parameters to be passed to the remote call.
<return value>	String List	The return result of the remote procedure call.

Calls a RPC at a remote site.

## 46.12 External Relations

Entity	Name	Description
Package	CLINICAL INFO RESOURCE NETWORK 1.0	Uses supported APIs.
Package	MASTER PATIENT INDEX VISTA 1.0	Uses supported APIs.
Package	RPC BROKER 1.1	Uses supported APIs.

## 46.13 Internal Relations

None.

## 46.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 46.15 Components

This component supports the following properties and methods:

### 46.15.1 IRemoteSites

#### 46.15.1.1 Properties

Property	Datatype	Access	Description
NoDataReason	String	R	Text giving reason why data was not returned.
RemoteDataExists	Boolean	R	True if remote data is present.
Selected	Boolean	R	True if at least one site has been selected.
SiteCount	Integer	R	Count of sites in Sites array.
Sites	IRemoteSite (array)	R	Array of remote sites.

#### 46.15.1.2 CallRemote

Scope: public.

Parameter	Datatype	Description
RPCName	String	Remote procedure name.
Params	Variant	Parameters for RPC call.
Callback	IRemoteCallback	Interface reference for callback.
<return data>	String	Unique signature for this call.

Queues a remote RPC for execution.

#### 46.15.1.3 GetSiteByID

Scope: public.

Parameter	Datatype	Description
AnID	String	Site identifier
<return value>	IRemoteSite	Returns a reference to the site matching the specified identifier, or null if none found.

Looks up the specified site identifier, returning an IRemoteSite interface reference if a match is found.

#### 46.15.1.4 Reset

Scope: public.

Performs a Reset operation on each site in the Sites array.

#### 46.15.1.5 Select

Scope: public.

Parameter	Datatype	Description
<return value>	Integer	Count of selected sites.

Invokes the Remote Site Selector dialog.

## 46.15.2 IRemoteSite

### 46.15.2.1 Properties

Property	Datatype	Access	Description
LastDate	Date/Time	R	Date of the last available data for the site.
ReportCount	Integer	R	Count of reports in the Reports array.
Reports	IRemoteReport (array)	R	Array of reports available for this site.
Selected	Boolean	RW	True if site is selected.
SiteId	String	R	Returns the identifier for the site.
SiteName	String	R	Returns the name of the site.

### 46.15.2.2 CallRemote

Scope: private.

Parameter	Datatype	Description
RPCSignature	String	Unique signature identifying the report.
Callback	IRemoteCallback	Interface reference to call when report is available.

Queues a remote RPC for execution.

### 46.15.2.3 GetReportBySignature

Scope: private.

Parameter	Datatype	Description
RPCSignature	String	Unique signature identifying the report.
<result value>	IRemoteReport	Reference to the report identified by the RPCSignature, or null if none found.

Looks up a report by its unique signature.

### 46.15.2.4 Reset

Scope: private.

Removes all reports from the Reports array.

## 46.15.3 IRemoteReport

### 46.15.3.1 Properties

The properties are described in the following table.

Property	Datatype	Access	Description
Handle	Integer	R	Unique handle associated with the report request.
RemoteHandle	String	R	Handle assigned by the site of origin for this report.
Signature	String	R	Unique signature identifying this report.
SiteId	String	R	Identifier of the site of origin.



Property	Datatype	Access	Description
Status	Enum	R	Status of the report. One of: 0 = Initial 1 = Pending 2 = Complete 3 = Error 4 = Aborted
Text	String	R	Text of the report.

#### 46.15.4 IRemoteReport2

##### 46.15.4.1 CallRemote

Scope: private.

Parameter	Datatype	Description
Callback	IRemoteCallback	Interface reference to call when request is complete.

Invokes the remote data call as an asynchronous RPC.

#### 46.15.5 IRemoteCallback

##### 46.15.5.1 RemoteResults

Scope: private.

Parameter	Datatype	Description
Report	IRemoteReport	Reference to the report that is now available.

Called when a requested report becomes available.

## 47.0 Reports (CPRS)

### 47.1 Introduction

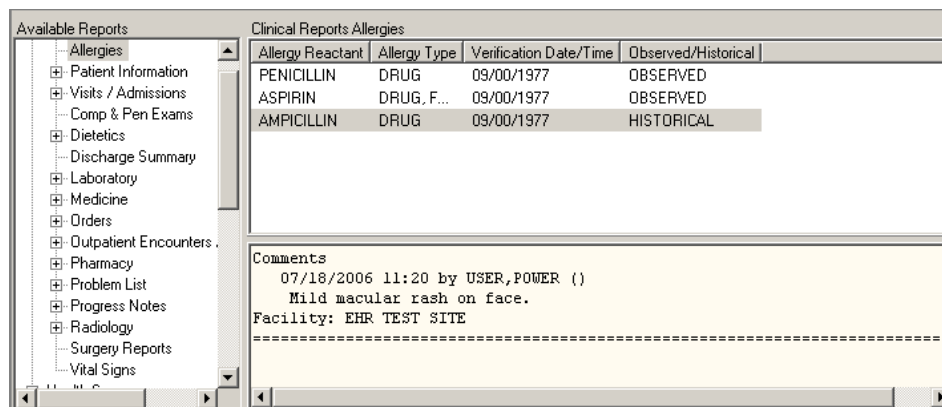


Figure 47-1: Sample Reports

Permits displaying and printing of a multitude of predefined reports.

### 47.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHREPORTS.REPORTS
Version	20.1.0.18
Class Identifier	{C3C01F1D-009D-497F-86FE-2BEDF6E9F4A3}
Image File	BEHReports.ocx
Property Initializations	none
Serializable Properties	AUTOSIZE=BOOL, BORDERSTYLE=ENUM, CAPTION=TEXT, CAPTIONCOLOR1=COLOR, CAPTIONCOLOR2=COLOR, CAPTIONSTYLE=ENUM, COLOR=COLOR
Required Files	BEHReports.chm
Security Keys	none
Multiple Instances Allowed	no
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BEHO*1.1*021001

There are no specific implementation or maintenance tasks associated with this component.

### 47.3 Routine Descriptions

None.

## 47.4 File List

None.

## 47.5 Cross References

None.

## 47.6 Exported Options

Option	Type	Description
BEHORP MAIN	menu	Report configuration menu.
BEHORP PRINT FORMAT	action	This option allows the user to define formats for printing labels and requisitions for orders.
BEHORP REPORT SYSTEM PARAM	action	System display parameters.
BEHORP REPORT USER PARAM	action	User display parameters.
BEHORPHS IHS ACT/INACT HMR	action	Activate/inactivate a health maintenance reminder.
BEHORPHS IHS CINQ	action	From the "B" index on HEALTH SUMMARY COMPONENT, displays available components from which a summary can be built.
BEHORPHS IHS DISPLAY HMR	action	Display one health maintenance reminder desc.
BEHORPHS IHS FIDEL	edit	Uses the APCH DELETE HS FLOWSHEET ITEM template to delete an entry from the HEALTH SUMMARY FLOWSHEET ITEM file.
BEHORPHS IHS FIED	edit	Uses the APCH EDIT HS FLOWSHEET ITEM template to create or modify a flowsheet item definition in the HEALTH SUMMARY FLOWSHEET ITEMS file.
BEHORPHS IHS FIINQ	action	From the "B" index on HEALTH SUMMARY FLOWSHEET ITEMS, displays available components from which a flowsheet can be built.
BEHORPHS IHS FSDEL	edit	Uses the APCH DELETE HLTH SUM FLOWSHEET template to delete an entry from the HEALTH SUMMARY FLOWSHEET file.
BEHORPHS IHS FSED	edit	Uses the APCH EDIT HLTH SUM FLOWSHEET template to create or modify a flowsheet definition in the HEALTH SUMMARY FLOWSHEET file.
BEHORPHS IHS FSINQ	action	From the "B" index on HEALTH SUMMARY FLOWSHEETS, displays available components from which a summary can be built.
BEHORPHS IHS HMIP	action	Prints all items with their associated descriptions in the health maintenance item file.

Option	Type	Description
BEHORPHS IHS HMR LOCAL SPEC	action	Add/modify locally defined reminder criteria.
BEHORPHS IHS INQ	inquire	An “inquire” option against the HEALTH SUMMARY TYPE dictionary to allow the user to view the structure of a summary type.
BEHORPHS IHS MAIN	menu	Submenu for health summary maintenance functions: - edit HEALTH SUMMARY COMPONENT file, - edit HEALTH SUMMARY TYPE file
BEHORPHS IHS MDEL	edit	Invokes APCH DEL HLTH SUM MEAS PANEL template to delete health summary measurement panel definitions (HEALTH SUMMARY MEASUREMENT PANELS file).
BEHORPHS IHS MED	edit	Invokes APCH EDIT HLTH SUM MEAS PANEL template to allow creation or editing of measurement panels (prototype definitions which control the content and ordering of a panel of measurements).
BEHORPHS IHS MENU HEALTH MAIN	menu	Health maintenance reminders menu.
BEHORPHS IHS MINQ	action	Displays measurement panel types from “B” index on HEALTH SUMMARY MEAS PANEL file.
BEHORPHS IHS SITE PARAMETER	edit	Update health summary site parameters.
BEHORPHS IHS SUM	action	Invokes APCHS to generate health summary, either printed or CRT display version. Requires presence of the following dictionaries: HEALTH SUMMARY TYPE, HEALTH SUMMARY COMPONENT, HEALTH SUMMARY MEASUREMENT PANELS (if measurement panels used).
BEHORPHS IHS TDEL	edit	Invokes APCH DELETE SUMMARY TYPE template to delete a health summary type. A separate option (APCHSTED) is available to edit summary types.
BEHORPHS IHS TED	action	Invokes APCH EDIT HEALTH SUMMARY TYPE template to create/edit a health summary type. A separate option (APCHSTDEL) is available to delete a health summary type.
BEHORPHS IHS TED FM	edit	Invokes APCH EDIT HEALTH SUMMARY TYPE template to create/edit a health summary type. A separate option (APCHSTDEL) is available to delete a health summary type.
BEHORPHS IHS TINQ	action	Displays health summary types from “B” index on HEALTH SUMMARY TYPE file.

Option	Type	Description
BEHORPHS IHS TYPES	action	This specifies the IHS health summaries that are to appear in the reports component. This is only used if the ORWRP HEALTH SUMMARY LIST ALL parameter is set to false.
BEHORPHS LIST ALL	action	Set to YES to list all health summaries. Set to NO to list only those health summaries specified in the ORWRPBHS HEALTH SUMMARY LIST and ORWRP HEALTH SUMMARY TYPE LIST parameters.
BEHORPHS MAIN	menu	Health summary configuration menu.
BEHORPHS VHA ADHOC EDIT	action	Enter/edit the default parameters (Time and Occurrence limits, Hospital Location displayed, ICD Text displayed, Provider Narrative displayed, Selection Items, and Header names) for components of the Ad Hoc Health Summary Type. Components can also be deleted from this option.
BEHORPHS VHA ADHOC LOAD	action	Reloads the Ad Hoc Health Summary Type to include ALL Components, including any defined by the site, and optionally all temporarily DISABLED components, sequenced alphabetically by name.
BEHORPHS VHA BUILD MENU	menu	This menu contains options to create or delete Health Summary Types, along with options which can help in that process.
BEHORPHS VHA COMP DESC LIST	print	Lists all components which can be used to define Health Summary Types, along with the abbreviation and a brief description of each.
BEHORPHS VHA COMP EDIT	action	Create/modify new Health Summary components, either by duplication and renaming of existing components, or by entering all appropriate fields for a component programmed on-site.
BEHORPHS VHA COMP INQ	action	This allows the user to display the characteristics of a Health Summary Component.
BEHORPHS VHA COMP LIST	print	Lists all components which can be used to define Health Summary Types, along with several component characteristics.
BEHORPHS VHA ENABLE/ DISABLE	edit	Selectively enable or disable Health Summary Components. Out of order messages can be entered and the disable action can now be identified as being either temporary or permanent.

Option	Type	Description
BEHORPHS VHA GUI HS LIST DEFAU	action	Use this option to display the list of Health Summary Types that will be contained in the 'Health Summary Types' section on the Reports Tab in CPRS. The display includes the precedence of parameters, the method for building the list, and finally the list that will actually appear on the Reports Tab in CPRS.
BEHORPHS VHA GUI HS LIST METHO	action	Use this option to edit the method used to build the list of Health Summary Types displayed on the Reports component of the EHR. There are two methods:  Overwrite: System-defined Health Summary Types are added to the list. If there are User-defined Health Summary Types, then they will replace the System-defined Health Summary Types already on the list.  Append: System-defined Health Summary Types are added to the list. If there are User-defined Health Summary Types, then they will be added to the list along with the System-defined types.
BEHORPHS VHA GUI HS LIST PRECE	action	Use this option to select the defined Health Summary Types to include on the list and arrange them in the order (precedence) that they should appear on the list. The three most common groups of Health Summary Types are:  SYS: Health Summary Types can be defined for all users on the system.  USR: Health Summary Types can be defined for a single user on the system.  NAT: Nationally exported Health Summary Types (NAT) defined for system users or a single user are treated independently to keep them together in the list box. In this option, you must first select those Health Summary Type groups (SYS, USR, NAT, etc.) to include on the Reports Tab, then specify the order in which they will appear on the Reports Tab.
BEHORPHS VHA GUI REPORTS LIST	menu	Use the options in the CPRS Reports Tab Health Summary Types List Menu to select the Health Summary Types to list on the Reports Tab, arrange the order of these Health Summaries on the list, and to view the user s preferences.
BEHORPHS VHA HS ADHOC	action	Generates an 'Ad Hoc' Health Summary for specified patients. Instead of selecting a pre-defined Health Summary Type, the user defines his own ad hoc Health Summary structure for temporary use while using this option. The user selects Health Summary components, time and occurrence limits when applicable, and selection items when applicable.

Option	Type	Description
BEHORPHS VHA HS BY LOC	action	Allows user to print health summaries interactively for all patients on a specified ward(s), for all patients with appointments at a specified outpatient clinic(s), or for all patients scheduled for a specified operating room(s) on a selected day or range of days.
BEHORPHS VHA HS BY LOC PARAMET	action	Allows the user to set-up Health Summary Types to be batch printed nightly for all patients at a specified hospital location. Location can be a ward or a clinic. If location is a clinic, the user is asked to specify 'Print Days Ahead.' Health Summaries will then be printed for all patients with appointments that many days in the future. To generate summaries, this option requires that option GMTS TASK STARTUP be queued to run nightly. Summaries should then be printed during the night and be ready for distribution by early morning. Thus, 0 Print Days Ahead means summaries should be ready to distribute by early morning of day of clinic appointment. For ease in separating printouts queued to the same device, a location banner appears in front of each locations Health Summaries.
BEHORPHS VHA HS BY PATIENT	action	Generates a Health Summary of a specified pre-defined Health Summary Type for a specified patient.
BEHORPHS VHA HS BY PATIENT & D	action	Generates Health Summaries of a specified pre-defined Health Summary Type for multiple patients. After patients are selected, the user can pick a date range. Data for summaries is based on the date range. This date range overrides Time Limits for components which allow this option.
BEHORPHS VHA HS BY PATIENT & V	action	Generates Health Summaries of a specified pre-defined Health Summary Type for multiple patients. After a patient is selected, the user can pick an outpatient visit (based on the VISIT File) or an inpatient hospital admission (based on the Patient Movement file). If the Patient Care Encounter (PCE) package isn't installed then the user can only choose an inpatient visit. Data for summaries is based on a date range from the outpatient visit or inpatient admission. This Date Range overrides Time Limits for components which allows this option.

Option	Type	Description
BEHORPHS VHA HS FOR ALL CLINIC	action	Allows user to interactively designate the batch print of all health summaries for patients with appointments at all outpatient clinics on a selected day. Summaries will be printed at the print device designated for that clinic in the Location Parameters. This option is an alternate method to designate when summaries should be printed for all clinics. The nightly batch job that prints summaries for wards, clinics and operating rooms probably should be disabled if the user wants to use this option on a daily basis.
BEHORPHS VHA HS MENU	menu	This menu includes all of the various print options available for Health Summaries, allowing the user to generate Health Summaries by Patient, by Patient and a date range, by patient and a outpatient visit or an admission, by Location, for patients at all clinics, and on an Ad Hoc basis.
BEHORPHS VHA INFO ONLY MENU	menu	This menu contains options which provide information only about Health Summary Types, Components, Measurement Panels, and Reminder/Maintenance Items.
BEHORPHS VHA MAIN	menu	This option is used by individuals who have the capability to build their own Health Summary Types. It has all of the capabilities of the 'Health Summary Users Menu' with the addition of the 'Build Health Summary Type Menu' and the option to set parameters for nightly batch printing of Health Summaries by Location.
BEHORPHS VHA MAINT MENU	menu	This option will provide the IRM staff with a set of tools for Health Summary implementation and maintenance
BEHORPHS VHA OBJ DELETE	action	This option is used to delete a Health Summary Object that you created. You cannot delete Health Summary Objects created by other people.
BEHORPHS VHA OBJ ENTER/EDIT	action	This option allows the user to create or modify a Health Summary Object. Objects are Health Summaries that can have their headers suppressed so they can be inserted into another document.
BEHORPHS VHA OBJ EXPORT	action	This option will allow a user to export a Health Summary Object, and its corresponding Health Summary Type to a Mailman message which can be sent to other accounts/sites for import.
BEHORPHS VHA OBJ EXPORT/IMPORT	menu	This menu contains options used to either export a Health Summary Object to a Mailman message, or import a Health Summary Object by reading the data unpacked from a Mailman message using the Packman Utilities.



Option	Type	Description
BEHORPHS VHA OBJ IMPORT/ INSTAL	action	This option is used to install a Health Summary Object exported using the GMTS OBJ EXPORT option. The incoming Object must first be unpacked from the incoming Mailman message using Packman utilities. Then this option is used to install the object into the Health Summary files. If there is a Health Summary Type with the same name as the Health Summary being imported, then you will be prompted to either use the existing Health Summary Type, or to rename the incoming Health Summary Type. At no time will an existing Health Summary Type be over written.
BEHORPHS VHA OBJ INQ	action	Display a Health Summary Object from file 142.5, to include an example of the object.
BEHORPHS VHA OBJ MENU	menu	Use this option to access utilities to add/edit a Health Summary Object, display the characteristics of a Health Summary Object, to test a Health Summary Object or to export/import a Health Summary Object.
BEHORPHS VHA OBJ TEST	action	This option allows the user test a Health Summary Object by displaying the Object to the screen exactly as it would appear inserted in another document.
BEHORPHS VHA PARAMETER EDIT	edit	This option allows IRM staff or Health Summary ADPAC to edit the Health Summary Parameters, to modify the behavior of Health Summary for your site.
BEHORPHS VHA SITE ADD/EDIT LIS	action	Use this option to add, edit or delete a Health Summary Type from the list of Health Summary Types defined for a given entity (System, Division, User, etc.).
BEHORPHS VHA SITE COMPILE METH	action	<p>Use this option to modify how the list of Health Summary Types is to be built for the site. This method will be used for all users who do not have user preferences set.</p> <p>Overwrite: System-defined Health Summary Types are added to the list. If there are User-defined Health Summary Types, then they will replace the System-defined Health Summary Types already on the list.</p> <p>Append: System-defined Health Summary Types are added to the list. If there are User-defined Health Summary Types, then they will added to the list along with the System-defined types.</p>
BEHORPHS VHA SITE DEFAULTS	menu	Use the options in the CPRS Health Summary Display/Edit Site Defaults Menu to display the site defaults, select the Health Summary Types to list on the Reports Tab, edit the method of building the list (append/overwrite), edit allowable entities for the list (i.e., User, System, Division, etc.), or resequence the allowable entities in the order they should be appear on the list.

Option	Type	Description
BEHORPHS VHA SITE DISPLAY DEFA	action	Use this option to display the site defaults for building the list of Health Summary Types on the Reports Tab in CPRS. The display includes the method for building the list and the precedence of parameters. These default values will be used in the event that the user has not established user preferences.
BEHORPHS VHA SITE PRECEDENCE	action	Use this option to add or delete an entity (System, Division, User, etc.) from the list of allowable entities for the parameter 'ORWRP HEALTH SUMMARY TYPE LIST' used on the CPRS Reports Tab to display the Health Summary Types List.
BEHORPHS VHA SITE RESEQUENCE	action	Use this option to resequence the order in which the allowable entities for parameter 'ORWRP HEALTH SUMMARY TYPE LIST' are used in building the list of Health Summary Types for the CPRS Reports Tab. This order is used for any user who does not have user preferences set.
BEHORPHS VHA TASK LOCATIONS LI	print	This option displays a list of all locations which will be checked to see if health summaries should be printed in the nightly run of GMTS TASK STARTUP.
BEHORPHS VHA TYPE DELETE	edit	This option allows the owner of a Health Summary Type to delete it. In addition anyone who holds the GMTSMGR key can use this option. Also, if a Health Summary Type has a lock, anyone who holds a key with that lock value can delete that Health Summary Type. This Option will not allow the deletion of a Health Summary Type in use by a Health Summary Object. It will not allow the deletion of special case Health Summary Types such as GMTS HS ADHOC OPTION, or REMOTE Health Summary Types.
BEHORPHS VHA TYPE ENTER/ EDIT	action	This option allows the user to create or modify a Health Summary Type. New types created under this option will automatically define the creator as the owner unless creator holds the GMTSMGR security key in which case GMTSMGR-holder can designate owner of his choice. Normally Health Summary Types can be modified only by their owner. Exceptions are: someone who holds the GMTSMGR key can modify any Health Summary Type. If a Health Summary Type has a lock, someone who holds a security key with that lock value can modify that Health Summary Type. Use the 'Delete Health Summary Type' option to delete the Health Summary Type.

Option	Type	Description
BEHORPHS VHA TYPE INQ	action	This allows the user to display the current definition of a Health Summary Type. The display includes the sequence of components, and occurrence and time limits and selection items when they apply to a component.
BEHORPHS VHA TYPE LIST	print	This option displays a list of all Health Summary Types currently defined, with owner and lock information related to each summary type.
BEHORPHS VHA TYPE RESEQUENCE	action	Resequence the components of a given Health Summary Type from 5, in increments of 5 (i.e., 5, 10, 15, 20, ..., 5n where n is the total number of components).
BEHORPHS VHA TYPES	action	This specifies the VistA health summaries that are to appear in the reports component. This is only used if the ORWRP HEALTH SUMMARY LIST ALL parameter is set to false.
BEHORPPA MAIN	menu	Report parameters menu.
BEHORPPA TIME/OCC LIMITS ALL	action	Default time and occurrence limits for all reports.
BEHORPPA TIME/OCC LIMITS INDV	action	Default time and occurrence limits by report.

## 47.7 Exported Security Keys

None.

## 47.8 Exported Protocols

None.

## 47.9 Exported Parameters

None.

## 47.10 Exported Mail Groups

None.

## 47.11 Callable Routines

None.

## 47.12 External Relations

Entity	Name	Description
Package	ORDER ENTRY/RESULTS REPORTING 3.0	Uses supported APIs.

## 47.13 Internal Relations

Entity	Name	Description
Component	CPRS Support Library	Uses supported APIs.

## 47.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 47.15 Components

This component supports the following properties and methods:

### 47.15.1 Properties

The properties are described in the following table.

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
AUTOSIZE	Boolean	RW	If true, the component automatically resizes itself to accommodate its contents.
BORDERSTYLE	Enum	RW	Sets the style of the border surrounding the component. Can be one of: 0 = None 1 = Single 2 = Sunken 3 = Raised
CAPTION	String	RW	Sets the text displayed in the title bar. To justify portions of the caption text, use the “\” character to delimit the left-, center-, and right-justified portions of the caption text.

Property	Datatype	Access	Description
CAPTIONCOLOR1 CAPTIONCOLOR2	Color	RW	Colors to apply to the title bar. If the two colors differ and a gradient style is set, a gradient effect is created. For a standard title bar style, only the first color is applied.
CAPTIONSTYLE	Enum	RW	Sets the caption style. Can be one of: 0 = None – No caption (hides title bar) 1 = Title – Standard title bar 2 = Frame – Framed title bar (group box style) 3 = Left – Left gradient title bar 4 = Right – Right gradient title bar 5 = Center – Center gradient title bar
COLOR	Color	RW	Sets the background color of the component.
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPPFILE	String	RW	Sets the name of the help file associated with the component.
LAYOUT	String	RW	Property representing the internal layout of the form.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 48.0 Triage Summary

### 48.1 Introduction

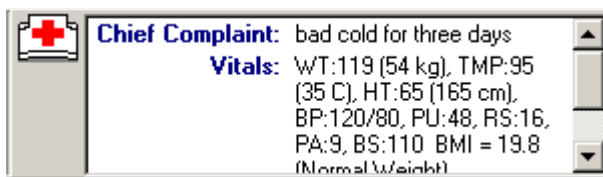


Figure 48-1: Sample Triage Summary

The Triage Summary presents a succinct overview of clinical data associated with the current visit.

### 48.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	IHSBGOTRIAGESUMMARY.BGOTRIAGESUMMARY
Version	1.1.0.147
Class Identifier	{DBD9C585-4B49-48CB-836D-CDDA607CCA9D}
Image File	IhsBgoTriageSummary.ocx
Property Initializations	none
Serializable Properties	ShowVitalsOnly=BOOL, ShowCCAuthor=BOOL
Required Files	IhsBgoTriageSummary.chm
Security Keys	none
Multiple Instances Allowed	yes
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BGO*1.1*3

There are no specific implementation or maintenance tasks associated with this component.

### 48.3 Routine Descriptions

This component has been assigned the namespace designation of "BGOTRG." The following routines are distributed:

Routine	Description
BGOTRG	Triage summary support.

### 48.4 File List

None.

## 48.5 Cross References

None.

## 48.6 Exported Options

None.

## 48.7 Exported Security Keys

None.

## 48.8 Exported Protocols

None.

## 48.9 Exported Parameters

None.

## 48.10 Exported Mail Groups

None.

## 48.11 Callable Routines

This section describes supported entry points for routines exported with this component.

### 48.11.1 RPC: BGOTRG GETSUM

Scope: private.

Parameter	Datatype	Description
INP	Pointer (#9000010)	Specified as: Visit IEN ^ Provider ^ Report List ^ Include CC Author
<return value>	String List	String containing list of entries to be displayed in triage summary.

Returns a list of entries associated with the specified visit.

## 48.12 External Relations

Entity	Name	Description
File	MEASUREMENT TYPE (#9999999.07)	Read access
File	V MEASUREMENT (#9000010.01)	Read access

<b>Entity</b>	<b>Name</b>	<b>Description</b>
File	ORDER (#100)	Read access
File	IMMUNIZATION (#9999999.14)	Read access
File	V IMMUNIZATION (#9000010.11)	Read access
File	SKIN TEST (#9999999.28)	Read access
File	V SKIN TEST (#9000010.12)	Read access
File	EDUCATION TOPICS (#9999999.09)	Read access
File	V PATIENT ED (#9000010.16)	Read access
File	EXAM (#9999999.15)	Read access
File	V EXAM (#9000010.13)	Read access
File	HEALTH FACTORS (#9999999.64)	Read access
File	V HEALTH FACTORS (#9000010.23)	Read access
File	CPT (#81)	Read access
File	V CPT (#9000010.18)	Read access
File	REPRODUCTIVE FACTORS (#9000017)	Read access
File	NARRATIVE TEXT TYPE (#9999999.89)	Read access
File	V NARRATIVE (#9000010.34)	Read access

### 48.13 Internal Relations

None.

### 48.14 Archiving and Purging

There are no archiving or purging requirements within this software.

### 48.15 Components

This component supports the following properties and methods:



Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
SHOWCCAUTOR	Boolean	RW	If true, displays the chief complaint author.
SHOWVITALSONLY	Boolean	RW	If true, only vital measurements are displayed.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 49.0 Patient Detail View

### 49.1 Introduction



Figure 49-1: Patient Detail View Button

The Patient Detail View component permits ready access to a customizable report summarizing important patient information.

### 49.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHPTDETAIL.PTDETAIL
Version	4.2.1.3
Class Identifier	{DE053ED3-6390-455F-AE98-B43820D726F6}
Image File	BEHPtDetail.ocx
Property Initializations	TITLE=Detail for \$(patient.name), RPCNAME=BEHOPTCX PTINQ
Serializable Properties	CAPTION=TEXT, COLOR=COLOR
Required Files	
Security Keys	none
Multiple Instances Allowed	yes
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BEHO*1.1*010001

There are no specific implementation or maintenance tasks associated with this component.

### 49.3 Routine Descriptions

None.

### 49.4 File List

None.

### 49.5 Cross References

None.

## 49.6 Exported Options

None.

## 49.7 Exported Security Keys

None.

## 49.8 Exported Protocols

None.

## 49.9 Exported Parameters

None.

## 49.10 Exported Mail Groups

None.

## 49.11 Callable Routines

None.

## 49.12 External Relations

None.

## 49.13 Internal Relations

None.

## 49.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 49.15 Components

This component supports the following properties and methods:

### 49.15.1 Properties

The properties are described in the following table:

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ALLOWPRINT	Boolean	RW	If true, a print button appears on the report dialog allowing printing of the report.
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
AUTOSIZE	Boolean	RW	If true, the component automatically resizes itself to accommodate its contents.
BORDERSTYLE	Enum	RW	Sets the style of the border surrounding the component. Can be one of: 0 = None 1 = Single 2 = Sunken 3 = Raised
CAPTION	String	RW	Sets the text displayed in the button caption.
COLOR	Color	RW	Sets the background color of the component.
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
GLYPH	String	RW	Name of bitmap file containing glyphs to be displayed on button surface.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPPFILE	String	RW	Sets the name of the help file associated with the component.
LAYOUT	String	RW	Property representing the internal layout of the form.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
NUMGLYPHS	Integer	RW	Number of glyphs contained in glyph file.
RPCNAME	String	RW	Name of the remote procedure that generates the patient detail report.
TITLE	String	RW	Title for the report.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 50.0 Notifications

### 50.1 Introduction

Patient	Notification	Delivered
STEINBERG,TL...	Order requires electronic signature.	02-May-2007 09:56
CHEE, EFFIE (1...	Order requires electronic signature.	25-May-2005 10:35
DEMO,FEMALE ...	Critical lab: HGB 5.4 03/24 10:19	24-Mar-2004 10:23
DEMO,FEMALE ...	Critical lab: K 6.8 01/01/41	24-Mar-2004 10:23
DEMO,FEMALE ...	Critical lab: NA 100 01/01/41	24-Mar-2004 11:37
DEMO,FEMALE ...	Critical lab: PLT 5 03/24 10:19	24-Mar-2004 11:39
DEMO,FEMALE ...	Critical lab: K 8.5 03/24 11:42	24-Mar-2004 11:42
DEMO,FEMALE ...	Critical labs - [POTASSIUM]	24-Mar-2004 11:42
DEMO,FEMALE ...	Order requires electronic signature.	20-Jun-2007 10:17
STEINBERG,TL...	UNSIGNED NOTE WITH BOILERPLATE available for SIGNATURE.	02-May-2007 12:41
CHICK,FREDERI...	UNSIGNED ADULT CARE HIGH RISK SCREENING INTAKE FORM avail...	20-Sep-2004 15:01
ABBEY,ROBERT...	UNSIGNED NOTE WITH BOILERPLATE available for SIGNATURE.	15-Mar-2006 15:57
DEMO,FEMALE ...	Imaging Results: ECHOCARDIOGRAM M-MODE &/OR REA	09-Apr-2004 15:07
DEMO,FEMALE ...	UNSIGNED Discharge Summary available for SIGNATURE.	20-Jun-2006 14:31
DEMO,FEMALE ...	UNSIGNED Discharge Summary available for SIGNATURE.	14-May-2007 13:08
DEMO,FEMALE ...	Completed Consult PROSTHETICS REQUEST	18-Jun-2007 09:21
DEMO,FEMALE ...	Completed Consult HOME OXYGEN REQUEST	19-Jun-2007 12:26

Figure 50-1: Sample Notifications

The Notifications component permits viewing and managing notifications.

### 50.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHNOTIFICATIONS.NOTIFICATIONS
Version	5.1.2.45
Class Identifier	{EA5178CE-9C27-4177-A992-3D7B01897965}
Image File	BEHNotifications.ocx
Property Initializations	ALERTTHRESHOLD=@BEHOXQ ALERT THRESHOLD, SHOWALL=@BEHOXQ SHOW ALL
Serializable Properties	AUTOSIZE=BOOL, BORDERSTYLE=ENUM, CAPTION=TEXT, CAPTIONCOLOR1=COLOR, CAPTIONCOLOR2=COLOR, CAPTIONSTYLE=ENUM, COLOR=COLOR, LEGEND=ENUM, LAYOUT=HIDDEN
Required Files	BEHNotifications.chm
Security Keys	none
Multiple Instances Allowed	no
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BEHO*1.1*002002

There are no specific implementation or maintenance tasks associated with this component.

## 50.3 Routine Descriptions

This component has been assigned the namespace designation of “BEHOXQ.” The following routines are distributed:

Routine	Description
BEHOXQ	Notification support.
BEHOXQIN	Installation support.
BEHOXQPC	PCC notifications.

## 50.4 File List

This component has been assigned the file number range of 90460.021 through 90460.029. The following files are distributed:

### 50.4.1 BEH ALERT CONTROL (#90460.021)

This file contains control information for the display and processing of notifications.

Field Name	#	Datatype	Indexes	Description
NAME	.01	String	B – Standard	Descriptive name for this notification class.
IDENT	1	M Code		Checks the notification identifier in variable AID and sets \$T to true if a match.
PARSE	2	M Code		Code to parse standard data elements from a notification. Elements should be returned in the ALR array as follows: ALR("DFN")=Patient IEN (if patient-orientation) ALR("INF")=Info only flag ALR("TYP")=Notification type code ALR("PRI")=Priority code ALR("XTR")=Additional data
PROCESS	3	M Code		Post processing logic for a notification.

### 50.4.2 BEH ALERT SCHEDULING (#90460.022)

This file contains notifications scheduled to be delivered at a later time.

Field Name	#	Datatype	Indexes	Description
DATETIME	.01	Date/Time	B – Standard	The date and time the notification is to be delivered.
ID	1	Text		The ID of the notification.
SENDER	2	Pointer (#200)	C – Standard	IEN of the sender.
SUBJECT	5	Text		Subject line for the notification.
DATA	6	Text		Control data associated with the notification.
RECIPIENT	10	Subfile (#90460.0221)		Intended recipients for this notification.
MESSAGE	20	Word Processing		Message text associated with the notification.

#### 50.4.2.1 RECIPIENT (#90460.0221)

This subfile contains the intended recipients of a scheduled notification.

Field Name	#	Datatype	Indexes	Description
RECIPIENT	.01	Variable Pointer (#200, 3.8)	B – Standard	The individual or mail group to receive the notification.

## 50.5 Cross References

Cross references are described in the preceding section.

## 50.6 Exported Options

Option	Type	Description
BEHOXQ ALERT RECIPIENTS	action	This option prompts for a patient then displays all Kernel Alerts for that patient. You are then prompted to select one or more alerts. Recipients of those alerts are then displayed or printed along with relevant data regarding how they processed the alert. This option uses the Alert Tracking file. If an alert has an associated order number, it is displayed in brackets.
BEHOXQ ALERT THRESHOLD	action	Priority Threshold for Popup Alerts
BEHOXQ ARCHIVE PERIOD	action	Enter the number of days to archive this notification before deletion. The number of days to archive a notification for a site. If not indicated, the default period of 30 days is used. The maximum number of 100,000 is about 220 years. This value is passed to the Kernal Alert Utility where the actual archiving and deletion of alerts/notifications occurs.
BEHOXQ DEFAULT RECIPIENT DEV	action	Enter 'Yes' if the device should always receive this notification. Devices will receive the notification on a regular basis - despite setting in the option Enable/Disable Notifications. These devices will always receive the notification, regardless of patient.
BEHOXQ DEFAULT RECIPIENTS	action	Enter 'Yes' if the recipient should always receive this notification. Recipients (users,teams) will receive the notification on a regular basis - despite settings in the option Enable/Disable Notifications. These recipients will always receive the notification, regardless of patient.

Option	Type	Description
BEHOXQ DELETE MECHANISM	action	<p>Enter 'I' if deleted on individual review/action; 'A' for all recipients. Set of codes used to determine how a notification will be deleted at a site. Codes include:</p> <p>I (Individual Recipient): delete the notification for an individual recipient when</p> <p>a) that individual completes the follow-up action on notifications with associated actions.</p> <p>A (All Recipients): delete the notification for all recipients when</p> <p>a) any recipient completes the follow-up action on notifications with follow-up actions.</p> <p>b) any recipient reviews notifications without follow-up actions.</p>
BEHOXQ DETERMINE RECIPIENTS	action	<p>Based on Notification, Patient and optional Order Number and Default/ Regular Recipients, a list of recipients and potential recipients is presented. This is a tool for IRM or Clinical Coordinators only. It will enable the user to determine if and why a user receives a notification.</p>
BEHOXQ ERASE NOTIFICATIONS	action	<p>Erase all notifications for a User, erase all notifications for a Patient, or erase a specific Notification.</p>
BEHOXQ FLAG ORDERABLE ITEMS	action	<p>Two sub-options to flag specific orderable items to send notifications when they are ordered or resulted. [Note: This option can not be available until late alpha testing.]</p>
BEHOXQ FLAGGED ORDERS BULLETIN	action	<p>Enter 'yes' to send a bulletin when an order is flagged for clarification. 'Yes' indicates a MailMan bulletin will be sent to the order's Current Provider (usually the Ordering Provider) when the order is flagged for clarification. This parameter has no effect on the Flagged Orders notification which is also triggered when an order is flagged for clarification.</p>
BEHOXQ FORWARD BACKUP REVIEWER	action	<p>The number of days to hold a notification before it is forwarded to a recipient's backup reviewer. The maximum is 30 days. If not indicated or zero, the notification will not be forwarded. For example, if a notification has a value of 14 for this parameter, it will be forwarded to the backup reviewer of each recipient who hasn't processed the notification after 14 days. Determination of recipients who have not processed the notification and who their supervisors are is made by the Kernel Alert Utility. It will not be forwarded to the backup reviewer of recipients who have processed the alert within 14 days. If the value of this parameter is zero or non-existent, the alert/notification will never be forwarded to the backup reviewer.</p>
BEHOXQ FORWARD NOTIFS MENU	menu	<p>This menu option is used to access options which set up parameters for forwarding notifications/alerts.</p>



Option	Type	Description
BEHOXQ FORWARD SUPERVISOR	action	The number of days to hold a notification before it is forwarded to a recipient's supervisor. The maximum is 30 days. If not indicated or zero, the notification will not be forwarded. For example, if a notification has a value of 14 for this parameter, it will be forwarded to the supervisor of each recipient who hasn't processed the notification after 14 days. Determination of recipients who have not processed the notification and who their supervisors are is made by the Kernel Alert Utility. It will not be forwarded to supervisors of recipients who have processed the alert within 14 days. If the value of this parameter is zero or non-existent, the alert/notification will never be forwarded. [Note: This option cannot be available until late alpha testing.]
BEHOXQ FORWARD SURROGATES	action	Number of days to hold notification before forwarding to recipient's surrogates. The maximum is 30 days. If not indicated or zero, the notification will not be forwarded. For example, if a notification has a value of 14 for this parameter, it will be forwarded to the supervisor of each recipient who hasn't processed the notification after 14 days. Determination of recipients who have not processed the notification and who their supervisors are is made by the Kernel Alert Utility. It will not be forwarded to supervisors of recipients who have processed the alert within 14 days. If the value of this parameter is zero or non-existent, the alert/notification will never be forwarded. [Note: This option cannot be available until late alpha testing.]
BEHOXQ MAIN	menu	Notification configuration menu.
BEHOXQ PARAMETERS	menu	Notification parameters menu.
BEHOXQ PROCESSING FLAG	action	<p>Set of codes indicating processing flag for the entity and notification.</p> <p>E (Enabled): Notification enabled for entity unless entity of higher precedence has notification disabled, e.g., Enabled at System level and Disabled at User level - User will not receive notification.</p> <p>D (Disabled): Notification disabled for entity unless entity of higher precedence has notification enabled, e.g., Disabled at System level and Enabled at User level - User will receive notification.</p> <p>Note: If the option Set Default Recipients for Notifications (parameter ORB3 REGULAR RECIPIENTS) is set to 'True' for a Team or User, that Team or User will always receive the notification, regardless of the Enabled/Disabled settings.</p> <p>Note: During alpha, Class and Service will not affect processing.</p>

Option	Type	Description
BEHOXQ PROVIDER RECIPIENTS	action	<p>Any one or combination of 'P', 'A', 'T', and/or 'O'. Set of codes indicating default (provider) recipients of a notification by their title or relationship to the patient. Notifications can be set up with any or all of the following codes:</p> <p>P (Primary Provider): Deliver notification to the patient's Primary Provider.  A (Attending Physician): Deliver notification to the patient's Attending Physician.  T (Patient Care Team): Deliver notification to the patient's primary care Team.  O (Ordering Provider): Deliver notification to the provider who placed the order which triggered the notification.</p> <p>The providers, physicians and teams must be set up properly and accurately for the correct individuals to receive the notification.</p>
	action	This option prompts for a user/recipient then processes each notification to determine if and why the user will receive the notification. This information is then displayed.
BEHOXQ SHOW ALL	action	Set show all notifications default.
BEHOXQ SORT BY	action	Notification sort column.
BEHOXQ SYSTEM ENABLE/DISABLE	action	Determines if any notification processing will occur in entire Notifications system. 'E' or 'Enable' indicates the Notification system is enabled and running. 'D' or 'Disabled' indicates the Notification system is disabled and not running. Can be set at the Institution or System level.
BEHOXQ UNVERIFIED MED ORDER	action	Enter the number of hours delay to wait after a medication order is placed before triggering an Unverified Medication Order notification/alert. The maximum number of hours is 10,000.
BEHOXQ UNVERIFIED ORDER	action	Enter the number of hours delay to wait after an order is placed before triggering an Unverified Order notification/alert. The maximum number of hours is 10,000.
BEHOXQ UNVERIFIED ORDERS MENU	menu	This menu option provides access to options which set up parameters for delaying unverified orders alert triggers.
BEHOXQ URGENCY	action	<p>Set of codes indicating the urgency for a notification for a site. The urgency is mainly used for sorting in displays. The codes include:</p> <p>1 (High): notification is Highly urgent  2 (Moderate): notification is Moderately urgent  3 (Low): notification is of Low urgency</p>
BEHOXQPC REQUIRES E&M CODE	action	Providers that Require an E&M Code

## 50.7 Exported Security Keys

None.

## 50.8 Exported Protocols

None.

## 50.9 Exported Parameters

Parameter	Instance Type	Value Type	Precedence	Description
BEHOXQ ALERT THRESHOLD		Set	User, Division, System	This is the alert priority threshold at or above which a popup alert will be displayed when an alert of that or greater priority is received. Possible values are:  0 = None 1 = High 2 = Medium 3 = Low
BEHOXQ SHOW ALL		Boolean	User, Division, System	If set to YES, the default setting for displaying notifications in the graphical interface is "all patient." If set to NO, the default setting is "selected patient only."
BEHOXQ SORT BY		Set	User, Division, System	Method for sorting notifications when displayed. Methods include: by Patient, by Type (Notification name), and by Urgency. Within these sort methods notifications are presented in reverse chronological order.
BEHOXQPC REQUIRES E&M CODE		Boolean	User, Class	Indicates whether or not a primary provider requires an E&M code for billing purposes.

## 50.10 Exported Mail Groups

None.

## 50.11 Callable Routines

This section describes supported entry points for routines exported with this component.

### 50.11.1 RPC: BEHOXQ ALRLIST

Scope: private.

Parameter	Datatype	Description
DFN (optional)	Pointer (#2)	Patient IEN. If not specified, retrieves for all patient.
ST (optional)	FM Date/Time	Option starting date/time for retrieval.
<return value>	String List	List of records in the format: Priority^Info Only^Patient Name^Display Text^Date Delivered^Sender Name^DFN^Alert Type^Alert ID^Can Delete^Extra Info

Returns a list of notifications matching the specified criteria.

### 50.11.2 RPC: BEHOXQ ALRMSG

Scope: private.

Parameter	Datatype	Description
AID	String	Alert id
<return value>	String List	Message text associated with the alert.

Retrieves comment and message text associated with an alert.

### 50.11.3 RPC: BEHOXQ ALRPP

Scope: private.

Parameter	Datatype	Description
AID	String	Alert id
<return value>	Pointer (#90460.021)	IEN of control file entry.

Performs post processing logic for an alert.

### 50.11.4 RPC: BEHOXQ FORWARD

Scope: private.

Parameter	Datatype	Description
AID	Array	List of alert id's to forward.
USR	Array	List of recipients
CMT (optional)	String	Comment text.
<return value>	Integer	Always 0.

Forwards an alert.

### 50.11.5 RPC: BEHOXQ SCHALR

Scope: private.

Parameter	Datatype	Description
DAT	FM Date/Time	Date and time of scheduled delivery.
ID	String	Notification ID
SBJ	String	Notification subject.
XTR	String	Additional control data.
MSG	String List	Message text.
RCP	Array	List of recipient IENs (if negative, assumes mail group IEN, otherwise user IEN).
<return value>	Boolean	True if successful.

Schedules an alert for later delivery.

### 50.11.6 RPC: BEHOXQ SCHDEL

Scope: private.

Parameter	Datatype	Description
IEN	Pointer (#90460.022)	IEN of entry to delete.
<return value>	Boolean	True if successful.

Deletes a scheduled alert.

### 50.11.7 RPC: BEHOXQ SCHLIST

Scope: private.

Parameter	Datatype	Description
ID	String	Alert ID
USR (optional)	Pointer (#200)	If not specified, assumes current user.
<return value>	String List	List of records in the format: IEN^Date^Patient Name^Subject^Data

Lists alerts scheduled by the specified user.

### 50.11.8 RPC: BEHOXQ SCHMSG

Scope: private.

Parameter	Datatype	Description
IEN	Pointer (#90460.022)	IEN of scheduled alert.
<return value>	String List	Message text.

Returns message text associated with a scheduled alert.

### 50.11.9 RPC: BEHOXQ SCHRECIP

Scope: private.

Parameter	Datatype	Description
IEN	Pointer (#90460.022)	IEN of scheduled alert.
<return value>	String List	List of recipients

Returns a list of recipients for a scheduled alert.

### 50.11.10 RPC: BEHOXQPC NOEMC

Scope: private.

Parameter	Datatype	Description
VSIT	Pointer (#9000010)	Visit IEN
PRI (optional)	Integer	Priority (defaults to 3)
<return value>	Boolean	Returns true if alert not needed.

Generates a missing E&M code alert.

### 50.11.11 RPC: BEHOXQPC NOPOV

Scope: private.

Parameter	Datatype	Description
VSIT	Pointer (#9000010)	Visit IEN
PRI (optional)	Integer	Priority (defaults to 3)
<return value>	Boolean	Returns true if alert not needed.

Generates a missing POV alert.

## 50.12 External Relations

Entity	Name	Description
Package	KERNEL 8.0	Uses supported alert APIs.

## 50.13 Internal Relations

None.

## 50.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 50.15 Components

This component supports the following properties and methods:

### 50.15.1 Properties

Property	Datatype	Access	Description
ALERTTHRESHOLD	Enum	RW	Sets the priority threshold at or above which a popup alert is issued when a notification is received. One of: 0 = Never 1 = High 2 = Medium 3 = Low
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom

Property	Datatype	Access	Description
AUTOSIZE	Boolean	RW	If true, the component automatically resizes itself to accommodate its contents.
BORDERSTYLE	Enum	RW	Sets the style of the border surrounding the component. Can be one of: 0 = None 1 = Single 2 = Sunken 3 = Raised
CAPTION	String	RW	Sets the text displayed in the title bar. To justify portions of the caption text, use the “\” character to delimit the left-, center-, and right-justified portions of the caption text.
CAPTIONCOLOR1 CAPTIONCOLOR2	Color	RW	Colors to apply to the title bar. If the two colors differ and a gradient style is set, a gradient effect is created. For a standard title bar style, only the first color is applied.
CAPTIONSTYLE	Enum	RW	Sets the caption style. Can be one of: 0 = None – No caption (hides title bar) 1 = Title – Standard title bar 2 = Frame – Framed title bar (group box style) 3 = Left – Left gradient title bar 4 = Right – Right gradient title bar 5 = Center – Center gradient title bar
COLOR	Color	RW	Sets the background color of the component.
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPPFILE	String	RW	Sets the name of the help file associated with the component.
LAYOUT	String	RW	Column layout information.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
LEGEND	Enum	RW	Controls the position of the legend. One of: 0 = Position on left 1 = Position on right 2 = Hide legend
SHOWALL	Boolean	RW	If true, notifications for all patients are shown. If false, only those for the current patient are shown.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

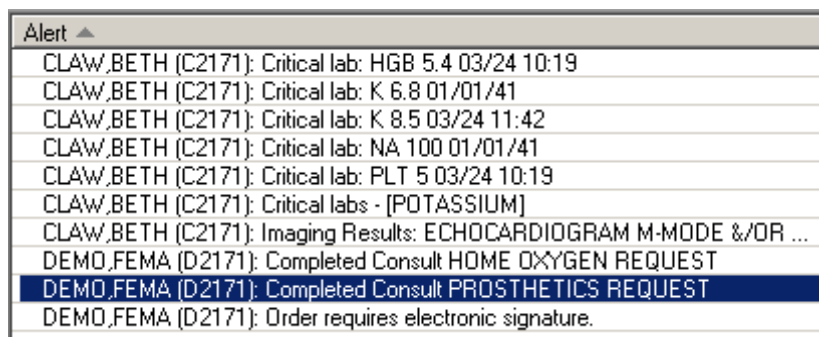
## 50.15.2 Schedule

Scope: private.

Invokes the notification scheduling dialog.

## 51.0 Alerts

### 51.1 Introduction



Alert
CLAW, BETH (C2171): Critical lab: HGB 5.4 03/24 10:19
CLAW, BETH (C2171): Critical lab: K 6.8 01/01/41
CLAW, BETH (C2171): Critical lab: K 8.5 03/24 11:42
CLAW, BETH (C2171): Critical lab: NA 100 01/01/41
CLAW, BETH (C2171): Critical lab: PLT 5 03/24 10:19
CLAW, BETH (C2171): Critical labs - [POTASSIUM]
CLAW, BETH (C2171): Imaging Results: ECHOCARDIOGRAM M-MODE &/OR ...
DEMO, FEMA (D2171): Completed Consult HOME OXYGEN REQUEST
DEMO, FEMA (D2171): Completed Consult PROSTHETICS REQUEST
DEMO, FEMA (D2171): Order requires electronic signature.

Figure 51-1: Sample Alerts

The Alerts component displays selected alerts for display on the cover sheet. This component is provided for backward compatibility and has been largely replaced by the Notifications component.

### 51.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHALERTS.ALERTS
Version	4.2.0.10
Class Identifier	{09C3EEF7-591F-4529-96D3-7E7B90B0D4F5}
Image File	BEHAlerts.ocx
Property Initializations	none
Serializable Properties	ALLOWPRINT=BOOL, AUTOSIZE=BOOL, BORDERSTYLE=ENUM, CAPTION=TEXT, CAPTIONCOLOR1=COLOR, CAPTIONCOLOR2=COLOR, CAPTIONSTYLE=ENUM, COLOR=COLOR, DEFERUPDATE=BOOL, DETAILPANE=BOOL, ORIENTATION=ENUM, LAYOUT=HIDDEN
Required Files	
Security Keys	none
Multiple Instances Allowed	yes
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BEHO*1.1*026001

There are no specific implementation or maintenance tasks associated with this component.



## 51.3 Routine Descriptions

This component has been assigned the namespace designation of “BEHOXQCV.” The following routines are distributed:

Routine	Description
BEHOXQCV	Alert cover sheet support.

## 51.4 File List

None.

## 51.5 Cross References

None.

## 51.6 Exported Options

None.

## 51.7 Exported Security Keys

None.

## 51.8 Exported Protocols

None.

## 51.9 Exported Parameters

None.

## 51.10 Exported Mail Groups

None.

## 51.11 Callable Routines

This section describes supported entry points for routines exported with this component.

### 51.11.1 RPC: BEHOXQCV DETAIL

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient IEN
AID	String	Alert ID
<return value>	String List	Detail text of alert.

This RPC has not been implemented.

## 51.11.2 RPC: BEHOXQCV LIST

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient IEN
FLG	String	Controls what alerts are returned. One of: A = one patient, all users (default) U = one patient, one user P = all patients, one user
<return value>	String List	List of alerts matching the specified criteria.

Returns a list of alerts matching the specified criteria.

## 51.12 External Relations

Entity	Name	Description
Package	KERNEL 8.0	Calls supported alert APIs.

## 51.13 Internal Relations

None.

## 51.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 51.15 Components

This component supports the following properties and methods:

### 51.15.1 Properties

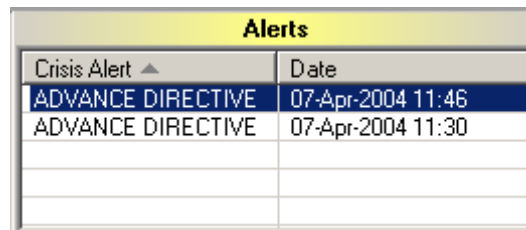
Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ALLOWPRINT	Boolean	RW	If true, a print button will appear on the detail dialog allowing printing of the contents.

Property	Datatype	Access	Description
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
AUTOSIZE	Boolean	RW	If true, the component automatically resizes itself to accommodate its contents.
BORDERSTYLE	Enum	RW	Sets the style of the border surrounding the component. Can be one of: 0 = None 1 = Single 2 = Sunken 3 = Raised
CAPTION	String	RW	Sets the text displayed in the title bar. To justify portions of the caption text, use the “\” character to delimit the left-, center-, and right-justified portions of the caption text.
CAPTIONCOLOR1 CAPTIONCOLOR2	Color	RW	Colors to apply to the title bar. If the two colors differ and a gradient style is set, a gradient effect is created. For a standard title bar style, only the first color is applied.
CAPTIONSTYLE	Enum	RW	Sets the caption style. Can be one of: 0 = None – No caption (hides title bar) 1 = Title – Standard title bar 2 = Frame – Framed title bar (group box style) 3 = Left – Left gradient title bar 4 = Right – Right gradient title bar 5 = Center – Center gradient title bar
COLOR	Color	RW	Sets the background color of the component.
DEFERUPDATE	Boolean	RW	If true, data refresh is deferred until the component becomes visible. If false, data refresh happens immediately.
DETAILPANE	Boolean	RW	If true, a detail pane appears next to the list view that displays detail text of the selected entry. If false, detail text appears in a popup dialog when an entry is clicked.
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPPFILE	String	RW	Sets the name of the help file associated with the component.
LAYOUT	String	RW	Property representing the internal layout of the form.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
ORIENTATION	Enum	RW	Sets the orientation of the detail pane. Can be one of: 0 = Horizontal 1 = Vertical
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.

<b>Property</b>	<b>Datatype</b>	<b>Access</b>	<b>Description</b>
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 52.0 Crisis Alerts

### 52.1 Introduction



Alerts	
Crisis Alert ▲	Date
ADVANCE DIRECTIVE	07-Apr-2004 11:46
ADVANCE DIRECTIVE	07-Apr-2004 11:30

Figure 52-1: Sample Crisis Alerts

The Crisis Alerts component displays crisis-related information.

### 52.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHCRISES.CRISES
Version	4.2.0.9
Class Identifier	{9D8BA58A-66EB-4246-9FF2-F6C252D0F467}
Image File	BEHCrises.ocx
Property Initializations	none
Serializable Properties	ALLOWPRINT=BOOL, AUTOSIZE=BOOL, BORDERSTYLE=ENUM, CAPTION=TEXT, CAPTIONCOLOR1=COLOR, CAPTIONCOLOR2=COLOR, CAPTIONSTYLE=ENUM, COLOR=COLOR, DEFERUPDATE=BOOL, DETAILPANE=BOOL, ORIENTATION=ENUM, LAYOUT=HIDDEN
Required Files	BEHCrises.chm
Security Keys	none
Multiple Instances Allowed	yes
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BEHO*1.1*029001

There are no specific implementation or maintenance tasks associated with this component.

### 52.3 Routine Descriptions

This component has been assigned the namespace designation of “BEHOCA.” The following routines are distributed:

<b>Routine</b>	<b>Description</b>
BEHOCACV	Crisis alerts cover sheet support.

## 52.4 File List

None.

## 52.5 Cross References

None.

## 52.6 Exported Options

None.

## 52.7 Exported Security Keys

None.

## 52.8 Exported Protocols

None.

## 52.9 Exported Parameters

None.

## 52.10 Exported Mail Groups

None.

## 52.11 Callable Routines

This section describes supported entry points for routines exported with this component.

### 52.11.1 RPC: BEHOCACV CWAD

Scope: private.

<b>Parameter</b>	<b>Datatype</b>	<b>Description</b>
DFN	Pointer (#2)	Patient IEN
<return value>	String	Returns string containing one or more of: C, W, A, D.

Returns string containing one or more of: C, W, A, D.

## 52.11.2 RPC: BEHOCACV DETAIL

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient IEN
<return value>	String List	Detail report for allergies/adverse reactions.

Produces a detail report of a patient's allergies/adverse reactions.

## 52.11.3 RPC: BEHOCACV LIST

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient IEN
<return value>	String List	List of crisis alerts for populating list.

Returns a list of crisis alerts for a patient.

## 52.12 External Relations

Entity	Name	Description
Package	TIU 1.0	Uses following API calls: ENCOVER^TIUUP3
Package	ADVERSE REACTION TRACKING 4.0	Uses following API calls: EN1^GMRAOR1

## 52.13 Internal Relations

None.

## 52.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 52.15 Components

This component supports the following properties and methods:

### 52.15.1 Properties

The properties are described in the following table.

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ALLOWPRINT	Boolean	RW	If true, a print button will appear on the detail dialog allowing printing of the contents.
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
AUTOSIZE	Boolean	RW	If true, the component automatically resizes itself to accommodate its contents.
BORDERSTYLE	Enum	RW	Sets the style of the border surrounding the component. Can be one of: 0 = None 1 = Single 2 = Sunken 3 = Raised
CAPTION	String	RW	Sets the text displayed in the title bar. To justify portions of the caption text, use the “\” character to delimit the left-, center-, and right-justified portions of the caption text.
CAPTIONCOLOR1 CAPTIONCOLOR2	Color	RW	Colors to apply to the title bar. If the two colors differ and a gradient style is set, a gradient effect is created. For a standard title bar style, only the first color is applied.
CAPTIONSTYLE	Enum	RW	Sets the caption style. Can be one of: 0 = None – No caption (hides title bar) 1 = Title – Standard title bar 2 = Frame – Framed title bar (group box style) 3 = Left – Left gradient title bar 4 = Right – Right gradient title bar 5 = Center – Center gradient title bar
COLOR	Color	RW	Sets the background color of the component.
DEFERUPDATE	Boolean	RW	If true, data refresh is deferred until the component becomes visible. If false, data refresh happens immediately.
DETAILPANE	Boolean	RW	If true, a detail pane appears next to the list view that displays detail text of the selected entry. If false, detail text appears in a popup dialog when an entry is clicked.
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.



<b>Property</b>	<b>Datatype</b>	<b>Access</b>	<b>Description</b>
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPPFILE	String	RW	Sets the name of the help file associated with the component.
LAYOUT	String	RW	Property representing the internal layout of the form.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
ORIENTATION	Enum	RW	Sets the orientation of the detail pane. Can be one of: 0 = Horizontal 1 = Vertical
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 53.0 Crises/Warnings/Alerts/Directives (CWAD)

### 53.1 Introduction



Figure 53-1: Sample Postings Button

The CWAD component provides a compact display of critical status flags.

### 53.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHCWAD.CWAD
Version	4.2.0.11
Class Identifier	{4BF06C7D-9E61-4F2E-B910-4793BDB0BD3C}
Image File	BEHCWAD.ocx
Property Initializations	none
Serializable Properties	COLOR=COLOR
Required Files	none
Security Keys	none
Multiple Instances Allowed	yes
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BEHO*1.1*030001

There are no specific implementation or maintenance tasks associated with this component.

### 53.3 Routine Descriptions

None.

### 53.4 File List

None.

### 53.5 Cross References

None.

### 53.6 Exported Options

None.

## 53.7 Exported Security Keys

None.

## 53.8 Exported Protocols

None.

## 53.9 Exported Parameters

None.

## 53.10 Exported Mail Groups

None.

## 53.11 Callable Routines

None.

## 53.12 External Relations

Entity	Name	Description
Package	TIU 1.0	Calls supported APIs.

## 53.13 Internal Relations

Entity	Name	Description
Component	Crisis Alerts	Calls supported APIs.

## 53.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 53.15 Components

This component supports the following properties and methods:

### 53.15.1 Properties

The properties are described in the following table.

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
AUTOSIZE	Boolean	RW	If true, the component automatically resizes itself to accommodate its contents.
BORDERSTYLE	Enum	RW	Sets the style of the border surrounding the component. Can be one of: 0 = None 1 = Single 2 = Sunken 3 = Raised
COLOR	Color	RW	Sets the background color of the component.
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPPFILE	String	RW	Sets the name of the help file associated with the component.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
MINHEIGHT	Integer	RW	Sets the minimum height, in pixels, that the component can attain.
MINWIDTH	Integer	RW	Sets the minimum width, in pixels, that the component can attain.
THEMEAWARE	Boolean	RW	If true, the component is rendered as a themed button.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 54.0 Reminders (PCC)

### 54.1 Introduction

Reminders	
Reminder ▲	Date
HepA Adult Immunization	DUE NOW

Figure 54-1: Sample Reminders

Provides an overview of active reminders for display on the cover sheet.

### 54.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHREMINDERS.REMINDERS
Version	4.2.0.4
Class Identifier	{F50C0969-8453-407D-AE42-F7326FC7833C}
Image File	BEHReminders.ocx
Property Initializations	none
Serializable Properties	ALLOWPRINT=BOOL, CAPTION=TEXT, CAPTIONCOLOR1=COLOR, CAPTIONCOLOR2=COLOR, CAPTIONSTYLE=ENUM, DEFERUPDATE=BOOL, DETAILPANE=BOOL, ORIENTATION=ENUM, LAYOUT=HIDDEN
Required Files	BEHReminders.chm
Security Keys	none
Multiple Instances Allowed	yes
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BEHO*1.1*041001

There are no specific implementation or maintenance tasks associated with this component.

### 54.3 Routine Descriptions

This component has been assigned the namespace designation of “BEHORMCV,” The following routines are distributed:

Routine	Description
BEHORMCV	Reminder cover sheet support.

## 54.4 File List

None.

## 54.5 Cross References

None.

## 54.6 Exported Options

None.

## 54.7 Exported Security Keys

None.

## 54.8 Exported Protocols

None.

## 54.9 Exported Parameters

None

## 54.10 Exported Mail Groups

None

## 54.11 Callable Routines

This section describes supported entry points for routines exported with this component.

### 54.11.1 BEHORMCV DETAIL

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient IEN
IEN	Pointer (#811.9)	Reminder IEN
<return value>	String List	Detailed information about the specified reminder.

Returns detailed information about the specified reminder.

### 54.11.2 BEHORMCV LIST

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient IEN
LOC	Pointer (#44)	Hospital Location IEN
SRV	Pointer (#49)	Service/Section IEN
<return value>	String List	List of records in the format: ien (811.9)^reminder print name^date due^last occur

Returns a list of currently due reminders matching specified criteria.

## 54.12 External Relations

Entity	Name	Description
Package	CLINICAL REMINDERS 1.5	Uses following API calls: MAIN^PXRM CATREM^PXRM^PIO INACTIVE^PXRM

## 54.13 Internal Relations

None.

## 54.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 54.15 Components

This component supports the following properties and methods:

### 54.15.1 Properties

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ALLOWPRINT	Boolean	RW	If true, a print button will appear on the detail dialog allowing printing of the contents.
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom

Property	Datatype	Access	Description
AUTOSIZE	Boolean	RW	If true, the component automatically resizes itself to accommodate its contents.
BORDERSTYLE	Enum	RW	Sets the style of the border surrounding the component. Can be one of: 0 = None 1 = Single 2 = Sunken 3 = Raised
CAPTION	String	RW	Sets the text displayed in the title bar. To justify portions of the caption text, use the “\” character to delimit the left-, center-, and right-justified portions of the caption text.
CAPTIONCOLOR1 CAPTIONCOLOR2	Color	RW	Colors to apply to the title bar. If the two colors differ and a gradient style is set, a gradient effect is created. For a standard title bar style, only the first color is applied.
CAPTIONSTYLE	Enum	RW	Sets the caption style. Can be one of: 0 = None – No caption (hides title bar) 1 = Title – Standard title bar 2 = Frame – Framed title bar (group box style) 3 = Left – Left gradient title bar 4 = Right – Right gradient title bar 5 = Center – Center gradient title bar
COLOR	Color	RW	Sets the background color of the component.
DEFERUPDATE	Boolean	RW	If true, data refresh is deferred until the component becomes visible. If false, data refresh happens immediately.
DETAILPANE	Boolean	RW	If true, a detail pane appears next to the list view that displays detail text of the selected entry. If false, detail text appears in a popup dialog when an entry is clicked.
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPPFILE	String	RW	Sets the name of the help file associated with the component.
LAYOUT	String	RW	Property representing the internal layout of the form.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
ORIENTATION	Enum	RW	Sets the orientation of the detail pane. Can be one of: 0 = Horizontal 1 = Vertical
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.



## 55.0 View Reminders (CPRS)

### 55.1 Introduction



Figure 55-1: View Reminders Button

The View Reminders component indicates if there are reminders due and provides a means to view them.

### 55.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHREMINDESVIEW.REMINDESVIEW
Version	4.2.0.23
Class Identifier	{8AFF85FD-6EA4-4653-AF48-B15C0911D81F}
Image File	BEHRemindersView.ocx
Property Initializations	none
Serializable Properties	CAPTION=TEXT, COLOR=COLOR
Required Files	none
Security Keys	none
Multiple Instances Allowed	no
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BEHO*1.1*024001

There are no specific implementation or maintenance tasks associated with this component.

### 55.3 Routine Descriptions

None.

### 55.4 File List

None.

### 55.5 Cross References

None.

## 55.6 Exported Options

Option	Type	Description
BEHORM (IN)/ACTIVATE REMINDERS	edit	This option is used to make reminders active or inactive.
BEHORM CATEGORY EDIT/ INQUIRE	action	Add/edit reminder categories.
BEHORM CF MANAGEMENT	menu	Reminder computed finding management menu.
BEHORM COMPUTED FINDING EDIT	action	Reminder computed finding edit.
BEHORM COMPUTED FINDING LIST	print	This option lists the computed findings that are defined at a site.
BEHORM DEFAULT LOCATION	action	Default outside location.
BEHORM DEFINITION COPY	action	Copy reminder definition.
BEHORM DEFINITION EDIT	action	This option is used to edit the clinical reminder definitions.
BEHORM DEFINITION INQUIRY	action	Inquire about a reminder definition.
BEHORM DEFINITION LIST	action	List reminder definitions.
BEHORM DIALOG MANAGEMENT	menu	Reminder dialog management menu.
BEHORM DIALOG PARAMETERS	menu	Dialog parameters menu.
BEHORM DIALOG/COMPONENT EDIT	action	Reminder dialog edit.
BEHORM EHR CONFIGURATION	action	Allow EHR configuration in GUI.
BEHORM EHR COVER SHEET LIST	action	EHR cover sheet reminder list.
BEHORM EHR LOOKUP CATEGORIES	action	EHR lookup categories.

Option	Type	Description
BEHORM EXTRACT EPI FINDING LIS	print	This option allows printing extract results stored in the REMINDER EXTRACT SUMMARY file (810.3). This option lists extracted data by finding item and social security number. The finding items are loaded into the REMINDER EXTRACT SUMMARY file when one of the following options is run: LREPI ENHANCE MANUAL RUN, Lab Search/Extract Manual Run (Enhanced), LREPI NIGHTLY TASK, Lab Search/Extract Nightly Task The Lab EPI (Emerging Pathogens Initiative) extract name follows the convention: LREPI YY/MM MMDDYY, where the YY/MM is the year and month from the date range specified by the LREPI extract option that created the extract. MMDDYY is the month, day and year that the extract was run. The prefix of LREPI identifies the extract as the result of LREPI extract options (manual or automatic run for the LREPI Lab Search/Extract Protocol). To print all extracts for the target reporting month, enter "LREPI YY/MM" at the FROM prompt, with the year and month's data to be reported. At the TO prompt, you can limit the print to extracts for the reporting month by adding a suffix to the LREPI YY/MM, for example "LREPI YY/MM z" will get all extracts that were created for the target reporting month, regardless of when they were run. If there have been multiple runs for the target month, you can further define the FROM and TO range by specifying LREPI YY/MM MMDDYY with the MMDDYY being the date the report was run. There will normally only be one extract per month. More than one will exist if your site runs the LREPI ENHANCE MANUAL RUN to correct errors found by the Austin Automation Center on the monthly transmission of EPI data sent by the LREPI NIGHTLY TASK.
BEHORM EXTRACT EPI TOTALS	print	This option is used to summarize total counts for each type of finding item that was extracted for the target date range of the LREPI ENHANCE MANUAL RUN or the LREPI NIGHTLY TASK option run.
BEHORM FINDING ITEM PARAMETERS	action	Finding item parameters.
BEHORM FINDING TYPE PARAMETERS	action	General finding type parameters.
BEHORM GUI REMINDERS ACTIVE	action	Reminder GUI resolution active.
BEHORM HEALTH FACTOR RESOLUTIO	action	Health factor resolutions.

<b>Option</b>	<b>Type</b>	<b>Description</b>
BEHORM INFO ONLY	menu	Reminder information only menu
BEHORM MAIN	menu	Reminder configuration menu.
BEHORM NEW REMINDER PARAMETERS	action	New reminder parameters.
BEHORM PARAMETERS	menu	Reminder parameters menu.
BEHORM PROGRESS NOTE HEADERS	action	Progress note headers.
BEHORM REMINDER EXCHANGE	action	Reminder exchange.
BEHORM REMINDER MANAGEMENT	menu	Reminder definition management.
BEHORM REMINDER REPORTS	menu	Reminder reports.
BEHORM REMINDER TEST	action	Reminder test.
BEHORM REMINDERS DUE	action	Reminders due report.
BEHORM REMINDERS DUE (USER)	action	Reminders due report (user).
BEHORM REPORT TEMPLATE (USER)	action	User report templates.
BEHORM RESOLUTION EDIT/ INQUIRE	action	Reminder resolution statuses.
BEHORM REVIEW DATES	action	Review date report.
BEHORM SPONSOR EDIT	action	Edit reminder sponsor.
BEHORM SPONSOR INQUIRY	action	Reminder sponsor inquiry.
BEHORM SPONSOR LIST	action	List reminder sponsors.
BEHORM SPONSOR MANAGEMENT	menu	Reminder sponsor management menu.
BEHORM TAXONOMY COPY	action	Copy taxonomy item.
BEHORM TAXONOMY DIALOG	action	Taxonomy dialog parameters.
BEHORM TAXONOMY EDIT	action	Edit taxonomy item.
BEHORM TAXONOMY INQUIRY	action	Inquire about taxonomy item.

Option	Type	Description
BEHORM TAXONOMY LIST	action	This option lists the current definitions of all the taxonomies defined in the REMINDER TAXONOMY file. The REMINDER TAXONOMY file is used to define a range of coded values from ICD Diagnosis codes, ICD Operation / Procedures codes, and CPT codes that can be viewed as being part of a clinical category (taxonomy). Each entry has a low value and a high value. The software will search for matches on all the codes between the low and high values inclusive. If there is a match then the taxonomy finding will be true for the patient.
BEHORM TAXONOMY MANAGEMENT	menu	Reminder taxonomy management menu.
BEHORM TERM COPY	action	Copy reminder term.
BEHORM TERM EDIT	action	Reminder term edit.
BEHORM TERM INQUIRY	action	Inquire about reminder term.
BEHORM TERM LIST	action	This option is used to give a brief listing of reminder terms.
BEHORM TERM MANAGEMENT	menu	Reminder term management menu.
BEHORM TEXT AT CURSOR	action	Position reminder text at cursor.

## 55.7 Exported Security Keys

None.

## 55.8 Exported Protocols

None.

## 55.9 Exported Parameters

None.

## 55.10 Exported Mail Groups

None.

## 55.11 Callable Routines

None.

## 55.12 External Relations

Entity	Name	Description
Package	CLINICAL REMINDERS 1.5	Uses supported APIs

## 55.13 Internal Relations

None.

## 55.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 55.15 Components

This component supports the following properties and methods:

### 55.15.1 Properties

The properties are described in the following table.

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
AUTOSIZE	Boolean	RW	If true, the component automatically resizes itself to accommodate its contents.
BORDERSTYLE	Enum	RW	Sets the style of the border surrounding the component. Can be one of: 0 = None 1 = Single 2 = Sunken 3 = Raised
COLOR	Color	RW	Sets the background color of the component.
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.

<b>Property</b>	<b>Datatype</b>	<b>Access</b>	<b>Description</b>
HELPPFILE	String	RW	Sets the name of the help file associated with the component.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
MINHEIGHT	Integer	RW	Sets the minimum height, in pixels, that the component can attain.
MINWIDTH	Integer	RW	Sets the minimum width, in pixels, that the component can attain.
THEMEAWARE	Boolean	RW	If true, the component is rendered as a themed button.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 56.0 Integrated Signature Tool

### 56.1 Introduction

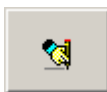


Figure 56-1: Integrated Signature Tool Button

The Integrated Signature Tool component provides ready access to the electronic signature service.

### 56.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHESIGREVIEW.REVIEW
Version	20.1.0.13
Class Identifier	{BE3EA83C-0D4A-428E-80E3-5D63ED3B0915}
Image File	BEHESigReview.ocx
Property Initializations	
Serializable Properties	CAPTION=TEXT, COLOR=COLOR
Required Files	
Security Keys	none
Multiple Instances Allowed	no
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BEHO*1.1*016001

There are no specific implementation or maintenance tasks associated with this component.

### 56.3 Routine Descriptions

None.

### 56.4 File List

None.

### 56.5 Cross References

None.



## 56.6 Exported Options

None.

## 56.7 Exported Security Keys

None.

## 56.8 Exported Protocols

None.

## 56.9 Exported Parameters

None.

## 56.10 Exported Mail Groups

None.

## 56.11 Callable Routines

None.

## 56.12 External Relations

None.

## 56.13 Internal Relations

Entity	Name	Description
Component	Electronic Signature Service	Uses supported APIs.

## 56.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 56.15 Components

This component supports the following properties and methods:

### 56.15.1 Properties

The properties are described in the following table.

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
AUTOSIZE	Boolean	RW	If true, the component automatically resizes itself to accommodate its contents.
BORDERSTYLE	Enum	RW	Sets the style of the border surrounding the component. Can be one of: 0 = None 1 = Single 2 = Sunken 3 = Raised
CAPTION	String	RW	Sets the text displayed in the button caption.
COLOR	Color	RW	Sets the background color of the component.
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
GLYPH	String	RW	Name of bitmap file containing glyphs to be displayed on button surface.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPPFILE	String	RW	Sets the name of the help file associated with the component.
LAYOUT	String	RW	Property representing the internal layout of the form.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
NUMGLYPHS	Integer	RW	Number of glyphs contained in glyph file.
TITLE	String	RW	Title for the report.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 57.0 Electronic Signature Service

### 57.1 Introduction

The Electronic Signature Service provides unified electronic signature capabilities across the application.

### 57.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHESIGSERVICE.ESIGSERVICE
Version	20.1.0.22
Class Identifier	{295FF753-C73B-449F-97F3-6A9EF2972E84}
Image File	BEHESigService.dll
Property Initializations	none
Serializable Properties	none
Required Files	none
Security Keys	none
Multiple Instances Allowed	no
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	yes
Side-by-Side Versioning	yes
Service	yes
.Net Component	no
Associated Build	BEHO*1.1*016001

There are no specific implementation or maintenance tasks associated with this component.

### 57.3 Routine Descriptions

None.

### 57.4 File List

None.

#### 57.4.1 Cross References

None.

#### 57.4.2 Exported Options

None.

### 57.4.3 Exported Security Keys

None.

### 57.4.4 Exported Protocols

None.

### 57.4.5 Exported Parameters

None.

## 57.5 Exported Mail Groups

None.

## 57.6 Callable Routines

None.

## 57.7 External Relations

Entity	Name	Description
Package	ORDER ENTRY/RESULTS REPORTING 3.0	Uses supported APIs.

## 57.8 Internal Relations

None.

## 57.9 Archiving and Purging

There are no archiving or purging requirements within this software.

## 57.10 Components

This component supports the following properties and methods:

### 57.10.1 Properties

Property	Datatype	Access	Description
CanSign	Boolean	R	True if any items in the signature list can be signed by the user.
Count	Integer	R	Count of items in the signature list.
RequireReview	Boolean	R	True if any items in signature list require manual review.

### 57.10.2 Add

Scope: public.

Parameter	Datatype	Description
ItemType	Integer	Indicates the registered type of the signature item.
ID	String	Unique ID of the signature item.
ItemText	String	Text to be displayed in the signature review list.
GroupName	String	Group under which item is to be displayed.
SignState	Enum	One of: 0 = Not applicable 1 = Yes 2 = No

Adds a signature item to the list.

### 57.10.3 Clear

Scope: public,

Removes all items from the signature list.

### 57.10.4 Exist

Scope: public.

Parameter	Datatype	Description
ItemType	Integer	Indicates the registered type of the signature item.
ID	String	Unique ID of the signature item.
<return value>	Boolean	True if the item exists.

Checks for the existence of a signature item.

### 57.10.5 ExistForOrder

Scope: public.

Parameter	Datatype	Description
ID	Integer	Unique ID of the signature item.
<return value>	Boolean	True if the order exists.

Checks for the existence of an order signature item with the specified ID.

### 57.10.6 RegisterType

Scope: public.

Parameter	Datatype	Description
ItemType	Integer	The ID of the item type to be registered.
GroupPrefix	String	Group prefix for this item type.

Registers an item type. An exception occurs if the item type already exists.

### 57.10.7 Remove

Scope: public.

Parameter	Datatype	Description
ItemType	Integer	Indicates the registered type of the signature item.
ID	String	Unique ID of the signature item.

Removes the specified item from the signature list.

### 57.10.8 RenameGroup

Scope: public.

Parameter	Datatype	Description
OldGroup	String	Original group name.
NewGroup	String	New group name.

Renames an existing group.

### 57.10.9 ReplaceGroup

Scope: public.

Parameter	Datatype	Description
ItemType	Integer	Indicates the registered type of the signature item.
ItemID	String	Unique ID of the signature item.
NewGroup	String	New group name.

Replaces the group name associated with a signature item.

### 57.10.10 ReplaceID

Scope: public.

Parameter	Datatype	Description
ItemType	Integer	Indicates the registered type of the signature item.
OldID	String	Unique ID of the signature item.
NewID	String	Unique ID of the signature item.

Replaces the ID associated with a signature item.

### 57.10.11 ReplaceSignState

Scope: public.

Parameter	Datatype	Description
ItemType	Integer	Indicates the registered type of the signature item.
ID	String	Unique ID of the signature item.
NewState	Enum	One of: 0 = Not applicable 1 = Yes 2 = No

Replaces the signature state for the specified signature item.

### 57.10.12 ReplaceText

Scope: public.

Parameter	Datatype	Description
ItemType	Integer	Indicates the registered type of the signature item.
ID	String	Unique ID of the signature item.
Text	String	New display text.

Replaces the display text for the specified signature item.

### 57.10.13 Review

Scope: public.

Parameter	Datatype	Description
Silent	Boolean	If true, process all signature items as unsigned without user interaction.
<return value>	Boolean	False if the signature review dialog was cancelled.

Reviews all signature items.

### 57.10.14 ReviewItems

Scope: public.

Parameter	Datatype	Description
ItemType	Integer	Indicates the registered type of the signature item.
ItemIDs	String	List of signature item ids to be reviewed.
<return value>	Boolean	False if the signature review dialog was cancelled.

Reviews selected signature items.

## 58.0 Allergies

### 58.1 Introduction

Adverse Reactions	
Agent ▲	Reaction
AMPICILLIN	RASH
ASPIRIN	RASH
PENICILLIN	HYPOTENSION;HIV...

Figure 58-1: Sample Adverse Reactions (Allergies)

The Allergies component provides a quick overview of recorded adverse reactions.

### 58.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHALLERGIES.ALLERGIES
Version	4.2.0.12
Class Identifier	{9678D129-51CF-438C-944E-A0AE7AA026A4}
Image File	BEHAllergies.ocx
Property Initializations	none
Serializable Properties	ALLOWPRINT=BOOL, AUTOSIZE=BOOL, BORDERSTYLE=ENUM, CAPTION=TEXT, CAPTIONCOLOR1=COLOR, CAPTIONCOLOR2=COLOR, CAPTIONSTYLE=ENUM, COLOR=COLOR, DEFERUPDATE=BOOL, DETAILPANE=BOOL, ORIENTATION=ENUM, LAYOUT=HIDDEN
Required Files	BEHAllergies.chm
Security Keys	none
Multiple Instances Allowed	yes
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BEHO*1.1*027001

There are no specific implementation or maintenance tasks associated with this component.

### 58.3 Routine Descriptions

This component has been assigned the namespace designation of “BEHOARCV.” The following routines are distributed:



<b>Routine</b>	<b>Description</b>
BEHOARCV	Adverse reaction cover sheet support.

## 58.4 File List

None.

## 58.5 Cross References

None.

## 58.6 Exported Options

None.

## 58.7 Exported Security Keys

None.

## 58.8 Exported Protocols

None.

## 58.9 Exported Parameters

None.

## 58.10 Exported Mail Groups

None.

## 58.11 Callable Routines

This section describes supported entry points for routines exported with this component.

### 58.11.1 RPC: BEHOARCV DETAIL

Scope: private.

<b>Parameter</b>	<b>Datatype</b>	<b>Description</b>
DFN	Pointer (#2)	Patient IEN
ADR	Pointer (#120.8)	Adverse reaction IEN.
<return value>	String List	Detail text.

Returns details about the specified adverse reaction.

## 58.11.2 RPC: BEHOARCV LIST

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient IEN
<return value>	String List	List of adverse reaction entries.

Returns a list of adverse reaction entries for populating the list view.

## 58.12 External Relations

Entity	Name	Description
Package	ADVERSE REACTION TRACKING 4.0	Uses supported APIs.

## 58.13 Internal Relations

None.

## 58.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 58.15 Components

This component supports the following properties and methods:

### 58.15.1 Properties

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ALLOWPRINT	Boolean	RW	If true, a print button will appear on the detail dialog allowing printing of the contents.
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
AUTOSIZE	Boolean	RW	If true, the component automatically resizes itself to accommodate its contents.

Property	Datatype	Access	Description
BORDERSTYLE	Enum	RW	Sets the style of the border surrounding the component. Can be one of: 0 = None 1 = Single 2 = Sunken 3 = Raised
CAPTION	String	RW	Sets the text displayed in the title bar. To justify portions of the caption text, use the “\” character to delimit the left-, center-, and right-justified portions of the caption text.
CAPTIONCOLOR1 CAPTIONCOLOR2	Color	RW	Colors to apply to the title bar. If the two colors differ and a gradient style is set, a gradient effect is created. For a standard title bar style, only the first color is applied.
CAPTIONSTYLE	Enum	RW	Sets the caption style. Can be one of: 0 = None – No caption (hides title bar) 1 = Title – Standard title bar 2 = Frame – Framed title bar (group box style) 3 = Left – Left gradient title bar 4 = Right – Right gradient title bar 5 = Center – Center gradient title bar
COLOR	Color	RW	Sets the background color of the component.
DEFERUPDATE	Boolean	RW	If true, data refresh is deferred until the component becomes visible. If false, data refresh happens immediately.
DETAILPANE	Boolean	RW	If true, a detail pane appears next to the list view that displays detail text of the selected entry. If false, detail text appears in a popup dialog when an entry is clicked.
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPERFILE	String	RW	Sets the name of the help file associated with the component.
LAYOUT	String	RW	Property representing the internal layout of the form.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
ORIENTATION	Enum	RW	Sets the orientation of the detail pane. Can be one of: 0 = Horizontal 1 = Vertical
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 59.0 VueCentric to CPRS Context Adapter

### 59.1 Introduction

This service broadcasts VueCentric<sup>®</sup> context change events to VistA Imaging and other VistA applications.

### 59.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHCONTEXTADAPTER.CONTEXTADAPTER
Version	4.2.0.9
Class Identifier	{7F143231-80B7-431B-B1D6-55D80ED2F795}
Image File	BEHContextAdapter.dll
Property Initializations	none
Serializable Properties	none
Required Files	none
Security Keys	none
Multiple Instances Allowed	no
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	yes
.Net Component	no
Associated Build	BEHO*1.1*035001

There are no specific implementation or maintenance tasks associated with this component.

### 59.3 Routine Descriptions

None.

### 59.4 File List

None.

### 59.5 Cross References

None.

### 59.6 Exported Options

None.

## 59.7 Exported Security Keys

None.

## 59.8 Exported Protocols

None.

## 59.9 Exported Parameters

None.

## 59.10 Exported Mail Groups

None.

## 59.11 Callable Routines

None.

## 59.12 External Relations

None.

## 59.13 Internal Relations

None.

## 59.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 59.15 Components

This component supports the following properties and methods:

### 59.15.1 NotifyOtherApps

Scope: private.

Parameter	Datatype	Description
AppEvent	String	The event name.
AppData	String	The event data.

Broadcasts a Windows message to subscribing applications, notifying them of the context change. The format of the message is fully compatible with that produced by the CPRS application.

## 60.0 CPRS Options

### 60.1 Introduction

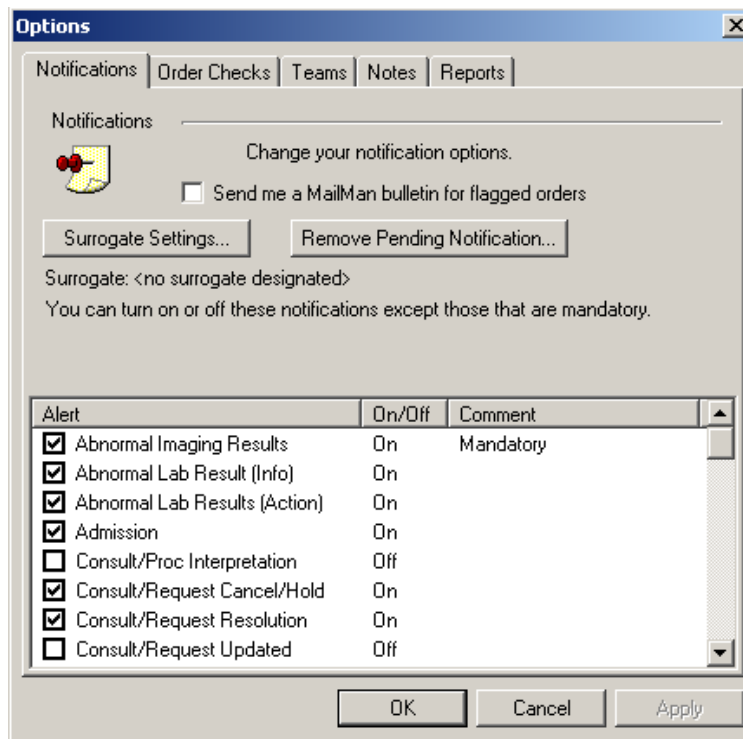


Figure 60-1: Sample Options

The CPRS Options service provides a means to access the Options dialog from a custom menu.

### 60.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHOPTIONS.OPTIONS
Version	1.1.0.17
Class Identifier	{557DA715-ACF0-43D8-9012-3FF13BD0B53D}
Image File	BEHOPTIONS.dll
Property Initializations	none
Serializable Properties	none
Required Files	none
Security Keys	none
Multiple Instances Allowed	no
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	yes
.Net Component	no
Associated Build	BEHO*1.1*022001

There are no specific implementation or maintenance tasks associated with this component.

### 60.3 Routine Descriptions

None.

### 60.4 File List

None.

### 60.5 Cross References

None.

### 60.6 Exported Options

None.

### 60.7 Exported Security Keys

None.

### 60.8 Exported Protocols

None.

### 60.9 Exported Parameters

None.

### 60.10 Exported Mail Groups

None.

### 60.11 Callable Routines

None.

## 60.12 External Relations

Entity	Name	Description
Package	ORDER ENTRY/RESULTS REPORTING 3.0	Uses supported APIs.

## 60.13 Internal Relations

Entity	Name	Description
Component	CPRS Support Library	Uses supported APIs.

## 60.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 60.15 Components

This component supports the following properties and methods:

### 60.15.1 Execute

Scope: private.

Parameter	Datatype	Description
VisibleOptions	Enum	Any combination of: 1 = Notifications 2 = Order Checks 4 = Teams 8 = Notes 16 = Reports

Invokes the Options dialog with the specified tabs visible.



## 61.0 Medication Counseling

### 61.1 Introduction

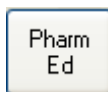


Figure 61-1: Pharmacy Ed Button

The Medication Counseling component provides a means to easily document the counseling of patients regarding their medications.

### 61.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHPHARMED.PHARMED
Version	1.0.0.192
Class Identifier	{8DAD2FB3-FFB6-48EA-AB6F-95950BD799AC}
Image File	BEHPharmED.ocx
Property Initializations	none
Serializable Properties	CAPTION=TEXT, GLYPH=IMAGE, LAYOUT=ENUM
Required Files	none
Security Keys	none
Multiple Instances Allowed	no
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
Associated Build	BEHO*1.1*044001

There are no specific implementation or maintenance tasks associated with this component.

### 61.3 Routine Descriptions

This component has been assigned the namespace designation of “BEHORXED.” The following routines are distributed:

<b>Routine</b>	<b>Description</b>
BEHORXED	Medication counseling support.

## 61.4 File List

None.

## 61.5 Cross References

None.

## 61.6 Exported Options

<b>Option</b>	<b>Type</b>	<b>Description</b>
BEHORXED COMPREHENSION DEFAULT	action	Default comprehension value.
BEHORXED COUNSEL TIME DEFAULT	action	Default counsel time.
BEHORXED DEF HOSP LOCATION	action	Allows user to specify a hospital location for visit creation. The hospital location must have a pharmacy stop code associated with it.
BEHORXED DEFAULT POV	action	Default POV.
BEHORXED DISCLAIMER TEXT	action	Edit disclaimer text.
BEHORXED EDUCATION TOPICS LIST	action	Education topics.
BEHORXED MAIN	menu	Medication counseling configuration menu.
BEHORXED POV LIST	action	POV list.
BEHORXED POV NARR TEXT	action	POV narrative text.

## 61.7 Exported Security Keys

None.

## 61.8 Exported Protocols

None.

## 61.9 Exported Parameters

Parameter	Instance Type	Value Type	Precedence	Description
BEHORXED COMPREHENSIO N DEFAULT		Set	Division, System	Allows specification of a default value for the comprehension level control. One of: 1 = Poor 2 = Fair 3 = Good 4 = Group – No Assessment 5 = Refused
BEHORXED COUNSEL TIME DEFAULT		Integer	Division, System	Define counseling time default.
BEHORXED DEF HOSP LOCATION		Pointer (#44)	Division, System	Holds a hospital location used for visit creation. Hospital location must have a pharmacy stop code.
BEHORXED DEFAULT POV		Pointer (#80)	Division, System	Selects default POV.
BEHORXED EDUCATION TOPICS LIST	Numeric (Sequence #)	Pointer (#999999 9.09)	Division, System	Identifies which Education Topics can appear in the PharmED component.
BEHORXED POV LIST	Numeric (Sequence #)	Pointer (#80)	Division, System	Identifies which POVs will appear for selection in the PharmED component.
BEHORXED DISCLAIMER TEXT		Word Processin g	Division, System	Holds disclaimer message text.
BEHORXED POV NARR TEXT	Numeric (Sequence #)	Text	Division, System	Holds the Narrative Text value. The data sequence of this parameter must match the order of the BEHORXED POV LIST parameter.

## 61.10 Exported Mail Groups

None.

## 61.11 Callable Routines

This section describes supported entry points for routines exported with this component.

### 61.11.1 RPC: BEHORXED CANUSE

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient IEN.
<return value>	Boolean	Return use status.

Returns true if component can be used for this patient.

## 61.11.2 RPC: BEHORXED COMPLST

Scope: private.

Parameter	Datatype	Description
<return value>	String List	Returns level of understanding codes from V PATIENT ED file.

Returns level of understanding codes from V PATIENT ED file.

## 61.11.3 RPC: BEHORXED EDLST

Scope: private.

Parameter	Datatype	Description
<return value>	String List	List of selectable education topics.

Returns a list of selectable education topics.

## 61.11.4 RPC: BEHORXED POVLST

Scope: private.

Parameter	Datatype	Description
<return value>	String List	List of selectable POVs.

Returns a list of selectable POVs.

## 61.11.5 RPC: BEHORXED PRVNRPC

Scope: private.

Parameter	Datatype	Description
TXT	String	Text to look up.
<return value>	Pointer (#9999999.27)	Provider Narrative IEN matching TXT.

Returns the IEN of the provider narrative matching the specified text. Creates an entry if necessary.

## 61.11.6 RPC: BEHORXED STORE

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient IEN.
VSTR	String	Visit string.
PCCARY	String Array	Patient education data to store.
<return value>	Integer	Nonzero if an error occurred.

Stores patient education data.

## 61.11.7 RPC: BEHORXED VSTLST

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient IEN.
SDT	FM Date	Date to check for visits.
CAT (optional)	String	Service category (defaults to "A").
<return value>	String List	List of visits.

Returns a list of available visits matching the specified criteria.

## 61.12 External Relations

Entity	Name	Description
File	EDUCATION TOPICS (#9999999.09)	Read access.
File	V PATIENT ED (#9000010.16)	Read and write access.

## 61.13 Internal Relations

None.

## 61.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 61.15 Components

This component supports the following properties and methods:

### 61.15.1 Properties

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
AUTOSIZE	Boolean	RW	If true, the component automatically resizes itself to accommodate its contents.

Property	Datatype	Access	Description
BORDERSTYLE	Enum	RW	Sets the style of the border surrounding the component. Can be one of: 0 = None 1 = Single 2 = Sunken 3 = Raised
CAPTION	String	RW	Sets the text displayed in the button caption.
COLOR	Color	RW	Sets the background color of the component.
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
GLYPH	String	RW	Name of bitmap file containing glyphs to be displayed on button surface.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPPFILE	String	RW	Sets the name of the help file associated with the component.
LAYOUT	String	RW	Property representing the internal layout of the form.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
NUMGLYPHS	Integer	RW	Number of glyphs contained in glyph file.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 62.0 Primary Care Information Header

### 62.1 Introduction

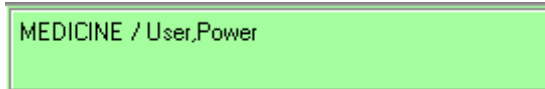


Figure 62-1: Sample Primary Care Information

The Primary Care Information Header displays information about the patient's primary care physician and team.

### 62.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmable Identifier	BEHPRIMARYCARE.PRIMARYCARE
Version	4.2.0.8
Class Identifier	{A0745103-C850-41B6-AD9E-172273930A7C}
Image File	BEHPrimaryCare.ocx
Property Initializations	MINHEIGHT=50, MINWIDTH=200
Serializable Properties	AUTOSIZE=BOOL, BORDERSTYLE=ENUM, COLOR=COLOR
Required Files	
Security Keys	none
Multiple Instances Allowed	yes
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BEHO*1.1*003001

There are no specific implementation or maintenance tasks associated with this component.

### 62.3 Routine Descriptions

None.

### 62.4 File List

None.

### 62.5 Cross References

None.

## 62.6 Exported Options

None.

## 62.7 Exported Security Keys

None.

## 62.8 Exported Protocols

None.

## 62.9 Exported Parameters

None.

## 62.10 Exported Mail Groups

None.

## 62.11 Callable Routines

None.

## 62.12 External Relations

Entity	Name	Description
File	PRIMARY CARE TEAMS (#9009017.5)	Read access.
File	BDP DESG SPECIALTY PROVIDER (#90360.1)	Read access.
File	BDP DESG SPEC PROV CATEGORY (#90360.3)	Read access.

## 62.13 Internal Relations

None.

## 62.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 62.15 Components

This component supports the following properties and methods:



## 62.15.1 Properties

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
AUTOSIZE	Boolean	RW	If true, the component automatically resizes itself to accommodate its contents.
BORDERSTYLE	Enum	RW	Sets the style of the border surrounding the component. Can be one of: 0 = None 1 = Single 2 = Sunken 3 = Raised
COLOR	Color	RW	Sets the background color of the component.
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPPFILE	String	RW	Sets the name of the help file associated with the component.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
MINHEIGHT	Integer	RW	Sets the minimum height, in pixels, that the component can attain.
MINWIDTH	Integer	RW	Sets the minimum width, in pixels, that the component can attain.
THEMEAWARE	Boolean	RW	If true, the component is rendered as a themed button.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 63.0 Spell Checking Service

### 63.1 Introduction

The Spell Checking Service provides spell checking capabilities to other components using the Microsoft Word spell checker.

### 63.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHSPELLCHECK.SPELLCHECK
Version	1.0.1.10
Class Identifier	{8C4D6F72-25BD-4A45-A334-75491634037D}
Image File	BEHSpellCheck.dll
Property Initializations	ENABLED=@BEHOSP ENABLE SPELLCHECKER
Serializable Properties	none
Required Files	Interop.BEHSpellCheck.dll
Security Keys	none
Multiple Instances Allowed	no
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	yes
.Net Component	no
Associated Build	BEHO*1.1*023001

There are no specific implementation or maintenance tasks associated with this component.

### 63.3 Routine Descriptions

This component has been assigned the namespace designation of “BEHOSP.” The following routines are distributed:

Routine	Description
BEHOSPUT	Spell checking support.

### 63.4 File List

None.

### 63.5 Cross References

None.

## 63.6 Exported Options

Option	Type	Description
BEHOSP ENABLE SPELLCHECKER	action	Enable spellchecking service.
BEHOSP MAIN	menu	Spellchecking configuration menu.
BEHOSP SERVICE PLUGIN	action	Specify spellchecking service plugin.

## 63.7 Exported Security Keys

None.

### 63.7.1 Exported Protocols

None.

### 63.7.2 Exported Parameters

Parameter	Instance Type	Value Type	Precedence	Description
BEHOSP ENABLE SPELLCHECKER		Boolean	User, Division, System	If set to YES, the spellchecking service is enabled. If NO, it is disabled. Note that the spellchecking service can require additional software to be installed on the workstation (e.g., MS Word). If these additional requirements are not met, the service will not be enabled regardless of this setting.
BEHOSP SERVICE PLUGIN		Pointer (#19930.2)	User, Division, System	This is the programmatic identifier of the spellchecker service to be used within the application.

## 63.8 Exported Mail Groups

None.

## 63.9 Callable Routines

This section describes supported entry points for routines exported with this component.

### 63.9.1 \$\$SVCSCN^BEHOSPUT

Scope: private.

Parameter	Datatype	Description
IEN	Pointer (#19930.2)	IEN of object registry entry to screen.
<return value>	Boolean	Returns true if object belong to the spell checking category.

Used to screen entries for the BEHOSP SERVICE PLUGIN parameter.

## 63.10 External Relations

None.

## 63.11 Internal Relations

None.

## 63.12 Archiving and Purging

There are no archiving or purging requirements within this software.

## 63.13 Components

This component supports the following properties and methods:

### 63.13.1 Properties

Property	Datatype	Access	Description
Capabilities	Enum	R	Describes the capabilities of the spell check engine. Any combination of: 1 = Spell check 2 = Grammar check 4 = Custom dictionaries
Text	String	RW	Text to be spell/grammar checked. Text can be modified as the result of a spell or grammar check operation.
Changes	Integer	R	Count of changes made during spell/grammar checking.
Errors	Integer	R	Count of spelling/grammar errors encountered.
EngineID	String	R	Identity of spell checking engine.
Enabled	Boolean	RW	Enable/disable spell checking service. This can be modified only if not disabled at the system level.

### 63.13.2 AddDictionary

Scope: public.

Parameter	Datatype	Description
FileName	String	Full path and filename for custom dictionary.
<return value>	Integer	Index of the dictionary that was added, or -1 if not added.

Adds a custom dictionary.

### 63.13.3 GrammarCheck

Scope: public.

Parameter	Datatype	Description
<return value>	Boolean	True if grammar check was completed.

Performs a grammar check on data in Text.

### 63.13.4 RemoveDictionary

Scope: public.

Parameter	Datatype	Description
FileName	String	Full path and filename for custom dictionary.
<return value>	Integer	Count of custom dictionaries remaining.

Removes a custom dictionary. FileName must be identical to that used to register the custom dictionary using AddDictionary.

### 63.13.5 Reset

Scope: public.

Clears the text buffer and resets the Changes and Errors properties to zero.

### 63.13.6 ShowOptions

Scope: public.

Displays an option dialog, if available, that permits setting spell checking parameters. This dialog will be specific to the spell check engine being used.

#### 63.13.6.1 SpellCheck

Scope: public.

Parameter	Datatype	Description
<return value>	Boolean	Returns true if spell check was completed.

Performs a spell check operation data in Text.

## 64.0 Appointments

### 64.1 Introduction

Appointments and Visits		
Date ▼	Appointment/Visit	Status ▲
25-Apr-2007 13:01	ADOLESCENT	AMBULATORY
15-Mar-2007 21:04	ACU WARD	AMBULATORY
12-Dec-2006 12:00	Unknown	EVENT (HISTORICAL)
08-Dec-2006 12:50	2ND FLU SHOT	AMBULATORY
08-Dec-2006 12:45	ADOLESCENT	AMBULATORY
08-Dec-2006 12:43	CHEST CLINIC	AMBULATORY
08-Dec-2006 12:00	Unknown	EVENT (HISTORICAL)
07-Dec-2006 12:00	Unknown	EVENT (HISTORICAL)
06-Dec-2006 11:54	ADULT WALKIN	AMBULATORY
04-Dec-2006 12:00	Unknown	EVENT (HISTORICAL)
03-Dec-2006 12:00	Unknown	EVENT (HISTORICAL)

Figure 64-1: Sample Appointments and Visits

The Appointments component provides a quick overview of a patient's future appointments and past visits for display on the cover sheet.

### 64.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHVISITS.VISITS
Version	4.2.0.5
Class Identifier	{71CABD3B-EA1A-47FD-ADE0-0039D97A6ED1}
Image File	BEHVisits.ocx
Property Initializations	none
Serializable Properties	ALLOWPRINT=BOOL, AUTOSIZE=BOOL, BORDERSTYLE=ENUM, CAPTION=TEXT, CAPTIONCOLOR1=COLOR, CAPTIONCOLOR2=COLOR, CAPTIONSTYLE=ENUM, COLOR=COLOR, DEFERUPDATE=BOOL, DETAILPANE=BOOL, ORIENTATION=ENUM, LAYOUT=HIDDEN
Required Files	BEHVisits.chm
Security Keys	none
Multiple Instances Allowed	yes
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	BEHO*1.1*037001

There are no specific implementation or maintenance tasks associated with this component.

## 64.3 Routine Descriptions

This component has been assigned the namespace designation of “BEHOENCV.” The following routines are distributed:

Routine	Description
BEHOENCV	Appointment/visit cover sheet support.

## 64.4 File List

None.

## 64.5 Cross References

None.

## 64.6 Exported Options

None.

## 64.7 Exported Security Keys

None.

## 64.8 Exported Protocols

None.

## 64.9 Exported Parameters

None.

## 64.10 Exported Mail Groups

None.

## 64.11 Callable Routines

This section describes supported entry points for routines exported with this component.

### 64.11.1 RPC: BEHOENCV DETAIL

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient IEN.
VST	String	Visit string.
<return value>	String List	Detail text.

Returns detailed information about the specified visit or appointment.

## 64.11.2 RPC: BEHOENCV LIST

Scope: private.

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient IEN.
<return value>	String List	List of appointment and/or visits.

Returns a list of appointment/visits for populating the list view.

## 64.12 External Relations

None.

## 64.13 Internal Relations

Entity	Name	Description
Component	Encounter Context	Uses supported APIs.

## 64.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 64.15 Components

This component supports the following properties and methods:

### 64.15.1 Properties

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ALLOWPRINT	Boolean	RW	If true, a print button will appear on the detail dialog allowing printing of the contents.



Property	Datatype	Access	Description
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
AUTOSIZE	Boolean	RW	If true, the component automatically resizes itself to accommodate its contents.
BORDERSTYLE	Enum	RW	Sets the style of the border surrounding the component. Can be one of: 0 = None 1 = Single 2 = Sunken 3 = Raised
CAPTION	String	RW	Sets the text displayed in the title bar. To justify portions of the caption text, use the “\” character to delimit the left-, center-, and right-justified portions of the caption text.
CAPTIONCOLOR1 CAPTIONCOLOR2	Color	RW	Colors to apply to the title bar. If the two colors differ and a gradient style is set, a gradient effect is created. For a standard title bar style, only the first color is applied.
CAPTIONSTYLE	Enum	RW	Sets the caption style. Can be one of: 0 = None – No caption (hides title bar) 1 = Title – Standard title bar 2 = Frame – Framed title bar (group box style) 3 = Left – Left gradient title bar 4 = Right – Right gradient title bar 5 = Center – Center gradient title bar
COLOR	Color	RW	Sets the background color of the component.
DEFERUPDATE	Boolean	RW	If true, data refresh is deferred until the component becomes visible. If false, data refresh happens immediately.
DETAILPANE	Boolean	RW	If true, a detail pane appears next to the list view that displays detail text of the selected entry. If false, detail text appears in a popup dialog when an entry is clicked.
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPPFILE	String	RW	Sets the name of the help file associated with the component.
LAYOUT	String	RW	Property representing the internal layout of the form.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
ORIENTATION	Enum	RW	Sets the orientation of the detail pane. Can be one of: 0 = Horizontal 1 = Vertical
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.

<b>Property</b>	<b>Datatype</b>	<b>Access</b>	<b>Description</b>
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 65.0 Message Broadcast

### 65.1 Introduction

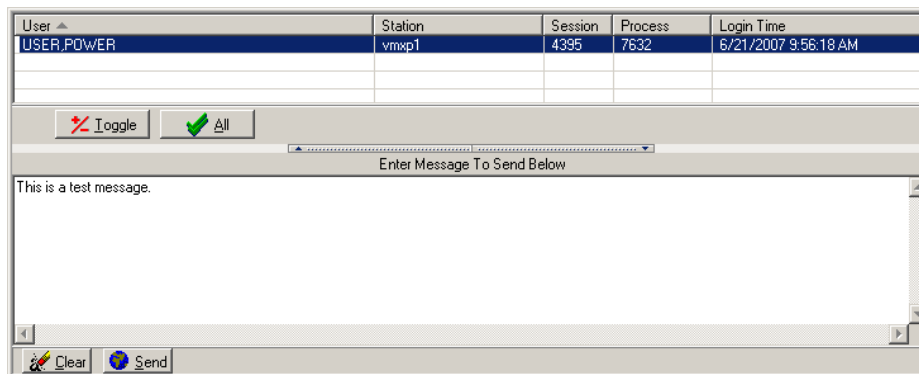


Figure 65-1: Sample message Broadcast

The Message Broadcast component permits sending a message to one or more EHR sessions. The received message appears as a popup dialog.

### 65.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	VCBROADCAST.BROADCAST
Version	4.2.0.4
Class Identifier	{E8B930A9-7B86-4B81-894B-FC96AA22AF7C}
Image File	vcBroadcast.ocx
Property Initializations	none
Serializable Properties	AUTOSIZE=BOOL, BORDERSTYLE=ENUM, CAPTION=TEXT, CAPTIONCOLOR1=COLOR, CAPTIONCOLOR2=COLOR, CAPTIONSTYLE=ENUM, COLOR=COLOR, DEFERUPDATE=BOOL, ORIENTATION=ENUM, LAYOUT=HIDDEN
Required Files	none
Security Keys	none
Multiple Instances Allowed	yes
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	CIAO*1.1*008001

There are no specific implementation or maintenance tasks associated with this component.

### 65.3 Routine Descriptions

None.

### 65.4 File List

None.

### 65.5 Cross References

None.

### 65.6 Exported Options

None.

### 65.7 Exported Security Keys

None.

### 65.8 Exported Protocols

None.

### 65.9 Exported Parameters

None.

### 65.10 Exported Mail Groups

None.

### 65.11 Callable Routines

None.

### 65.12 External Relations

None.

### 65.13 Internal Relations

None.

### 65.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 65.15 Components

This component supports the following properties and methods:

### 65.15.1 Properties

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ALLOWPRINT	Boolean	RW	If true, a print button will appear on the detail dialog allowing printing of the contents.
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
AUTOSIZE	Boolean	RW	If true, the component automatically resizes itself to accommodate its contents.
BORDERSTYLE	Enum	RW	Sets the style of the border surrounding the component. Can be one of: 0 = None 1 = Single 2 = Sunken 3 = Raised
CAPTION	String	RW	Sets the text displayed in the title bar. To justify portions of the caption text, use the “\” character to delimit the left-, center-, and right-justified portions of the caption text.
CAPTIONCOLOR1 CAPTIONCOLOR2	Color	RW	Colors to apply to the title bar. If the two colors differ and a gradient style is set, a gradient effect is created. For a standard title bar style, only the first color is applied.
CAPTIONSTYLE	Enum	RW	Sets the caption style. Can be one of: 0 = None – No caption (hides title bar) 1 = Title – Standard title bar 2 = Frame – Framed title bar (group box style) 3 = Left – Left gradient title bar 4 = Right – Right gradient title bar 5 = Center – Center gradient title bar
COLOR	Color	RW	Sets the background color of the component.
DEFERUPDATE	Boolean	RW	If true, data refresh is deferred until the component becomes visible. If false, data refresh happens immediately.

Property	Datatype	Access	Description
DETAILPANE	Boolean	RW	If true, a detail pane appears next to the list view that displays detail text of the selected entry. If false, detail text appears in a popup dialog when an entry is clicked.
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPPFILE	String	RW	Sets the name of the help file associated with the component.
LAYOUT	String	RW	Property representing the internal layout of the form.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
ORIENTATION	Enum	RW	Sets the orientation of the detail pane. Can be one of: 0 = Horizontal 1 = Vertical
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 66.0 Chat Service

### 66.1 Introduction

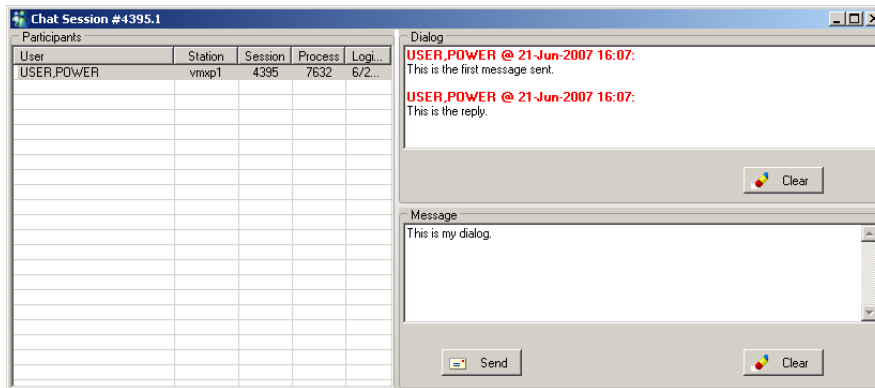


Figure 66-1: Sample Chat Service

The Chat Service enables users to communicate with one another.

### 66.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	VCCHATSERVICE.CHATSERVICE
Version	1.0.1.3
Class Identifier	{F5FD6702-1C56-4C78-850C-1F3552969E43}
Image File	vcChatService.dll
Property Initializations	
Serializable Properties	
Required Files	
Security Keys	none
Multiple Instances Allowed	no
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	yes
.Net Component	no
Associated Build	CIAO*1.1*002001

There are no specific implementation or maintenance tasks associated with this component.

### 66.3 Routine Descriptions

None.

### 66.4 File List

None.

## 66.5 Cross References

None.

## 66.6 Exported Options

None.

## 66.7 Exported Security Keys

None.

## 66.8 Exported Protocols

None.

## 66.9 Exported Parameters

None.

## 66.10 Exported Mail Groups

None.

## 66.11 Callable Routines

None.

## 66.12 External Relations

None.

## 66.13 Internal Relations

None.

## 66.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 66.15 Components

This component supports the following properties and methods:



## 66.15.1 IChatService

### 66.15.1.1 Properties

Property	Datatype	Access	Description
Listening	Boolean	RW	Set listening mode for chat service. If the service is listening, it will show up in the list of potential participants and will respond to invitation requests.

### 66.15.1.2 NewSession

Scope: public.

Starts a new chat session.

## 66.15.2 IChatSession

### 66.15.2.1 Properties

Property	Datatype	Access	Description
ID	String	R	Unique identifier for the session.

## 67.0 Internet Explorer

### 67.1 Introduction

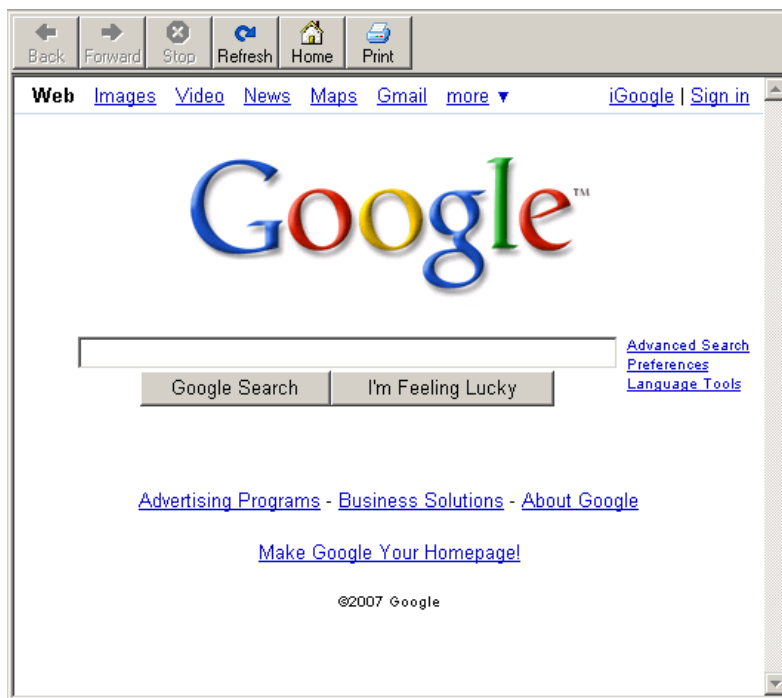


Figure 67-1: Sample of Internet Explorer

The Internet Explorer component is a wrapper for Internet Explorer itself and is useful for imbedding Web content with the EHR application.

### 67.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	VCIEXPLORER.IEXPLORER
Version	4.2.1.14
Class Identifier	{3C15F2D9-D949-4967-ACD2-E94874E07519}
Image File	vcIExplorer.ocx
Property Initializations	none

Entity	Value
Serializable Properties	ALLOWNEWWINDOW=BOOL, AUTOSIZE=BOOL, BORDERSTYLE=ENUM, CAPTION=TEXT, CAPTIONCOLOR1=COLOR, CAPTIONCOLOR2=COLOR, CAPTIONSTYLE=ENUM, COLOR=COLOR, HOMEPAGE=TEXT, RESETONPATIENTCHANGE=BOOL, TOOLBARVISIBLE=BOOL
Required Files	none
Security Keys	none
Multiple Instances Allowed	yes
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	CIAO*1.1*003001

There are no specific implementation or maintenance tasks associated with this component.

### 67.3 Routine Descriptions

None.

### 67.4 File List

None.

### 67.5 Cross References

None.

### 67.6 Exported Options

None.

### 67.7 Exported Security Keys

None.

## 67.8 Exported Protocols

None.

## 67.9 Exported Parameters

None.

## 67.10 Exported Mail Groups

None.

## 67.11 Callable Routines

None.

## 67.12 External Relations

None.

## 67.13 Internal Relations

None.

## 67.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 67.15 Components

This component supports the following properties and methods:

### 67.15.1 Properties

Property	Datatype	Access	Description
AllowNewWindow	Boolean	RW	If true, popup windows are allowed. Some web pages cannot function properly if this is set to false.
HomePage	String	RW	The URL for the default home page.
HTML	String	W	Allows placing HTML content directly into the browser.
ResetOnPatientChange	Boolean	RW	If true, the browser is reset to its home page when the patient context changes. Use this if the browser is used to display patient-specific content.
ToolbarVisible	Boolean	RW	If true, the browser toolbar appears at the top of the form.

## 68.0 Image

### 68.1 Introduction

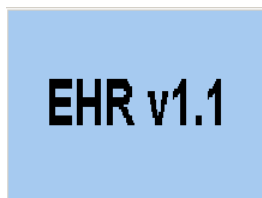


Figure 68-1: Sample Image

The Image component can be used to imbed a graphic within the application. Multiple image formats are supported.

### 68.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	VCIMAGE.IMAGEX
Version	4.1.0.16
Class Identifier	{2F154977-4CCA-4EF2-AC6D-7070ADF10D7E}
Image File	vcImage.ocx
Property Initializations	
Serializable Properties	BORDERSTYLE=ENUM, FILENAME=FILE, STRETCH=BOOL
Required Files	none
Security Keys	none
Multiple Instances Allowed	yes
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	CIAO*1.1*004001

There are no specific implementation or maintenance tasks associated with this component.

### 68.3 Routine Descriptions

None.

### 68.4 File List

None.

## 68.5 Cross References

None.

## 68.6 Exported Options

None.

## 68.7 Exported Security Keys

None.

## 68.8 Exported Protocols

None.

## 68.9 Exported Parameters

None.

## 68.10 Exported Mail Groups

None.

## 68.11 Callable Routines

None.

## 68.12 External Relations

None.

## 68.13 Internal Relations

None.

## 68.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 68.15 Components

This component supports the following properties and methods:

## 68.15.1 Properties

<b>Property</b>	<b>Datatype</b>	<b>Access</b>	<b>Description</b>
FileName	String	RW	Full path and file name of the graphic.
Stretch	Boolean	RW	If true, the image is stretched to the dimensions of the component.

## 69.0 Program Launcher

### 69.1 Introduction



Figure 69-1: Sample Program Launcher Button

The Program Launcher is a button-based component that can be used to launch an external program from with the EHR application.

### 69.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	VCLAUNCHER.LAUNCHER
Version	4.1.2.3
Class Identifier	{A09EFD91-9666-4826-B89C-8A257AA843A0}
Image File	vcLauncher.ocx
Property Initializations	none
Serializable Properties	CAPTION=TEXT, COLOR=COLOR, COMMANDLINE=TEXT, EXENAME=FILE, GLYPH=IMAGE, LAYOUT=ENUM, NUMGLYPHS=INT
Required Files	none
Security Keys	none
Multiple Instances Allowed	yes
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	CIAO*1.1*005001

There are no specific implementation or maintenance tasks associated with this component.

### 69.3 Routine Descriptions

None.

### 69.4 File List

None.

### 69.5 Cross References

None.



## 69.6 Exported Options

None.

## 69.7 Exported Security Keys

None.

## 69.8 Exported Protocols

None.

## 69.9 Exported Parameters

None.

## 69.10 Exported Mail Groups

None.

## 69.11 Callable Routines

None.

## 69.12 External Relations

None.

## 69.13 Internal Relations

None.

## 69.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 69.15 Components

This component supports the following properties and methods:

### 69.15.1 Properties

Property	Datatype	Access	Description
CommandLine	String	RW	Specifies any additional command line parameters to be passed to the external application.

<b>Property</b>	<b>Datatype</b>	<b>Access</b>	<b>Description</b>
ExeName	String	RW	Full path and file name of the executable image of the external application.
WindowState	Enum	RW	Controls the initial appearance of the external application. Can be one of: 0 = Hide 1 = Normal 2 = Minimized 3 = Maximized

## 70.0 Patient Photo

### 70.1 Introduction



Figure 70-1: Sample Patient Photo

The Patient Photo component enables the display of patient photographs.

### 70.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	VCPATPHOTO.PATPHOTO
Version	4.1.1.5
Class Identifier	{A03F898E-D8D3-4817-8E4B-009D61B55327}
Image File	vcPatPhoto.ocx
Property Initializations	CAPTION=
Serializable Properties	BORDERSTYLE=ENUM, CAPTION=TEXT, COLOR=COLOR
Required Files	
Security Keys	none
Multiple Instances Allowed	yes
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	CIAO*1.1*009001

There are no specific implementation or maintenance tasks associated with this component.

### 70.3 Routine Descriptions

None.

### 70.4 File List

None.

### 70.5 Cross References

None.

## 70.6 Exported Options

None.

## 70.7 Exported Security Keys

None.

## 70.8 Exported Protocols

None.

## 70.9 Exported Parameters

None.

## 70.10 Exported Mail Groups

None.

## 70.11 Callable Routines

None.

## 70.12 External Relations

None.

## 70.13 Internal Relations

None.

## 70.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 70.15 Components

This component supports the following properties and methods:

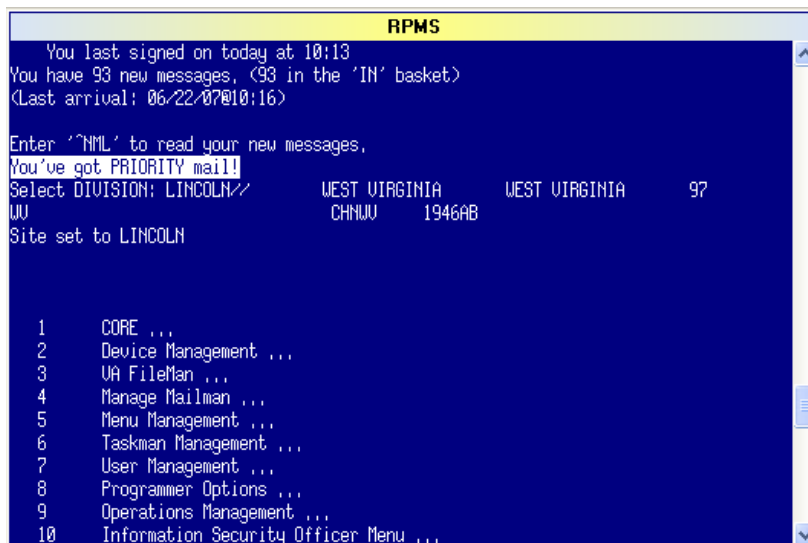
### 70.15.1 Properties

The properties are described in the following table.

Property	Datatype	Access	Description
ALIGN	Enum	RW	Sets the alignment of the component relative to its parent. One of: 0 = None – no alignment occurs 1 = Top – aligns to the top boundary of the parent 2 = Bottom – aligns to the bottom boundary of the parent 3 = Left – aligns to the left boundary of the parent 4 = Right – aligns to the right boundary of the parent 5 = All – expands to the dimensions of the parent 6 = Center – centers itself within the parent
ANCHORS	Flag	RW	Anchors the component's position relative to its parent. Zero or more of: 1 = Top 2 = Left 4 = Right 8 = Bottom
AUTOSIZE	Boolean	RW	If true, the component automatically resizes itself to accommodate its contents.
BORDERSTYLE	Enum	RW	Sets the style of the border surrounding the component. Can be one of: 0 = None 1 = Single 2 = Sunken 3 = Raised
COLOR	Color	RW	Sets the background color of the component.
FONT	Font	RW	Set the default font used by the component. Some elements of a component can override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPPFILE	String	RW	Sets the name of the help file associated with the component.
LEFT	Integer	RW	Sets the position (in pixels) of the left boundary of the component.
MINHEIGHT	Integer	RW	Sets the minimum height, in pixels, that the component can attain.
MINWIDTH	Integer	RW	Sets the minimum width, in pixels, that the component can attain.
THEMEAWARE	Boolean	RW	If true, the component is rendered as a themed button.
TOP	Integer	RW	Sets the position (in pixels) of the top boundary of the component.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

## 71.0 Telnet

### 71.1 Introduction



```

RPMS
You last signed on today at 10:13
You have 93 new messages, (93 in the 'IN' basket)
(Last arrival: 06/22/07@10:16)

Enter '^NML' to read your new messages,
You've got PRIORITY mail!
Select DIVISION: LINCOLN//      WEST VIRGINIA      WEST VIRGINIA      97
WU          CHNWU      1946AB
Site set to LINCOLN

1  CORE ...
2  Device Management ...
3  UA FileMan ...
4  Manage Mailman ...
5  Menu Management ...
6  Taskman Management ...
7  User Management ...
8  Programmer Options ...
9  Operations Management ...
10 Information Security Officer Menu ...

```

Figure 71-1: Sample RPMS Screen

The Telnet component permits connecting to the character-based (“roll-and-scroll”) interface of the host system.

### 71.2 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	VCTELNET.TELNETX
Version	1.1.1.15
Class Identifier	{1DBA08B6-0FCD-49EF-A977-93A46D332A4C}
Image File	vcTelnet.ocx
Property Initializations	REMOTEHOST=\$(SESSION.HOSTADDRESS)
Serializable Properties	AUTORESIZE=BOOL, BACKCOLOR=COLOR, BOLDCOLOR=COLOR, BORDERSTYLE=ENUM, CAPTION=TEXT, CAPTIONCOLOR1=COLOR, CAPTIONCOLOR2=COLOR, CAPTIONSTYLE=ENUM, FORECOLOR=COLOR
Required Files	vcTelnet.lic
Security Keys	none
Multiple Instances Allowed	no
Internal Property Editor	no
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	no
.Net Component	no
Associated Build	CIAO*1.1*006001

A license file is required to use the Telnet component. A default license file is delivered with the component that permits its use in demonstration mode that imposes a five minute limit on connection time. To use the component in a production environment, you must contact the vendor to purchase a license.

### 71.3 Routine Descriptions

None.

### 71.4 File List

None.

### 71.5 Cross References

None.

### 71.6 Exported Options

None.

### 71.7 Exported Security Keys

None.

### 71.8 Exported Protocols

None.

### 71.9 Exported Parameters

None.

### 71.10 Exported Mail Groups

None.

### 71.11 Callable Routines

None.

## 71.12 External Relations

Entity	Name	Description
Libraries	DartTelnet.dll, DartSock.dll, DartVT.dll	Third-party redistributable libraries from Dart Technologies.
License	vcTelnet.lic	License file purchased from vendor (Medsphere Systems Corporation) is required for production use.

## 71.13 Internal Relations

None.

## 71.14 Archiving and Purging

There are no archiving or purging requirements within this software.

## 71.15 Components

This component supports the following properties and methods:

### 71.15.1 Properties

Property	Datatype	Access	Description
AnswerBack	String	RW	The character string returned by an answerback request.
AutoConnect	Boolean	RW	Time, in milliseconds, to wait for a connection to be established.
AutoPrint	Boolean	RW	If true, each line sent to the screen is also sent to the default printer.
AutoRepeat	Boolean	RW	If true, depressed keys generate more than one character.
AutoResize	Boolean	RW	If true, the font size is automatically resized as the screen is resized.
AutoWrap	Boolean	RW	If true, text automatically wraps at the right edge of the screen.
BackColor	Color	RW	The background color of the display.
BoldColor	Color	RW	The background color of the display.
BufferRows	Integer	RW	The maximum number of rows stored in the screen buffer.
Columns	Enum	RW	The maximum character width of the screen. One of: 0 = 80 columns 1 = 132 columns
Connect	Boolean	RW	If true, an active connection exists.
CursorStyle	Enum	RW	Affects cursor appearance. One of: 0 = Block 1 = Underline
FontBold	Boolean	RW	If true, a bold display font is selected.



Property	Datatype	Access	Description
FontSize	Integer	RW	The point size of the display font.
ForeColor	Color	RW	The foreground (text) color of the display.
RemoteHost	String	RW	The address or name of the remote host. Replaceable parameters are allowed.
RemotePort	Integer	RW	The port number for the connection. The default Telnet port is 23.
Rows	Integer	RW	The maximum number of rows that can be displayed without scrolling.
ScrollType	Enum	RW	Controls the manner in which text scrolls vertically. One of: 0 = Jump 1 = Smooth
StatusLine	Boolean	RW	If true, a status line appears at the bottom of the screen.
TabStops	String	RW	A list of comma-delimited tab positions. The default value is: 9,17,25,33,41,49,57,65,73,81,89,97,105,113,121,129
TermType	Enum	RW	Terminal emulation mode. One of: 0 = TTY 1 = VT52 2 = VT100 3 = VT220(7) 4 = VT220(8) 5 = VT320(7) 6 = VT320(8)
TimeOut	Integer	RW	Time, in milliseconds, to wait for a connection to be established.

## 72.0 Glossary

<b>Term</b>	<b>Definition</b>
Application Context	This is an entry in the Option file that provides access control at the client application level. The Broker passes this information on the initial connection request. A user must have access to the given option to establish a broker connection.
Broker	Software that marshals remote procedure requests between a client application and a remote host. Also known as RPC Broker.
CCOW	Common Context Object Workgroup – an HL7-sponsored workgroup responsible for specifying standards for context exchange among applications.
Context Object	A specialized service that maintains a common context for a specific entity (e.g., patient or encounter) and supports the context change event interface.
CMS	Component Management Service.
CSL	Communication Services Layer.
CSS	Component Support Services
Daemon	A background process that performs a specified service.
EHR	Electronic Health Record.
Listener	A daemon that monitors a specified TCP port and handles communications between the client and remote host.
Primary Listener Daemon	The daemon that listens for the initial connection request. The primary listener daemon immediately passes the connection to a secondary listener daemon.
Remote Procedure	A procedure residing on a remote host that can be invoked by a client process through a broker intermediary.
Remote Procedure Call	The act of invoking a remote procedure. This is often used interchangeably with the term “Remote Procedure,” especially using its abbreviated form “RPC.”
RPC	Remote Procedure Call
RPC Context	This is an entry in the Option file that provides access control at the remote procedure level. The Broker passes this information with each remote procedure request. A user must have access to the given option to invoke the associated remote procedure.
RPMS	Resource and Patient Management System
Secondary Listener Daemon	The daemon that handles communication interchange between the client and remote host processes.
Session Context	This refers to the persistent state information associated with an establish session.
VIM	Visual Interface Manager

## 73.0 Appendix I – Developer Tutorial

This tutorial targets software developers wishing to learn the skills and techniques necessary to create components for the VueCentric<sup>®</sup> Framework. While the document provides examples for the Delphi and Visual Basic user, it should prove to be of use to developers using other programming languages as well. Upon completion of this tutorial, the software developer will have an in-depth understanding of the VueCentric<sup>®</sup> Framework and its major components and how to create visual and non-visual plug-in components.

**Note:** The examples in this document apply to version 6.0 of Delphi and Visual Basic and Visual Studio.Net 2003. Other versions can differ somewhat from the examples shown.

### 73.1 Introduction

The first VistA/RPMS applications sported relatively simple, character-based user interfaces. This limited in a significant way the types of user interactions that could be supported. With the advent of client-server programming by way of the remote procedure call broker (aka, RPC Broker), developers were empowered to create much more sophisticated user interfaces supported under the Windows operating system. However, being of the traditional monolithic design, applications so developed were bulky, integrated poorly with other applications, did little to embrace code reuse, and generally were not very extensible. The VueCentric<sup>®</sup> Framework was developed to address these deficiencies by creating an enabling infrastructure that supports the deployment of applications as reusable components rather than standalone executables. To this end, the Framework provides a number of benefits to the software developer:

- Version-control and Automated Deployment
- User Authentication and Access Control
- Support for Common Tasks
- Debugging Support
- Synchronous and Asynchronous Remote Data Access
- Event Subscription/Propagation
- Choice of Software Development Tools
- Collaborative Software Development Environment
- Context Management
- Visual Design Tool

A component can be either a visual component (ActiveX) or a service (in-process COM server). An example of a visual component might be one that supports management of the problem list. In contrast, a component that maintains information about the currently selected patient would be an example of a service. Note that services can manifest themselves visually (e.g., the patient selection dialog produced

by the patient context object), but in their baseline state, they remain invisible to the user. While the initial creation of these two component types differs somewhat, the programming techniques for both are essentially the same.

## 73.2 Using Debug Mode

Debug mode is activated when the debug command line parameter is specified at application startup. This feature permits debugging of remote procedure calls on the remote host. Normally, when a connection request is made, a dedicated background process is automatically created on the server to handle communications with the client. In contrast, in debug mode when a connection request is made the following dialog appears:

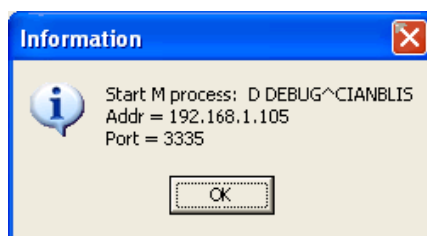


Figure 73-1: Information Dialog

This dialog instructs you to manually start the listener process on the target machine using the specified entry point. Be sure to set any breakpoints you might need before doing this if your M environment requires this. After starting the listener in this manner, it will prompt you for an IP address and port to use. Enter the information supplied in the dialog shown above. Once you do this, you can clear the dialog by clicking the OK button. The remainder of the connection process will resume in the usual manner.

Some special notes of interest:

- If you close the above dialog before manually starting a listener, a background listener will be used instead.
- If you use the Serenji debugger, be sure to start the listener after invoking the Serenji shell.
- Using debug mode with some network configurations can fail to establish a connection. This is especially true for some VPNs and networks where NAT is utilized. Because debug mode performs a connection callback from the remote host to the client, the remote host must be able to determine the route to the client's IP address. If a VPN is in use, supply the VPN-assigned IP address when starting the listener process rather than the address supplied by the client.

## 73.3 Using the Trace Log

Trace mode is an extremely useful debugging tool. This mode is activated by starting the VIM with the /trace command line parameter. When activated, a new menu item

appears on the system menu called Show Trace Log. Checking this menu item reveals the Trace Log Viewer (see below). The viewer can be used to examine a multitude of activities such as remote procedure calls (synchronous and asynchronous), events, context changes, status messages, as well as custom entries created by running components (see *Creating Trace Log Entries*).

The Trace Log is divided into four panes: the trace list, the trace detail, the toolbar, and the status bar. It has the following layout:

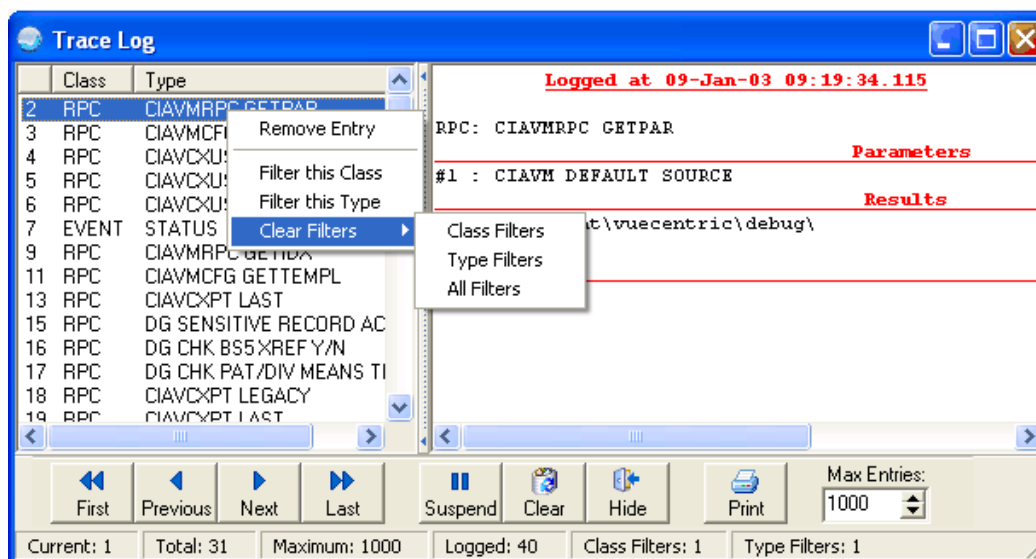
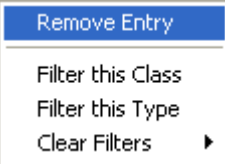
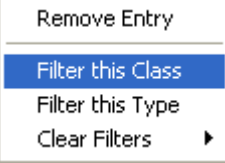
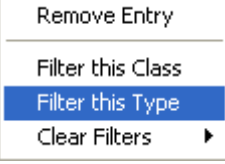
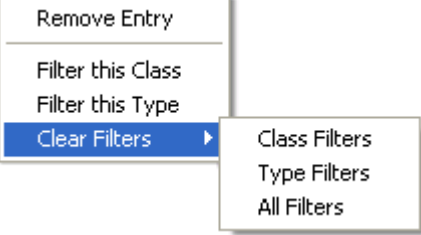
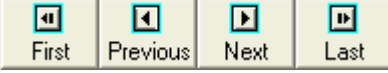







Figure 73-2: Sample Trace Log

The trace list is located at the upper left. If this pane is not visible, click the minimize bar that separates this from the trace detail pane. The trace list shows a list of all activities logged by the application. Each entry consists of three columns. From left to right these are the *sequence number*, which reflects the chronology of log entries and is incremented as each new entry is logged, the *class*, which represents the category of the entry (e.g., remote procedure call vs. host event), and the *type*, which provides further subclassification of the entry within its class. The trace list can be sorted by any column by clicking on the column header. Clicking on the same column header in succession toggles the sort order of that column. Right-clicking on an entry elicits a popup menu that permits further manipulation of the event list.

The upper right pane represents detailed information about the currently selected event. The toolbar permits navigation of the event list and controls other aspects of event logging. The status bar at the bottom of the dialog displays information about the state of the event log.

The following table describes the actions that can be elicited from either the popup menu or the toolbar:

	Removes the selected entry from the event log.
	Removes from the event log all entries belonging to the same class as the selected entry and suppresses logging of future events of the same class.
	Removes from the event log all entries belonging to the same type as the selected entry and suppresses logging of future events of the same type. This affects only event types within the currently selected event class.
	Removes existing filters. Choose the type of filter to remove, or all filters.
	Navigates the event log.
	Toggles between suspending and resuming event capture.
	Clears the event log.
	Hides the event log dialog. This does not affect the capture of event data.
	Prints the contents of the event detail pane.
	Sets the maximum number of event log entries. When this limit is reached, the oldest entries are removed.

## 73.4 About Component Support Services

The Component Support Services (CSS) provide the following types of services to the component author:

- Remote Procedure Support (synchronous and asynchronous)
- Event Management
- Context Management
- Dynamic Discovery

- Miscellaneous Utilities

Accessing these services requires acquiring an interface reference to the session object. The session object is shared across all components within the same application process space. It cannot be instantiated directly, but instead must be acquired from the server object. The sole purpose of the server object is to create a single instance of the session object and return its reference to the caller. Thus, accessing the session object involves two steps: instantiate the server object and request of it a reference to the session object. Once this is done, the server object is no longer needed and can be released.

The following are examples of how this process is done in Delphi and Visual Basic:

Delphi	Visual Basic
<pre>uses CIA_CSS_TLB; ... var   vcSession: ICSS_Session; begin   vcSession := CoCSS_Server.Create.Session; end;</pre>	<pre>Dim vcServer As New CIA_CSS.CSS_Server Dim vcSession As CIA_CSS.CSS_Session Set vcSession = vcServer.Session Set vcServer = Nothing</pre>

Typically, a programmer will acquire the reference to the session object in the component's initializer (the Initialize method in both Delphi and Visual Basic), storing it in an instance variable that persists for the life of the component. This avoids the overhead of repeatedly creating the server object each time the session object needs to be accessed.

A detailed description of the services provided by the CSS and how to use them can be found in the sections that follow.

## 73.5 About COM and ActiveX

The Component Object Model, or COM, is a Microsoft specification for ensuring interoperability between components regardless of the programming language or development tool used in their creation. Key to this specification is the concept of interfaces. An interface is a collection of method (i.e., procedures and functions) and property declarations. More than this, an interface represents an immutable contract - a guarantee that any object supporting a particular interface will always implement each and every method or property declared within it. While the implementations can differ, the means for invoking them are identical. COM builds upon the concept of interfaces by providing a standardized means of invoking interfaces that works within and across process boundaries and adjusts for internal differences in calling conventions and data representation. COM further standardizes on an established set of supported core datatypes (though these can be extended using custom marshaling techniques - a practice that is rarely necessary and very complex).

Also central to the success of COM is the concept of self-description, or the ability to interrogate a component's metadata to determine the interfaces it supports and their

declarations. This metadata is provided in the form of a type library. A type library can be supplied separately from the component's executable file, but more typically it is imbedded in that file as a resource. Visual Basic automatically generates type library information for its components based on the program code itself and has no provision for directly accessing the type library itself. Delphi provides a type library editor that can be used to modify an existing type library, as shown below:

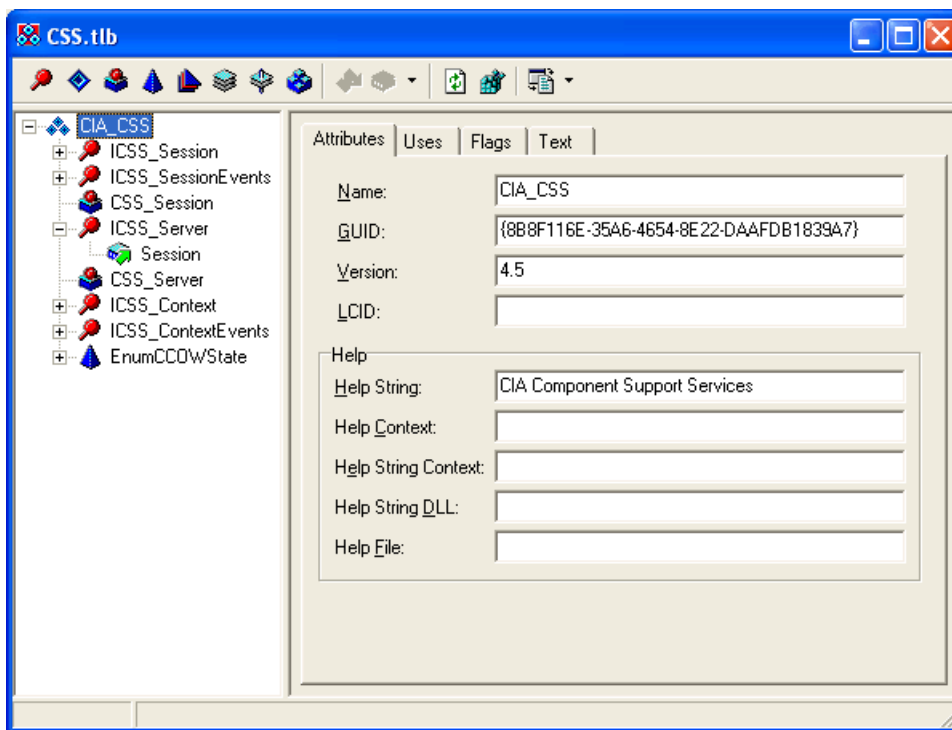


Figure 73-3: Sample Editor

ActiveX, also a Microsoft specification, builds upon the COM specification by defining a specific set of interfaces designed to support the hosting of visual components within container applications. A COM component that supports these interfaces is known as an ActiveX component. In Delphi, wizards exist to create ActiveX components either as wrappers for existing components, or as Active Forms which behave like regular forms in that other components can be placed upon them. Visual Basic has the equivalent of the Active Form in the User Control class. Again, because ActiveX controls are COM objects, it does not matter to the application hosting them what programming language or tool created them.

All components within the VueCentric<sup>®</sup> Framework are COM objects. This includes the VIM, CSS, CMS and all plug-in services and visual components. Further, the visual components are ActiveX components. This means that any visual object created for the VueCentric<sup>®</sup> Framework can be hosted in any ActiveX-compliant container. The VIM is one such container. Internet Explorer is an example of another.



## 73.6 Component Types

The VueCentric<sup>®</sup> Framework recognizes two basic types of components: **visual components** (ActiveX) and **services** (COM). Visual components can be manipulated in the VIM Designer (services cannot). Services extend the capabilities of the framework and although they can manifest themselves visually, they do not require a container (unlike visual components). The techniques for creating these two component types differ somewhat as does the manner in which they are loaded and executed. However, once created, the coding is very similar for both types. In fact, it is possible to create a visual component and a service within one executable file.

The sections that follow first address programming techniques common to both component types. Following that are sections that describe the procedures specific to a particular component type.

## 73.7 Component Registration

Component registration has three separate meanings, depending upon the context. They are:

### 73.7.1 COM Registration

In the COM model, all components must be registered before they can be used. In this sense, information about the component's interfaces and capabilities is stored in the Windows Registry under the HKEY\_CLASSES\_ROOT key. This registration is typically done by the component itself through a standard published entry point (DllRegisterServer). For Visual Basic and Delphi programmers, this code is automatically generated. When a component is requested from the CSS that has not been previously registered on the target machine, it is retrieved and registered automatically. A tool is also available in Windows called regsvr32.exe (usually located in the system directory) that can be used to manually register and unregister components. Running this tool with no command line options displays information on how it is used. The VueCentric System Management Utility can also be used to register and unregister components. See the online documentation that accompanies this utility for more information.

### 73.7.2 Framework Registration

Before a component can be utilized within the VueCentric<sup>®</sup> Framework, it must also be registered to the framework. This registration information is stored on the remote host in the VUECENTRIC OBJECT REGISTRY file. This can be done using the VueCentric System Management Utility, or it can be directly entered into the file using VA FileMan. This information determines what objects are available, how they are classified, who can use them, where to get them, and what their capabilities and special requirements are.

### 73.7.3 Runtime Registration

At execution time under the VueCentric<sup>®</sup> Framework, component registration has yet another meaning. In this context, registration means that the component advertises its presence to the CSS at runtime. The CSS maintains an internal table of all components so registered. It also interrogates the component to determine what context events it handles and connects any handlers it finds to the appropriate context objects. This registration process is automatic if the component is loaded by the VIM (unless its object registry settings suppress this). An object can also register itself by calling the RegisterObject method, passing a reference to itself. If an object registers itself in this manner, it must also unregister itself prior to unloading by calling the UnregisterObject method. In general, self-registration is only necessary if the object is to be used outside the VIM (e.g., in Internet Explorer). For more information about this topic, see the section on *Other Containers*.

While the latter form of registration is not required to use an object within the framework, unregistered objects cannot be discovered by other objects nor will they be connected to any context events.

## 73.8 Naming Conventions

Every COM object has a globally unique identifier (GUID) associated with it and each of its interfaces. While an object can be, and often is, referenced by its GUID, there is a second identifier that is more user friendly: the programmatic identifier (PROGID). The PROGID typically consists of two parts, separated by a period (e.g., “Word.Application”). Occasionally, a third part consisting of a version number is added (the so-called version-dependent PROGID, e.g., “Word.Application.8”). There are no fixed conventions for these identifiers and it should be readily apparent that the risk of naming conflicts is real. In the event that there is a naming conflict, the last object registered is the one referenced. Therefore, some naming convention is needed to minimize this risk.

For Delphi and Visual Basic environments, the PROGID is generated automatically, formed from the type library name (which initially comes from the project name) and the component name. In Delphi, both can later be changed using the type library editor. Refer to COM and ActiveX in [“Component Registration” on page 478](#)). Thus, the selection of a PROGID is contingent upon your selection of a project name and a component name. We advise beginning your project name with a namespace prefix. We have reserved the prefixes ‘CIA’, ‘VC’, and ‘CW’ for our projects. These prefixes refer to framework components, plug-in components, and legacy components, respectively. For example, CIA\_VIM.VIM refers to the automation interface for the VIM and vcNotifications.vcNotificationsX refers to the visual notifications component.

PROGIDs are mapped to their GUID equivalents in the Windows Registry under the HKEY\_CLASS\_ROOT node. For example, if you were to examine the registry key HKEY\_CLASSES\_ROOT\Word.Application\CLSID, you would find that the default

value associated with that key is the GUID corresponding to the object. Knowing this structure makes it easy to understand how two objects with the same PROGID cannot happily coexist on the same machine, even though their GUIDs differ. Remember: it is much harder to change a PROGID after deployment than to give careful thought to naming at the beginning of a project.

## 73.9 Multiple vs. Single Instancing

Multiple instancing refers to the ability to create more than one copy of a component within the same application space at the same time. For some components, this can clearly not be desirable for a couple of reasons. First, some components that perform a very central function should be limited to a single instance. For example, one would not want to have more than one order-entry component as this would be very confusing to the user. On the other hand, a component that provides a read-only view of the problem list might be useful in several different locations within the user interface. Second, some components are programmatically designed to be singletons (single instance). This would be true if, for example, a component required exclusive use of a shared resource, such as a database file. In Delphi, the use of global variables (i.e., variables declared outside the scope of a class or method) can cause serious problems because these are shared across multiple instances of the component. A good example of this are form declarations. The Delphi form wizard automatically generates a global variable to hold the form instance (this is not the case for Active Forms, but does hold true for any additional forms created within an Active Form project). If one instance of a component creates a form and saves the reference in the global form variable, and a second instance does the same, the first instance loses its reference and now refers to the form created by the second. This can result in some very odd behaviors. On the other hand, this can be used to advantage if, for example, the programmer wants to create a shared instance of a form to be used among all instances of the component. Visual Basic, on the other hand, declares all of its variables (even its “global” variables) as instance variables. This eliminates the problem (but also the advantage) of shared variables.

To control whether more than one instance of a component is allowed, set the “Allow Multiple Instances” field in the VUECENTRIC OBJECT REGISTRY file accordingly. If you are uncertain as to a component’s suitability for multiple instancing, disallow multiple instances.

## 73.10 Remote Procedure Calls

Remote procedure calls are conceptually very simple. Essentially, they permit calling a procedure on a remote host as if it were local. The VueCentric<sup>®</sup> Framework encapsulates the Medsphere Remote Procedure Broker in its Communications Services Layer (CSL). This encapsulation permits all components within the application space to share a single instance of the broker. Interaction with the CSL occurs through a suite of API’s provided by the session object.

Remote procedure creation is very straightforward. First, create and configure the remote procedure on the remote host. Then write the client code to invoke the remote procedure. Remote procedures can be invoked synchronously, where the client pauses until the procedure completes and returns its data, or asynchronously, where control immediately returns to the client after the call and the client is notified when the call has completed through a callback. All of these techniques are described in the sections that follow.

### 73.10.1 Create the M Routine

First, you must create M routine that contains the code that is to be executed. The entry point must be parameterized, with the first parameter reserved for returning data to the caller. Each subsequent parameter corresponds to a parameter passed by the client application and can be a single scalar value or a local array of values. Up to forty parameters can be specified.

The following example illustrates code implementing a simple remote procedure that accepts a string and a count as input parameters and returns an array of that string duplicated the number of times specified by the count. The return data type is assumed to be a global array (see next section).

```

DATA = Return value (assumes global array type)
; VALUE = String to duplicate
; COUNT = Number of times to duplicate
ENTRY (DATA, VALUE, COUNT) ;
N X
F X=1:1:COUNT S @DATA@(X)=VALUE
Q

```

### 73.10.2 Create a Remote Procedure Definition

The next step is to create an entry for the remote procedure in the REMOTE PROCEDURE file. At a minimum, you must define the following fields:

Field	Length
NAME	The name of the remote procedure. This is the name that the client application will use to invoke the remote procedure. The name should be appropriately namespaced and descriptive of its purpose.
TAG	The line label of the entry point in the M code. In the above example, this would be "ENTRY."
ROUTINE	The name of the routine in which the remote procedure code resides.
RETURN VALUE TYPE	The type of data returned by the remote procedure. This can be one of the following: 1 = Single value, 2 = Local array, 3 = Word processing, 4 = Global array, 5 = Global instance, H = Host file
WORD WRAP ON	Affects only return types 3, 4, and H. If true, a carriage return character is appended to each value in the array or line in the host file. If false, no carriage return is appended.

Though not required, the INPUT PARAMETER multiple should be completed to document the expected parameters and their purpose. The DESCRIPTION field should likewise be completed to document the purpose of the remote procedure.

The RETURN VALUE TYPE merits further elaboration. The Broker daemon passes the return value parameter (the first parameter of the remote procedure entry point) by reference. Therefore, any change to the parameter is returned to the daemon and then to the caller. The simplest return type, single value, returns whatever value is set in the first parameter. Like the single value type, the global instance type returns a single value to the caller. In this case, the return value parameter must be set to a valid global reference. For the local array and word processing return types, all subscripted values are returned to the caller. The global array return type returns all subscripted values beneath a root node specified in the return value parameter. The daemon sets this to a default value prior to calling the remote procedure. The remote procedure can use this value, or assign a different one. **Note that the daemon deletes this global root upon completion.** For all return types that are arrays (global and local), the daemon returns the data to the caller in the collation order of the subscripts (the subscripts themselves are not returned, only the data).

### 73.10.3 Register the Remote Procedure

Every remote procedure is invoked within an execution context (see the RPCContext property). This context dictates which remote procedures can be executed and rejects any attempts to execute a remote procedure outside the context. Execution contexts are simply entries in the OPTION file of type Broker. To register a remote procedure to an execution context, select the desired entry in the OPTION file (or create a new one with the TYPE field set to Broker), and add the remote procedure to the RPC multiple.

### 73.10.4 Calling a Remote Procedure

Once the remote procedure has been created and properly registered on the remote host, it can be invoked by the client application using one of a number of method calls provided by the session object. All methods for calling remote procedures begin with “CallRPC” and can be divided into two groups: synchronous and asynchronous, reflecting the two modes in which a remote procedure can be invoked. Synchronous calls do not return until the remote procedure has completed and returned its data. Asynchronous calls return immediately, notifying the caller at a later time when the remote procedure has completed. Choosing which mode to use depends upon a number of factors:

Requirement	Use
User interaction requires immediate results before application can continue.	Synchronous
Availability of data is not time critical.	Asynchronous
Remote procedure requires considerable time to complete.	Asynchronous
Series of sequenced transactions.	Synchronous

## 73.10.5 Synchronous Calls

### 73.10.5.1 RPC Methods

Calling a remote procedure in synchronous mode is very straightforward. There are six method calls that support this mode. They differ only in the type of return data that is expected. They are:

Method Name	Return Type
CallRPCBool	Boolean value
CallRPCDate	Date (Delphi: TDateTime; Visual Basic: Date)
CallRPCInt	32-bit integer
CallRPCList	Multi-valued string list in “comma-text” format. This format encloses each entry in double quotes and separates them with commas.
CallRPCString	String (Delphi: WideString; Visual Basic: BSTR)
CallRPCText	Multi-valued string list in “plain-text” format. This format separates each entry with a <CR><LF> pair.

If the remote procedure return type is multi-valued (either local or global array), you must use either CallRPCList or CallRPCText to retrieve all of the list elements returned. Otherwise, only the first element is returned. The choice between “comma-text” and “plain-text” formats is somewhat arbitrary. If list elements could contain either a <CR> or <LF> character, comma-text format is preferred. Delphi’s TStrings class provides a means to convert either of these formats into string lists (the CommaText and Text properties, respectively). Visual Basic requires parsing the result values into a string array.

Regardless of which of the above methods is used, the parameters passed are identical:

Parameter	Datatype	Description
RPCName	String	Name of the remote procedure to be invoked. If an execution context other than the default is desired, precede the RPC name with a context name and the ‘^’ delimiter. To specify a version number, prefix the RPC name with the version number enclosed in vertical bars.
Parameters	Variant	Parameters to be passed to the remote procedure.

### 73.10.5.2 Specifying the Remote Procedure

The RPCName property specifies the name of the remote procedure to invoke. If the remote procedure needs to be invoked in an execution context other than the default, precede the remote procedure name with the context name and a ‘^’ character. If the remote procedure requires a version number, you can include this by prefixing the version number enclosed in vertical bars (‘|’) to the remote procedure name. Consider the following examples:

RPCName Value	Description
CIAV GET PATIENT	Executes the remote procedure named ‘CIAV GET PATIENT’ in the default framework execution context. No version information is passed.
MYCONTEXT^MYRPC	Executes the remote procedure named ‘MYRPC’ in the execution context named ‘MYCONTEXT’. No version information is passed

RPCName Value	Description
1.5 MYRPC	Executes the remote procedure named 'MYRPC' in the default execution context. The XWBAPVER variable will be set to 1.5.
MYCONTEXT^ 2.1 MYRPC	Executes the remote procedure named 'MYRPC' in the execution context named 'MYCONTEXT'. The XWBAPVER variable will be set to 2.1.

### 73.10.5.3 Specifying the Parameter List

The Parameters property specifies the parameter values to be passed to the remote procedure. These correspond to the second and subsequent parameters of the remote procedure's entry point (recall that the first parameter is reserved for the return value). Because parameter lists typically differ for each remote procedure, the datatype of the Parameters property must accommodate these variations. As a consequence, the datatype of the Parameters property is a variant. The following table describes the techniques for packaging scalar (non-array) parameters in the Parameters property:

# of Parameters	Format
0	Set Parameters to NULL (Nothing).
1	Set Parameters to the value of the single parameter.
>1	Set Parameters to an array of variants. Set each element of the array to the value of the corresponding parameter.

### 73.10.5.4 Specifying Array Parameters

Passing array parameters to your remote procedure requires some additional effort. An array parameter must be passed as an array of variants. If any parameter to be passed is an array type, the Parameters property must be set to an array of variants even if the array parameter is the only parameter. Otherwise, the CSS would not be able to distinguish a parameter list from a single array parameter. Thus, to pass an array parameter, two arrays of variants must be created: one for the parameter list (we'll call it the parameter list array) and one for the array parameter (we'll call it the array parameter array). Set the reference to the array parameter array into the corresponding entry of the parameter list array. Set each entry of the array parameter array to the values to be passed. By default, array parameters are passed to the remote host with the same subscript values as the original. To override the default subscript value for an entry, prefix the entry with the subscript value (in comma-text format) followed by a character whose ASCII value is 1.

If passing parameters, especially array parameters, sounds confusing, it can be. This process can be greatly simplified by writing helper functions that take parameters in a format native to the programming language used and packages them in the manner described above. CIA has developed a number of Delphi and Visual Basic helper functions for this purpose. See Delphi Helper Functions elsewhere in this document.

### 73.10.5.5 Handling Exceptions

If the remote procedure raises an unhandled exception on the remote host, that exception is trapped and raised on the client. Therefore, the caller should be prepared to handle any exceptions that can be raised during the remote procedure invocation.

### 73.10.6 Asynchronous Calls

Calling a remote procedure in asynchronous mode is not altogether different from doing so in synchronous mode. Naming the remote procedure and packaging the parameter list are identical. However, there is only one method call for invoking a remote procedure in asynchronous mode and its return value is a handle that uniquely identifies the call. So how does the caller know when the remote procedure has completed and how is its data returned? The CSS provides a declaration for a callback interface (ICSS\_SessionEvents) that is used to signal all asynchronous activities (including events and asynchronous RPC's). The caller must implement this interface and provide implementations for each of its methods. When the caller invokes a remote procedure asynchronously, it must pass a reference to this callback interface. The session object then invokes methods (callbacks) on this interface to notify the caller when an asynchronous activity has completed.

#### 73.10.6.1 Calling the Remote Procedure Asynchronously

The Session object provides one method, `CallRPCAsync`, for calling a remote procedure asynchronously. It takes the following parameters:

Parameter	Datatype	Description
RPCName	String	Name of the remote procedure to be invoked. If an execution context other than the default is desired, precede the RPC name with a context name and the '^' delimiter. To specify a version number, prefix the RPC name with the version number enclosed in vertical bars.
Parameters	Variant	Parameters to be passed to the remote procedure.
Callback	ICSS_SessionEvents	Callback interface to be invoked on completion of the remote procedure.
PlainText	Boolean	If true, data returned to the callback interface is in plain-text format. Otherwise, format is in comma-text format.
<return value>	Integer	A 32-bit handle that uniquely identifies this asynchronous call.

Note that the first two parameters are identical to those used by the synchronous remote procedure methods. The Callback parameter refers to the callback interface the Session object will use to notify the caller when the remote procedure has completed. PlainText indicates whether the data is to be returned in plain-text or comma-text format. Data returned from asynchronous calls are always formatted as lists in one of these two formats. This is not to imply that only remote procedures that return lists can be called asynchronously. Indeed, a remote procedure that returns, for example, a single Boolean value can be called asynchronously (in fact, any remote procedure can be called asynchronously). In the case of a remote procedure returning a single Boolean value, the return value would be a list containing a single element and the caller must convert that element from a string to the desired format.



The CallRPCAsync method returns a 32-bit integer handle that uniquely identifies the call. Because a client can have multiple outstanding asynchronous calls, this handle is critical to identify which call is being signaled.

### 73.10.6.2 Implementing the Callback Interface

The ICSS\_SessionEvents interface has the following declaration:

Delphi
<pre> procedure RPCCallback(Handle: Integer; const Data: WideString); safecall; procedure RPCCallbackError(Handle: Integer; ErrorCode: Integer; const ErrorText: WideString); safecall; procedure EventCallback(const EventType: WideString; const EventStub: WideString); safecall; </pre>
Visual Basic
<pre> Sub ICSS_SessionEvents_RPCCallback(ByVal Handle As Long, ByVal Data As String) Sub ICSS_SessionEvents_RPCCallbackError(ByVal Handle As Long, ByVal ErrorCode As Long, ByVal ErrorText As String) Sub ICSS_SessionEvents_EventCallback(ByVal EventType As String, ByVal EventStub As String) </pre>

The first two methods in the ICSS\_SessionEvents interface provide support for asynchronous remote procedure calls. The third is used for reporting events and is discussed in the section on *events*. Note that the client must supply implementations for all three methods, even if only one or two are actually used. This is a requirement for implementing any COM interface. If a method is not needed, its implementation can be left empty (a “NOP” implementation).

The Session object invokes the caller’s RPCCallback method when an asynchronous remote procedure has completed successfully. The Handle parameter is the same value returned by the CallRPCAsync method. The Data parameter is the data returned by the remote procedure in either plain-text or comma-text format.

If an asynchronous remote procedure raises an unhandled exception on the remote host, that exception is trapped and the RPCCallbackError method is called instead. This method includes the Handle parameter as expected, but also supplies ErrorCode and ErrorText parameters that provide information about the exception.

### 73.10.6.3 Aborting a Pending Call

Occasionally, the caller can wish to abort an asynchronous call that has not yet completed. This might occur, for example, if a patient context change has occurred and the pending remote procedure no longer applies. To abort an asynchronous remote procedure, call the CallRPCAbort method, passing the handle of the call that is to be aborted. If the remote procedure has already completed, the request is ignored. If the remote procedure is pending execution, it is removed from the execution queue. If the remote procedure is in progress, it is requested to abort. Even if the remote procedure ignores the abort request, any data it can return is discarded and its completion is not signaled to the caller.

## 73.11 Context Management

The VueCentric<sup>®</sup> Framework provides support for special services known as context objects. A context object is used to establish a common context for a specific entity, such as patient or user. For context objects to be useful, a mechanism must exist to notify interested parties when the context is about to change. The Framework accomplishes this through the use of a callback interface.

Every context object defines a callback interface that is a descendant of the ICSS\_ContextEvents interface. When a context object registers itself as a service with the CSS, the CSS recognizes that it is a context object and enters its callback interface into an internal table. When a component (visual or non-visual) registers itself with the CSS, that component is interrogated to determine if it implements any of the callback interfaces in this table. If it does, the component is registered as a subscriber to the callback. This means that all a component must do to be notified of context changes for a particular context object is to implement the callback interface for that context object. The CSS takes care of the rest.

### 73.11.1 Callbacks

Every context object defines a callback interface that is a descendant of the ICSS\_ContextEvents interface. This interface has the following declaration:

Delphi
<pre>function Pending(Silent: WordBool): WideString; safecall; procedure Canceled; safecall; procedure Committed; safecall;</pre>
Visual Basic
<pre>Function ICSS_ContextEvents_Pending(ByVal Silent As Boolean) As String Sub ICSS_ContextEvents_Canceled() Sub ICSS_ContextEvents_Committed()</pre>

Context change is a democratic process and occurs in two phases. In the first phase, known as the polling phase, every subscriber to the context change is interrogated for its acceptance of the change. This is done through a call to the Pending method. Subscribers indicate their acquiescence to the proposed change by returning a null value. By returning a non-null value, the subscriber is indicating that it does not wish the context change to occur. The Silent parameter indicates whether or not the subscriber is permitted to interact with the user during this decision-making process. If Silent is true, no user interaction is permitted. This will be the case under one of two conditions: the context change was initiated by an external application through the CCOW interface, or the context change was initiated during a forced shutdown of the application.

If all subscribers acquiesce to the proposed context change, the context is changed and each subscriber is notified of the change through a call to the Committed method. If any subscriber rejects the context change, the proposed change is aborted and

subscribers are notified through the Canceled method. Note that in the case where a silent context change is occurring, the context change can occur even when a subscriber rejects the change. This will always be the case in a forced shutdown and can also occur in the case of a CCOW-initiated context change if the requester decides to force the change despite the objection. Therefore, a component must be prepared to react to a context change even if it has rejected the request.

### 73.11.2 Requesting a Context Change

How a context change request is made is dependent upon the individual context object. For example, setting the Handle property of the patient context object initiates a context change request. Depending on whether or not the context change is accepted, the value of the Handle property can or cannot be changed. The Select method of the patient context object is another means for changing the patient context. Yet a third method, is through a call to the SETCTX^BEHOPTCX method on the remote host. Thus, there is no “standard” mechanism for initiating a context change request.

## 73.12 Events

Events are an extremely powerful tool for communicating information. The VueCentric<sup>®</sup> Framework implements events utilizing a subscribe/publish (consumer/producer) model. In this model, clients can subscribe to a particular event. When an event is generated (published), the Framework forwards that event to all subscribers of that type of event. Some important features of the VueCentric<sup>®</sup> event model are:

- Events are fully extensible. New event types can be defined at any time.
- Event notification is asynchronous. Subscribers can be notified at any time that an event has occurred.
- Sequence of event delivery depends upon order of event subscription. Generally one should not rely upon a particular sequence of event delivery.
- Event subscription and publication can be restricted using security keys.
- Event publication can be local (current process) or remote (all active sessions).
- Events are hierarchical. This allows the publishing of events at one level of detail and subscribing of events at another.
- Events can be generated by the client (local or remote) or by the remote host (remote only).
- Distribution of remote events can be restricted to a subset of subscribers (at user or session level).

### 73.12.1 Defining an Event

Events should be defined by creating an entry in the CIA EVENT TYPE file. While this is technically not necessary in order to use an event, it is strongly encouraged to do so for two reasons. First, this file provides a place to formally document each event

and helps avoid naming collisions between events. Second, this file permits applying business rules that can restrict publication and subscription rights.

### 73.12.2 Firing Events: Local vs. Remote

Events fired locally are distributed only to subscribers within the current session. Events fired remotely are sent to the host system which then distributes them to all subscribing sessions. A receiver of an event handles both types identically.

The Session object provides two methods for firing (signaling) events: `EventFireLocal` and `EventFireRemote`, for firing an event to local and remote subscribers, respectively. They are defined elsewhere in this document.

### 73.12.3 Receiving Events: Callbacks

When an event is fired, the Session object notifies each subscriber through the `EventCallback` method of the `ICSS_SessionEvents` callback interface. This callback interface is specified at the time the subscription is established in the call to the `EventSubscribe` method. This callback interface is the same `ICSS_SessionEvents` interface described in the section on asynchronous remote procedure calls. Any component wishing to subscribe to an event must implement this interface and provide implementations for all three methods declared within it. The `EventCallback` method has the following parameters:

## 0.0.1 Hierarchical Events

There are times when an event publisher might want to specify an event at one level of granularity, but an event subscriber might wish to subscribe at a less granular level. For example, the publisher of a NOTIFY event might want to specify the type of notification (e.g., critical lab value vs. order needing signature) as part of the event type. While one subscriber might want to know about only notifications of a particular type, another subscriber might want to know about all NOTIFY events. To reconcile differing event granularity requirements between producers and consumers, the Broker implements a hierarchical schema for event naming and subscription.

Hierarchical events are named using the following convention:

`<Event name>.<Event subtype>.<Event sub-subtype>.<...>`

The convention uses a period to separate each subtype specifier. Subtypes appear in order of increasing specificity, from left to right. A consumer can subscribe to a hierarchical event at any level of specificity. In the above example, the event producer can publish the following two events:

NOTIFY.CRITICALVALUE

NOTIFY.ORDERNEEDSSIG

A consumer caring only about critical value notifications can subscribe to the NOTIFY.CRITICALVALUE event while a second consumer wanting to know about all notification events can subscribe to NOTIFY events. Both subscribers would receive NOTIFY.CRITICALVALUE events, but only the second would receive NOTIFY.ORDERNEEDSSIG events.

When defining hierarchical events in the CIA EVENT TYPE file, one can create an entry for only the top level event type or for each subtype or both. Setting the DISABLE field to YES for an event also disables all events of greater specificity below it in the hierarchy. In the above example, setting the DISABLE field of the NOTIFY entry to YES would disable the NOTIFY.CRITICALVALUE and NOTIFY.ORDERNEEDSSIG events even if those events had separate entries in the file with their DISABLE field set to NO. This inheritance pattern also holds true for security keys associated with events. Thus, it is only necessary to create entries for event subtypes if one desires to apply business rules to those subtypes that differ from the parent entry.

## 73.13 Creating Visual Components with Delphi

Delphi offers two project types for creating visual components: Active Form and ActiveX Control. The latter type generates a simple wrapper for an existing control and cannot be used for creating more complex controls. Therefore, its use will not be addressed here. The Active Form, on the other hand, allows one to create a form as an ActiveX control. This project type is much more useful.

### 73.13.1 Creating the Active Form Project

To create an Active Form project, select File | New | Other... This will bring up the project selection dialog as shown below:

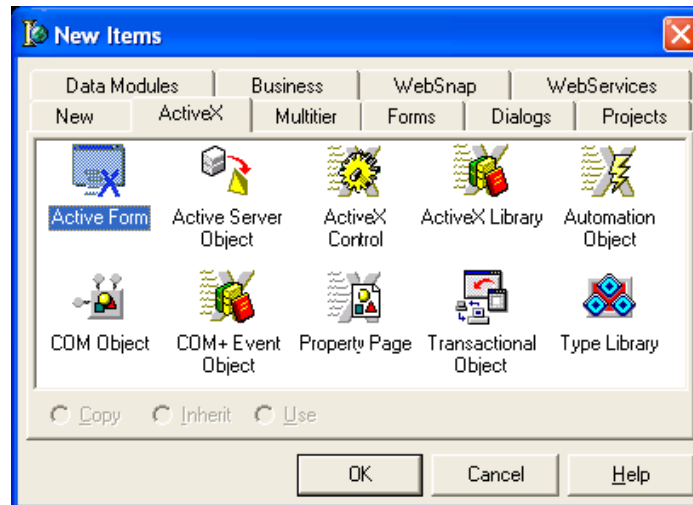


Figure 73-4: Sample Selection Dialog

Select the ActiveX tab, pick the Active Form project type, and click **OK**. You will then be presented with the Active Form Wizard:

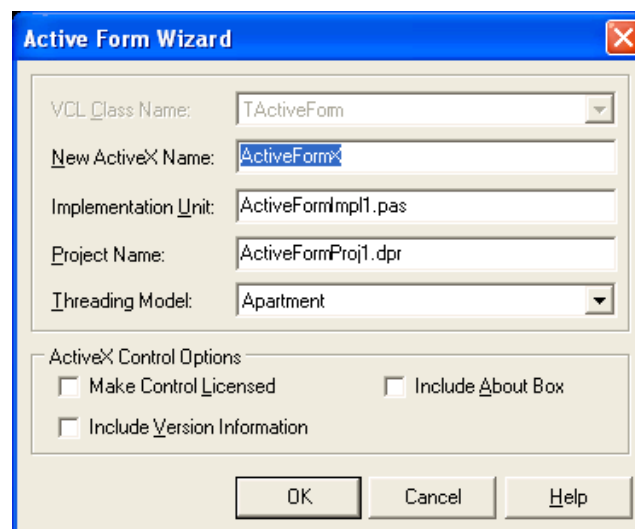


Figure 73-5: Sample Active Form Wizard

Change the first three editable entries to the desired names. Note that the New ActiveX Name will become the name of your component (and the second part of the programmatic identifier) and the Project Name will become the name of the type library (and the first part of the programmatic identifier). Changing these later can be done, but can be problematic, so choose wisely here. Leave the Threading Model as Apartment. Check the Include Version Information option. The other two options are at the component author's discretion, but we will leave them unchecked. After making these changes, the Active Form Wizard now looks like this:

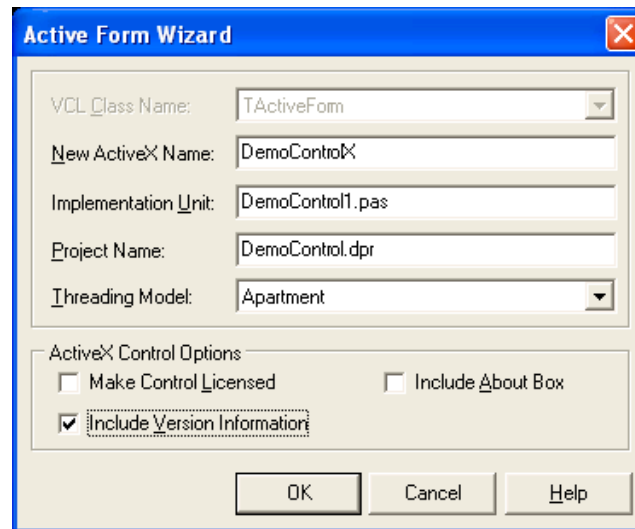


Figure 73-6: Sample Changes Made

Click the **OK** button and the wizard will create your project. The project will consist of a single form and its class declaration (class name TDemoControlX) and a type library.

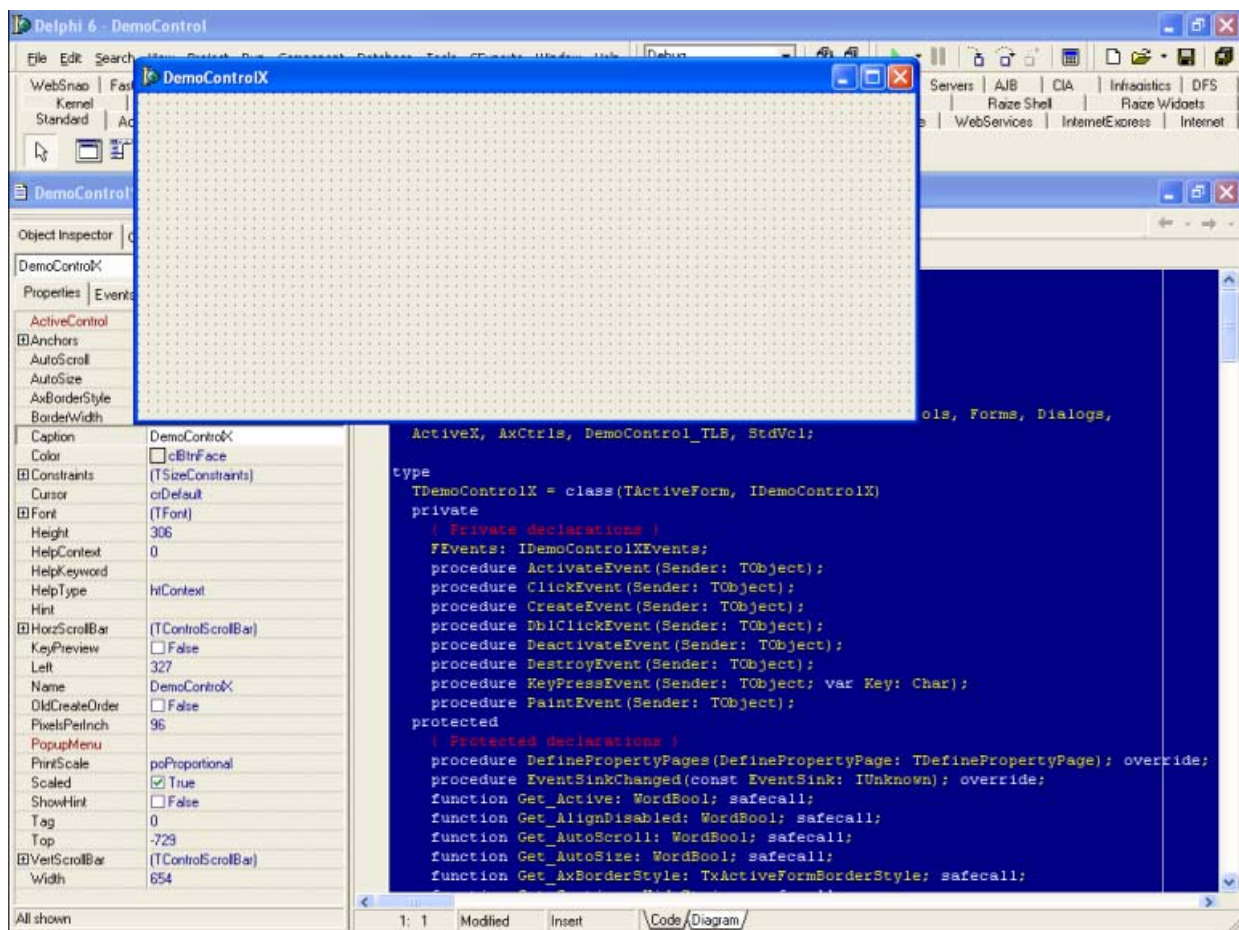


Figure 73-7: Created Project

Note that the component class declaration already has a number of event handlers and methods defined. These are standard entries created by the Active Form Wizard. While the declarations for these should not be modified, their implementations can be. Among the events of particular interest are the `DestroyEvent` and the `PaintEvent`, whose implementations can be modified to accomplish tasks that need to occur during object destruction and painting, respectively. For tasks that need to occur immediately following object creation, complete the implementation of the `Initialize` method that can be found in the public declarations for the Active Form class. Do not use the `CreateEvent` for this purpose as our experience has shown that this is not reliably invoked during object creation.

### 73.13.2 Designing the Form

Next, we are going to add a `TMemo` control and four buttons to the form. First, double-click the `TMemo` component on the component palette to add it to the form. Set its `Align` property to `alTop` and adjust its height as desired. Add four `TButton` components to the form using the same technique, and arrange them at the bottom of the form as shown below. Set the `Anchors` property of each to `[akLeft,akBottom]`.

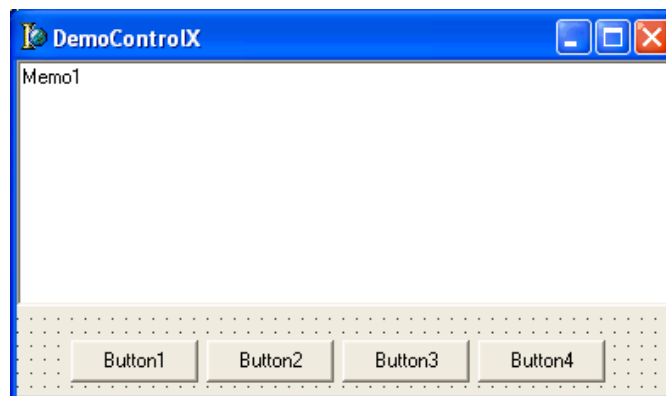


Figure 73-8: Adding Four Button Components

Now modify the `Caption` property of each button, from left to right, to **Sync RPC**, **Async RPC**, **Fire Event**, and **Clear**. Double click the right-most button that is now labeled **Clear** to generate a handler for its click event and complete it as shown below:

```
procedure TDemoControlX.Button4Click(Sender: TObject);  
begin  
    Memo1.Clear;  
end;
```

Figure 73-9: Modified Caption Property

### 73.13.3 Accessing the Session Object

Before we can perform a remote procedure call, we must obtain access to the `Session` object within the `CSS`. For Delphi to access any `COM` object, it must first generate a type declaration for the object's type library. This type declaration consists of an



automatically generated unit that contains code that is the Delphi equivalent of the type library's object and interface declarations. This allows the Delphi compiler to understand a type library declaration in its own native format. To do this, select **Project | Import Type Library...** from the Delphi menu. Select "CIAI Component Support Services (Version x)" from the list box. If there are multiple versions listed, select the highest version. If this entry does not exist, it means that CSS has never been registered on your machine (see the section on COM registration to see how this is done). Uncheck the box labeled **Generate Component Wrapper** as shown below:

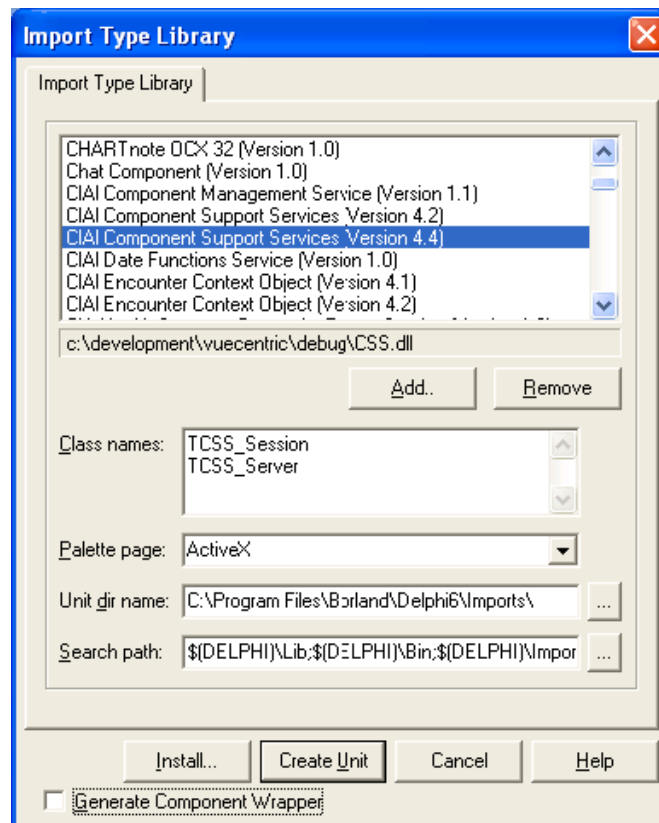


Figure 73-10: Unchecked General Component Wrapper Checkbox

Finally, click **Create Unit**. This will create a file called CIA\_CSS\_TLB.pas in the Imports folder under the Delphi installation directory. Note that you only need to do this procedure the first time you need to reference a COM object. The file created in this manner is then available to all projects. It also should be done whenever an object's type library changes to insure that Delphi recognizes the changes.

Now that you have a Delphi type declaration for the CSS, we can add code to access the Session object. Select the DemoControl1 unit in the code editor and add a reference to the CIA\_CSS\_TLB unit to the uses clause at the top as shown:

```
uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
ActiveX, AxCtrls, DemoControl_TLB, StdVcl, StdCtrls, CIA_CSS_TLB;
```

Figure 73-11: Added Reference

Next, add an instance variable to the private section of the TDemoControlX class declaration call FSession. This will be used to hold a reference to the session object.

```
private
( Private declarations )
FSession: ICSS Session;
FEvents: IDemoControlXEvents;
procedure ActivateEvent(Sender: TObject);
```

Figure 73-12: Added Instance Variable

Finally, add the following line of code to the Initialize method to obtain the Session object reference:

```
procedure TDemoControlX.Initialize;
begin
inherited Initialize;
OnActivate := ActivateEvent;
OnClick := ClickEvent;
OnCreate := CreateEvent;
OnDblClick := DblClickEvent;
OnDeactivate := DeactivateEvent;
OnDestroy := DestroyEvent;
OnKeyPress := KeyPressEvent;
OnPaint := PaintEvent;
FSession := CoCSS_Server.Create.Session;
end;
```

Figure 73-13: Added Line of Code

#### 73.13.4 Accessing the Patient Context Object

Before we can access the Patient Context object, its type library must be imported in the same manner as shown above. From the Import Type Library dialog, select the entry “CIAI Patient Context Object (Version x)” and click **Create Unit**. This will create the CSS\_Patient\_TLB.pas file which contains the Delphi declarations necessary for accessing the Patient Context object. Add a reference to this unit to the uses clause of the DemoControl1 unit:

```
uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
ActiveX, AxCtrls, DemoControl_TLB, StdVcl, StdCtrls, CIA_CSS_TLB, CSS_Patient_TLB;
```

Figure 73-14: Added Reference

Declare an instance variable to hold the reference:

```
private
  ( Private declarations )
  FPatient: ICSS_Patient;
  FSession: ICSS_Session;
  FEvents: IDemoControlXEvents;
```

Figure 73-15: Declared Instance Variable

Now, we need to obtain a reference to the Patient Context object. Because this object is just a special kind of service, we use the Session object to retrieve a reference to it. Insert the following line of code in the Initialize method to do this:

```
procedure TDemoControlX.Initialize;
begin
  inherited Initialize;
  OnActivate := ActivateEvent;
  OnClick := ClickEvent;
  OnCreate := CreateEvent;
  OnDblClick := DblClickEvent;
  OnDeactivate := DeactivateEvent;
  OnDestroy := DestroyEvent;
  OnKeyPress := KeyPressEvent;
  OnPaint := PaintEvent;
  FSession := CoCSS_Server.Create.Session;
  FPatient := FSession.FindServiceByCLSID(CLASS_Patient) as ICSS_Patient;
end;
```

Figure 73-16: Inserted Line of Code

This code will cause the Session to locate (and start if not already started) the indicated service, in this case the Patient Context object. Because this function returns a reference to the default IUnknown interface, it must be cast to the desired interface, in this case ICSS\_Patient.

### 73.13.5 Calling a Remote Procedure in Synchronous Mode

Now we are ready to add code that will invoke a remote procedure and return its data in the TMemo control. First, we will add a click handler to the first button (labeled **Sync RPC**) to execute a remote procedure that returns detailed information about the currently selected patient and populates the TMemo control with the results. Double-click the left-most button in the form designer and add the following code to its click event handler:

```
procedure TDemoControlX.Button1Click(Sender: TObject);
begin
  if FPatient.Handle = 0
  then Memo1.Lines.Text := 'No patient is currently selected'
  else Memo1.Lines.Text := FSession.CallRPCText('BEHOPTCX_PTINQ', FPatient.Handle);
end;
```

Figure 73-17: Added Code

This code calls the remote procedure named BEHOPTCX\_PTINQ, passing it a single parameter corresponding to the patient's internal entry number (DFN), and places the

return text in the TMemo control. If the patient context is empty, it displays a message to that effect instead.

### 73.13.6 Testing the Component

Before we proceed to add additional functionality, let us test what we currently have. To do this, we must first compile the project. Select **Project | Build DemoControl** from Delphi's menu. The project should compile without errors. Next we have to do the COM registration. Select **Run | Register ActiveX Server** from Delphi's menu to do this. You should receive confirmation of successful registration. Finally, we need to register our new component to the VueCentric® Framework. The easiest way to do this is to use the VueCentric System Management Utility. Run the tool and login to the remote host. Select the Object Registry tab and in the Restrict List To box, check **Local Registry** (this allows us to see local COM objects that aren't yet registered to the Framework). The list of objects will refresh. Now search the list for the programmatic identifier of our newly created component, DemoControl.DemoControlX, and select that entry. In the upper right pane, click **Copy** to copy the local COM registration information into the VueCentric Settings pane. In that pane, fill in the Name field with the display name for the control. The choice of name is arbitrary. We'll call it Demo Control. Finally fill in the height and width fields with 100 and 200, respectively. Now click **Apply** at the bottom. The display should look something like this:

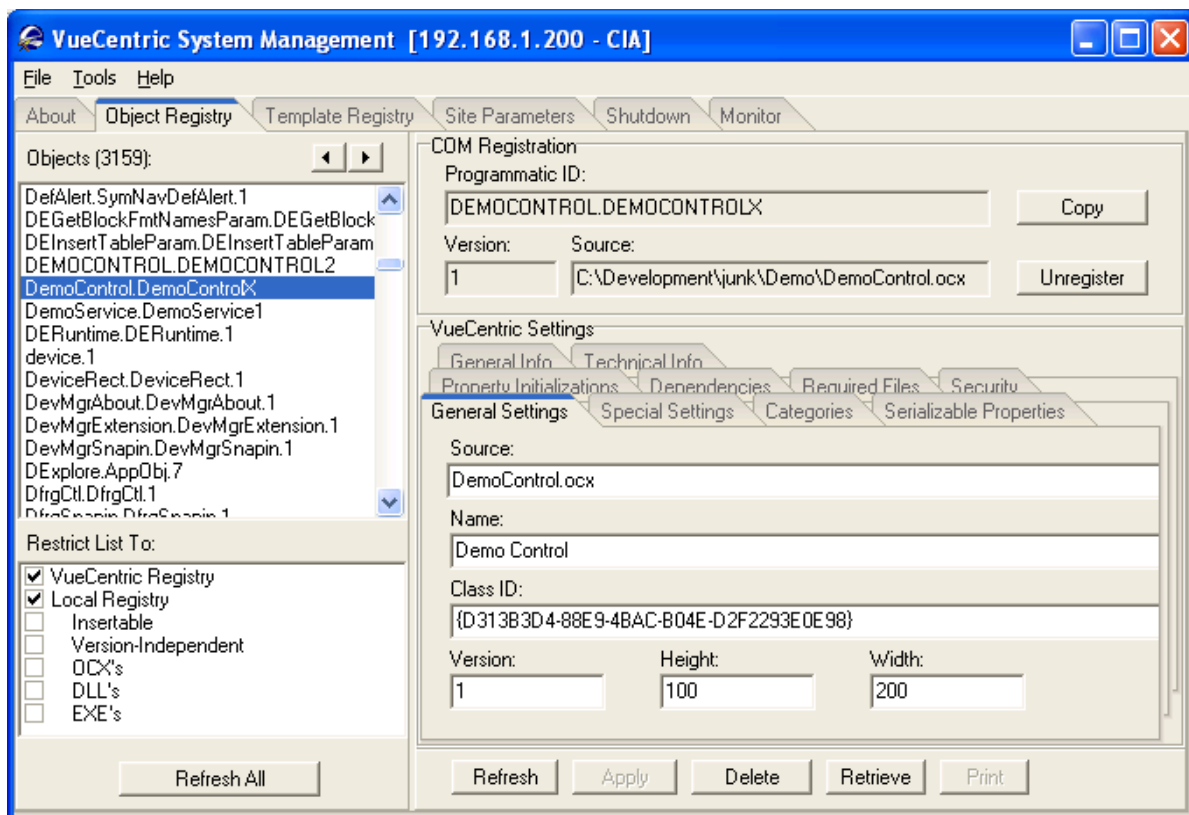


Figure 73-18: Sample Form

You can now close the utility. To test the component, start the Visual Interface Manager with the following flags:

```
vim.exe /noupdate /blank /trace
```

After logging in, enter design mode, right click the desktop and select **Add Object**. Expand the Name node and locate and add the Patient Identification Header. Top align this control (right-click on it in design mode to do this). Next find and add the Demo Control. Set the alignment of this control to all. Save this as a template named %DEMO for later retrieval. Now exit design mode and click on the **Sync RPC** button. You should see text in the memo control. Try clicking on the Patient Identification Header control and changing the patient selection. Note that the contents of the memo control is unaffected, because we have not yet subscribed to patient context change events. However, if we click the **Sync RPC** button again, the text that appears in the memo control now pertains to the newly selected patient.

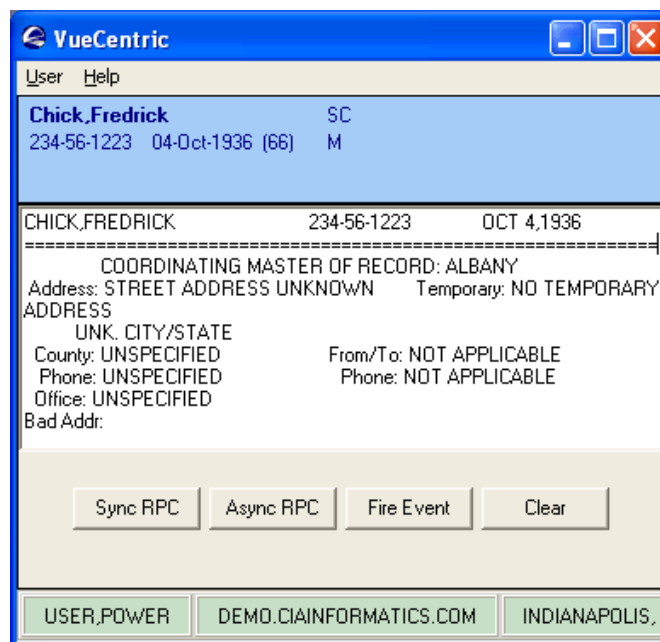


Figure 73-19: Sample of Text Pertaining to Selected Patient

Now close the application (note: if you do not close the application now, you will receive an error the next time you try to compile the project because the control's executable image is locked).

### 73.13.7 Subscribing to Patient Context Changes

As we noticed, our component retrieves information based on the currently selected patient, but it does not respond when the patient selection changes. Let us fix this deficiency by having our component subscribe to patient context changes and modify the memo control's contents when the context change occurs. To do this, we need to use the type library editor. To view the type library editor, choose the **View | Type**

Library menu option from Delphi's main menu. You should now see something similar to this:

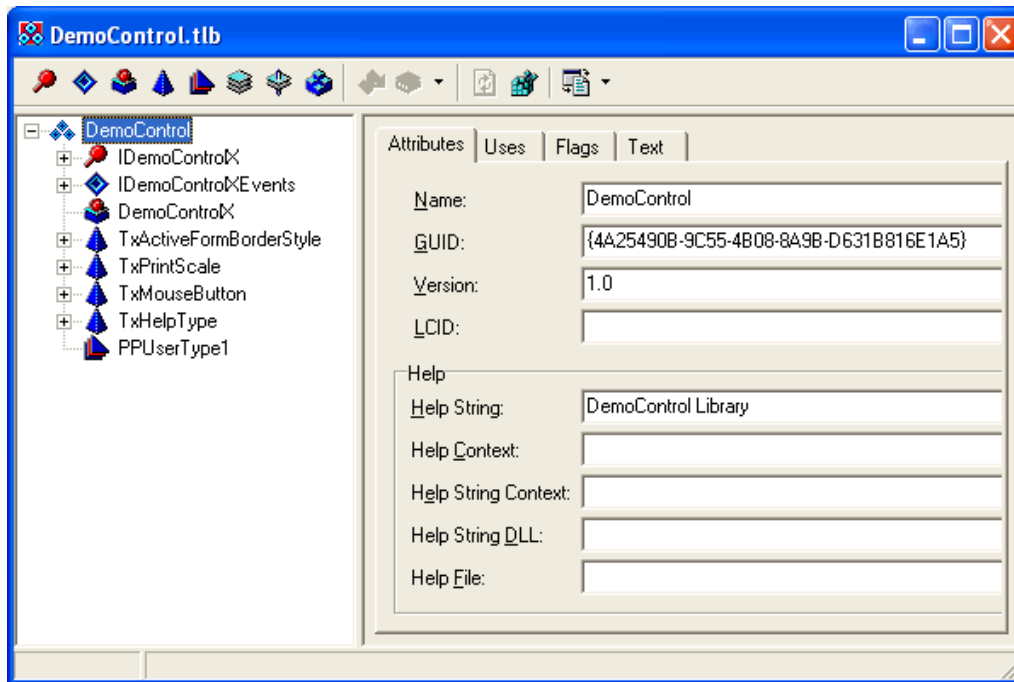


Figure 73-20: Sample Main Menu

Your GUID's will be different, but otherwise the dialog should look the same as this one. On the left, you see several entries. The IDemoControlX entry refers to the default interface for our control. Expanding that entry would reveal all of the properties and methods that are exposed on the COM interface. IDemoControlXEvents is the standard event interface for our component, which we will not make use of. DemoControlX is the actual component itself. To make a component respond to context changes, the component must implement the context change callback interface that is declared by the context object itself. To add this interface declaration to our component, we must first reference the context object's type library in our component's type library. To do this, select the top node labeled DemoControl. This node corresponds to our type library. When it is selected, several tabs appear on the right. Select the one labeled Uses. You should see the following:

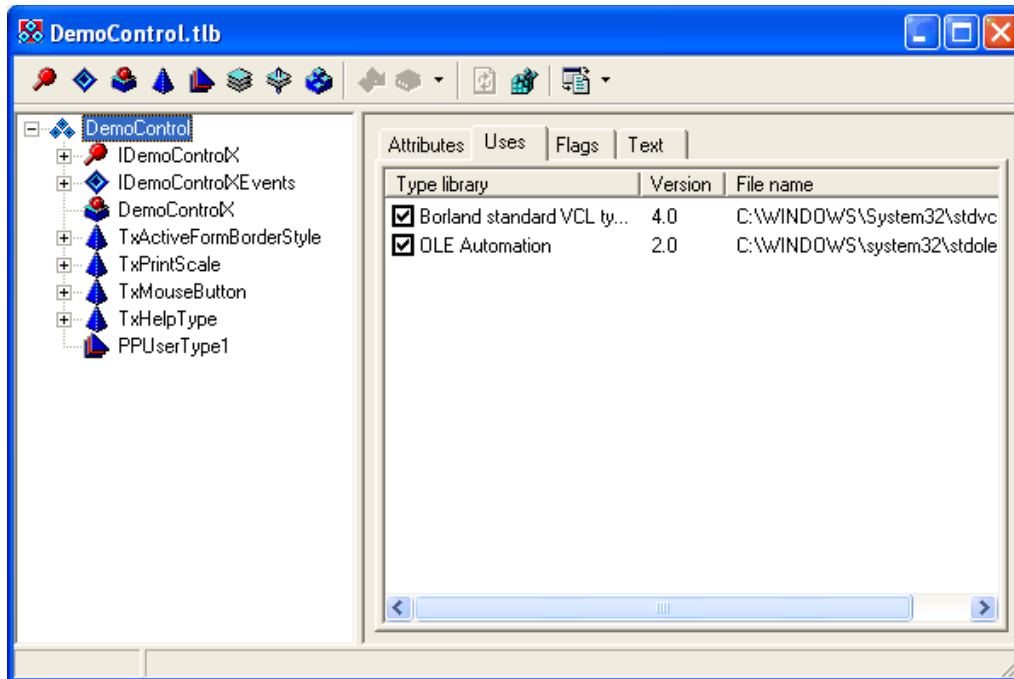


Figure 73-21: Selected Uses Tab

This view shows the type libraries that are currently referenced by this type library. To view all known type libraries, right-click on the right pane and select **Show All Type Libraries** from the popup menu. You will now see a much longer list. Scroll down until you find the Patient Context object and check that entry:

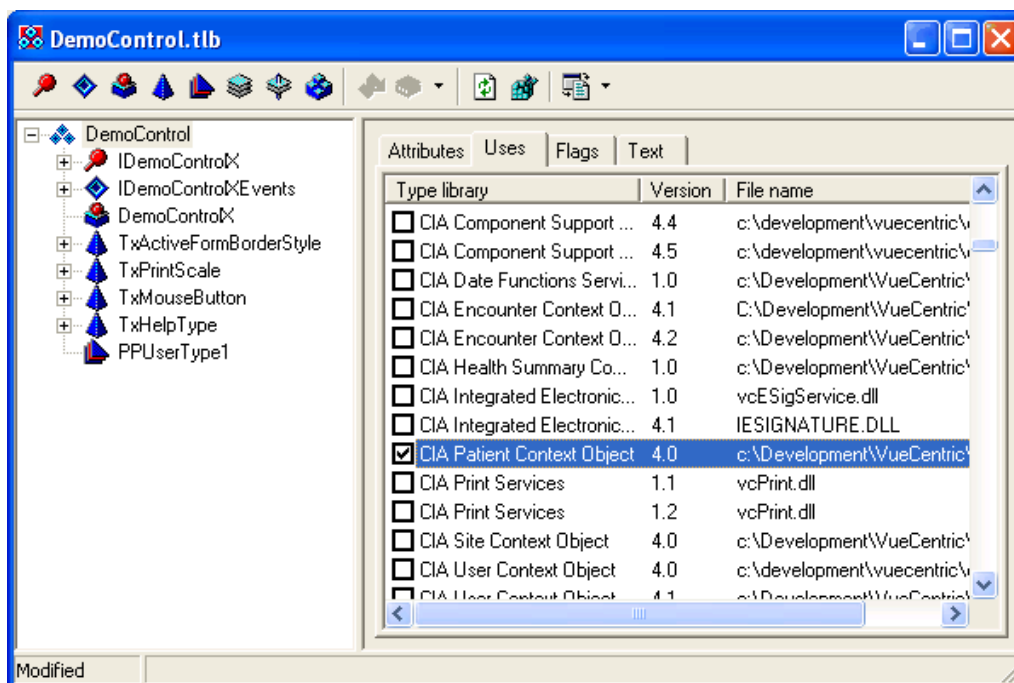


Figure 73-22: Uses Tab Showing All Type Libraries

Now we want to add the Patient Context object's context change interface to our component. To do this, select the DemoControlX node on the left and the tab labeled Implements on the right. Right-click on the right pane and choose **Insert Interface** from the popup menu. You will be presented with a list of known interfaces:

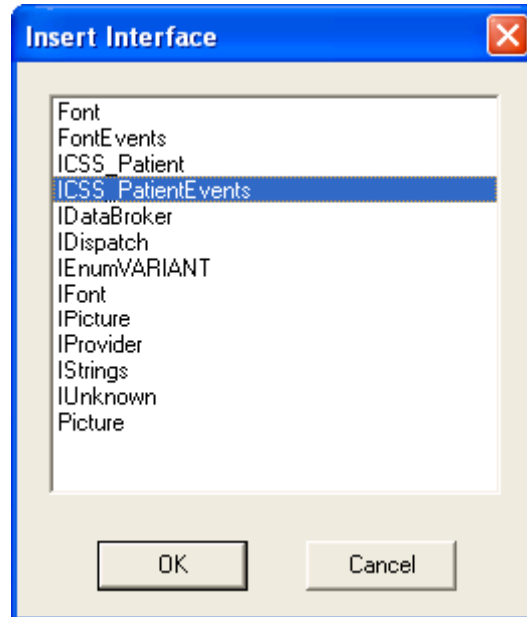


Figure 73-23: Sample Insert Interface Pop-up

Select the ICSS\_PatientEvents interface from this list and click **OK**. The type library editor should now look like this:

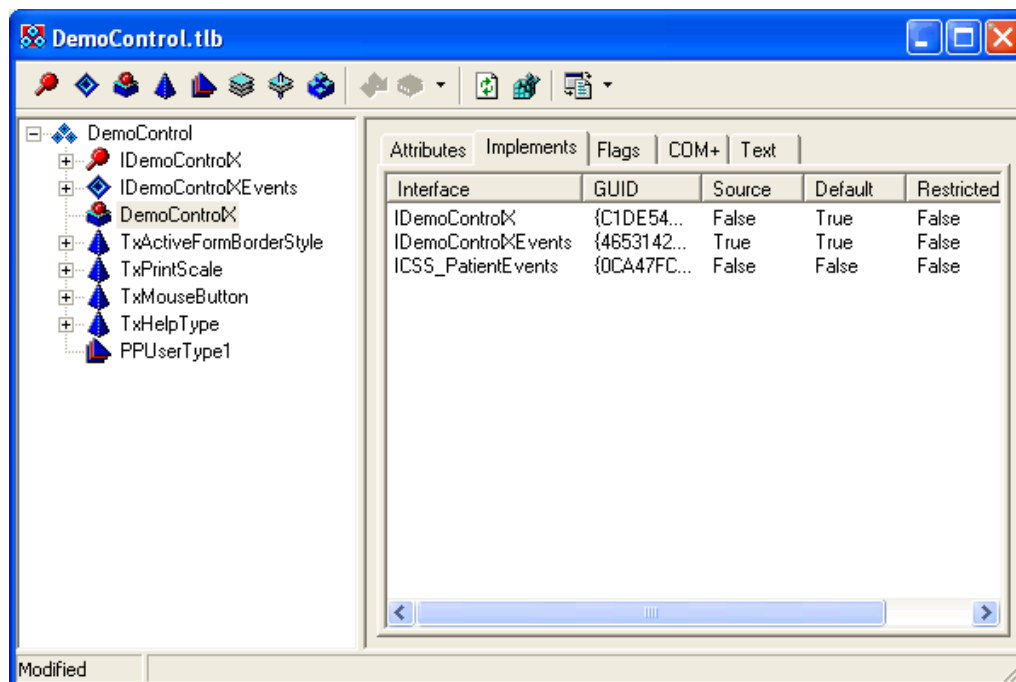



Figure 73-24: Selected ICSS\_PatientEvents Interface



Now we need to synchronize our program code with the changes we have made in the type library editor. To do this, click the  button in the toolbar to refresh your code. You should now see three new methods in your components class declaration and three empty implementations in the code section:

```
function TDemoControlX.Pending(Silent: WordBool): WideString;
begin

end;

procedure TDemoControlX.Canceled;
begin

end;

procedure TDemoControlX.Committed;
begin

end;
```

Figure 73-25: Sample Three Methods

We will add code to each of these implementations to populate the memo control with text indicating that each method has been invoked. Complete the implementations with the code shown below:

```
function TDemoControlX.Pending(Silent: WordBool): WideString;
begin
    Memo1.Clear;
end;

procedure TDemoControlX.Canceled;
begin
    Memo1.Lines.Text := 'Context Change Canceled.';
end;

procedure TDemoControlX.Committed;
begin
    Memo1.Lines.Text := 'Context Change Committed.';
end;
```

Figure 73-26: Added Code

When a context change is initiated, the Pending method will be called first. In this method, we simply clear the memo control's contents. Because we are not setting the return value for the Pending method, we are essentially voting YES to the context change. Assuming nothing else cancels the pending change, the Committed method is called next. In that method, we set the memo control's text to indicate that the context change was committed. Now compile the project by selecting **Project | Build DemoControl** from the menu. Because we made changes to the type library, we need to

re-register the control by selecting **Run | Register ActiveX Server**. Now restart the Visual Interface Manager, this time with the following command line options:

```
vim.exe /noupdate /trace /template=%DEMO
```

Once you login, you should see your component much as it looked before. Click the **Sync RPC** button to verify that this still works. Now click on the patient identification header and select a different patient. Notice how the memo control's contents have now changed. We have now responded to a context change event.

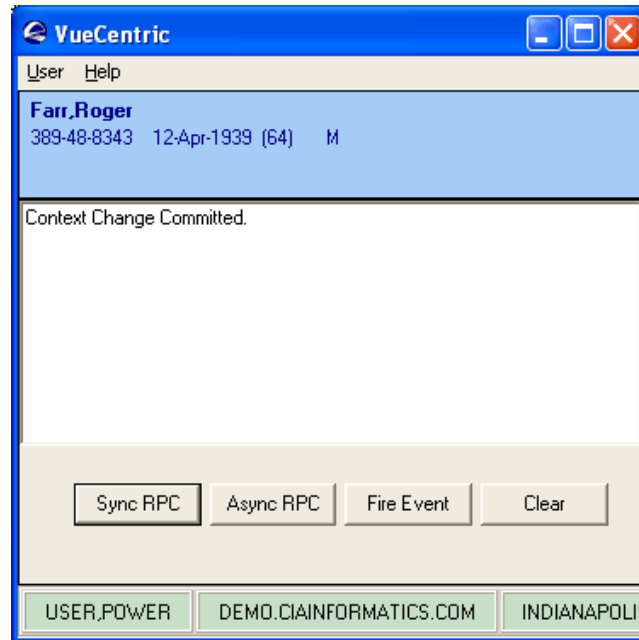



Figure 73-27: Sample Changed Context

### 73.13.8 Calling a Remote Procedure in Asynchronous Mode

Our next task is to support calling our remote procedure asynchronously. To do this, we need to implement the `ICSS_SessionEvents` interface that is defined in the `CSS` type library. This is done in an identical manner to the `ICSS_PatientEvents` interface we implemented in the previous section. First, bring up the type library editor by selecting **View | Type Library**. Select the `DemoControl` node on the left and the `Uses` tab on the right. If only checked entries are showing, right-click on the right pane and select **Show All Type Libraries** from the popup menu. Find the `CIA Component Support Services` library in the list (select the highest version number if you see more than one) and check it. Next select the `DemoControlX` node on the left and the `Implements` tab on the right. Right-click on the right pane and select **Insert Interface** from the popup menu. Select the `ICSS_SessionEvents` interface and click **OK**. Finally, click the  button to refresh your code. You should now see three new methods in your code, all with empty implementations:

```

procedure TDemoControlX.EventCallback(const EventType,
    EventStub: WideString);
begin
end;

procedure TDemoControlX.RPCCallback(Handle: Integer;
    const Data: WideString);
begin
end;

procedure TDemoControlX.RPCCallbackError(Handle, ErrorCode: Integer;
    const ErrorText: WideString);
begin
end;

```

Figure 73-28: Three New Methods

At this point, we will ignore the EventCallback method, but will return to it later. Let us add the following code to the other two methods:

```

procedure TDemoControlX.RPCCallback(Handle: Integer;
    const Data: WideString);
begin
    FHandle := 0;
    Memo1.Lines.Text := Data;
end;

procedure TDemoControlX.RPCCallbackError(Handle, ErrorCode: Integer;
    const ErrorText: WideString);
begin
    FHandle := 0;
    Memo1.Lines.Text := 'An error occurred: ' + ErrorText;
end;

```

Figure 73-29: Added Code

Here, we add the data returned by the asynchronous call to the memo control if it completed normally, or the text of the reported error if it did not.

Next, we will add code to the **Async RPC** button to call our remote procedure in asynchronous mode. To do this, double-click on that button in the form designer and complete the click event handler as shown:

```

procedure TDemoControlX.Button2Click(Sender: TObject);
begin
    if FHandle <> 0
    then FSession.CallRPCAbort(FHandle);

    Memo1.Lines.Text := 'Asynchronous Remote Procedure Invoked.';

    FHandle := FSession.CallRPCAsync('BEHOPTCX PTINQ', FPatient.Handle, self, True);
end;

```

Figure 73-30: Completing the Event Handle

Note that we are first aborting any asynchronous remote procedure in progress before we call it again.

Now add a declaration to the private section of the component's class for the FHandle variable used to stored the returned handle:

```
private
( Private declarations )
FHandle: Integer;
FPatient: ICSS_Patient;
FSession: ICSS Session;
```

Figure 73-31: Added Declaration

Now, because we are already equipped to respond to patient context changes, let us add code to abort an asynchronous call in progress when a context change occurs. We will add this code to the Committed method:

```
procedure TDemoControlX.Committed;
begin
  Memo1.Lines.Text := 'Context Change Committed.';

  if FHandle <> 0
  then begin
    FSession.CallRPCAbort(FHandle);
    FHandle := 0;
  end;
end;
```

Figure 73-32: Code Added to Committed Method

Now it is time to test our component again. Recompile the project by selecting **Project | Build DemoControl** and, because we have again made type library changes, re-register the component by selecting **Run | Register ActiveX Server**. Now restart the Visual Interface Manager as before. This time, click the **Async RPC** button. You should at first see the text “Asynchronous Remote Procedure Invoked.” in the memo control. After a small delay, you should see the results of the remote procedure appear. Note that TaskMan must be running to invoke a remote procedure asynchronously, so if you do not receive data from the call, make certain TaskMan is running.

### 73.13.9 Firing an Event

Let us now add the capability of firing an event. We are going to fire a local event called ‘STATUS’. The Visual Interface Manager subscribes to this event and displays the data associated with it in its status bar. Double-click on the **Fire Event** button in the form designer and complete the click event handler as follows:

```

procedure TDemoControlX.Button3Click(Sender: TObject);
begin
  FSession.EventFireLocal('STATUS','Status event fired by DemoControl.');
```

Figure 73-33: Completed Click Event Handler

Recompile the project and test in the Visual Interface Manager as before. Click the **Fire Event** button and you should see the event data appear in the status bar:



Figure 73-34: Sample Status Bar

### 73.13.10 Subscribing and Responding to an Event

Finally, we will enable our component to respond to STATUS events. This requires two steps. First we must implement the callback interface for responding to events. Because this is the same interface (ICSS\_SessionEvents) we implemented earlier for responding to asynchronous remote procedures, we have already done this. All we need to do is to fill in the implementation for the EventCallback method. Find this method in your component's implementation section and complete as follows:

```

procedure TDemoControlX.EventCallback(const EventType,
  EventStub: WideString);
begin
  Memo1.Lines.Text := EventType + ': ' + EventStub;
end;
```

Figure 73-35: Completed Implementation Section

This will display the event name and data in the memo control.

Next, we need to subscribe to the STATUS event. To do this, return to the Initialize method and add the following line of code:

```

procedure TDemoControlX.Initialize;
begin
  inherited Initialize;
  OnActivate := ActivateEvent;
  OnClick := ClickEvent;
  OnCreate := CreateEvent;
  OnDb1Click := Db1ClickEvent;
  OnDeactivate := DeactivateEvent;
  OnDestroy := DestroyEvent;
  OnKeyPress := KeyPressEvent;
  OnPaint := PaintEvent;
  FSession := CoCSS_Server.Create.Session;
  FPatient := FSession.FindServiceByCLSID(CLASS_Patient) as ICSS_Patient;
  FSession.EventSubscribe('STATUS',self);
end;
```

Figure 73-36: Added Code

Now recompile and test your component as before. Now, clicking the **Fire Event** button also displays the event data in the memo control:

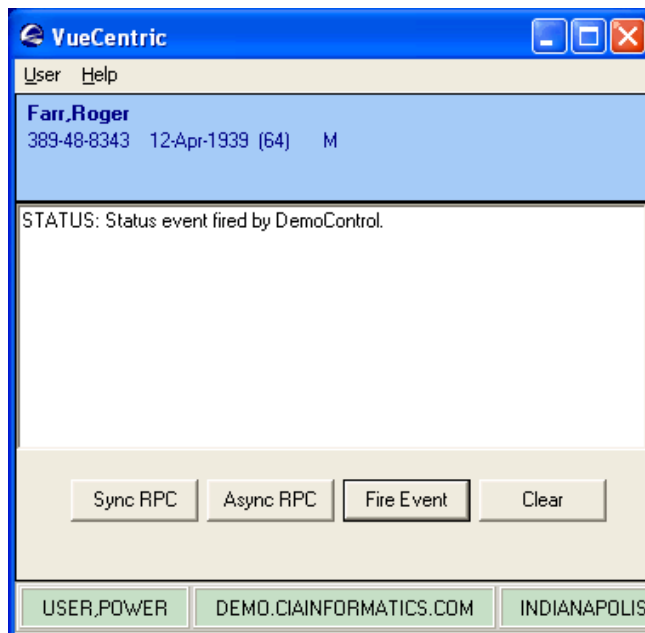


Figure 73-37: Sample Event Data in Memo Control

### 73.13.11 Summary

You have learned how to create an Active Form control, edit type libraries, reference the Session and Patient Context objects, call remote procedures synchronously and asynchronously, and fire and receive events. You should now be well prepared to create components on your own.

## 73.14 Creating Visual Components with Visual Basic

Visual Basic offers the ActiveX Control project type for creating visual components. This control is the equivalent to Delphi's Active Form in that it is essentially a form hosted within an ActiveX wrapper.

### 73.14.1 Creating the ActiveX Control Project

To create an ActiveX Control project, select File | New Project from Visual Basic's main menu. This will bring up the project selection dialog as show below:

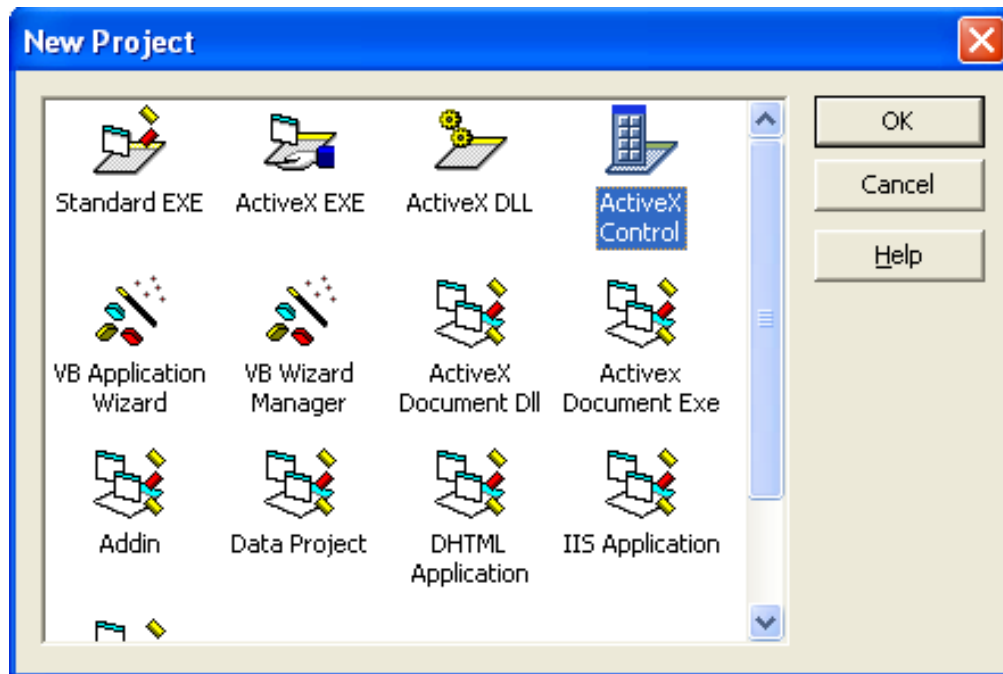


Figure 73-38: New Project Dialog

Select the ActiveX Control project type and click **OK**. This creates a project called Project1 (which will form the first part of the component's programmatic identifier) containing an ActiveX control called UserControl1 (which will form the second part of the component's programmatic identifier):

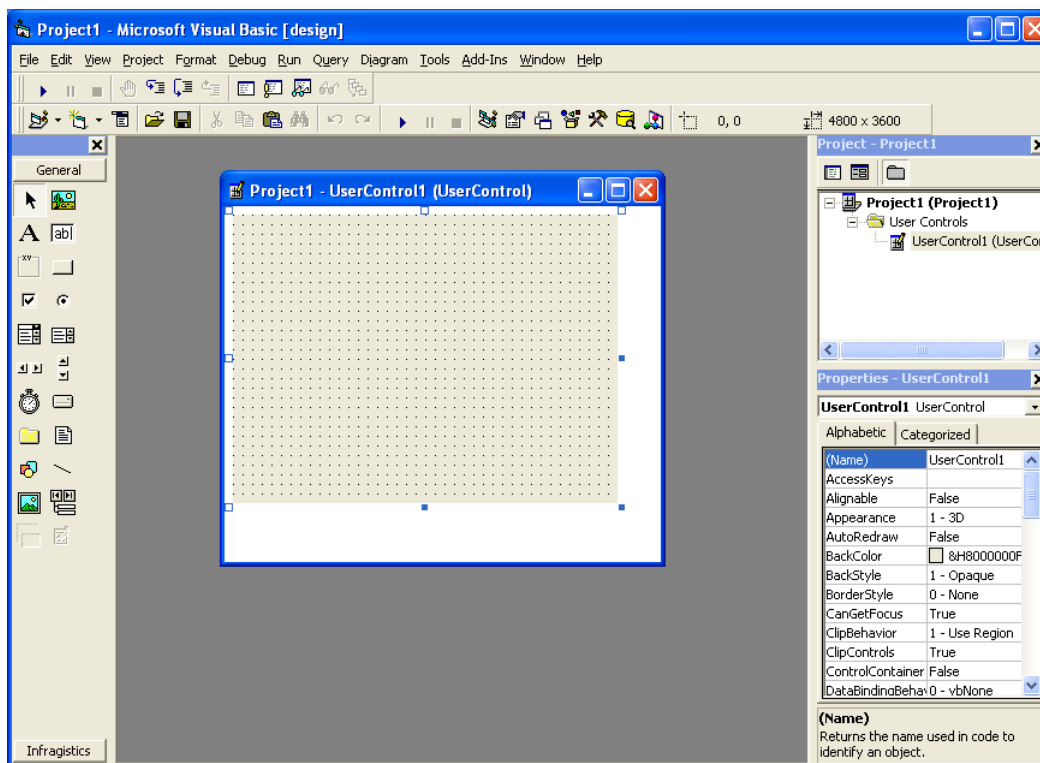


Figure 73-39: New Project Created

Because we would like our programmatic identifier for this control to be DemoControl.DemoControl2, we need to rename the project and the control. To rename the project, select the project in the Project pane (upper right in the above illustration), and edit its Name property in the Properties pane (lower right) to be DemoControl. To rename the control, select the control name in the Project pane and edit its Name property in the Properties pane to be DemoControl2. Our project should now look like this:

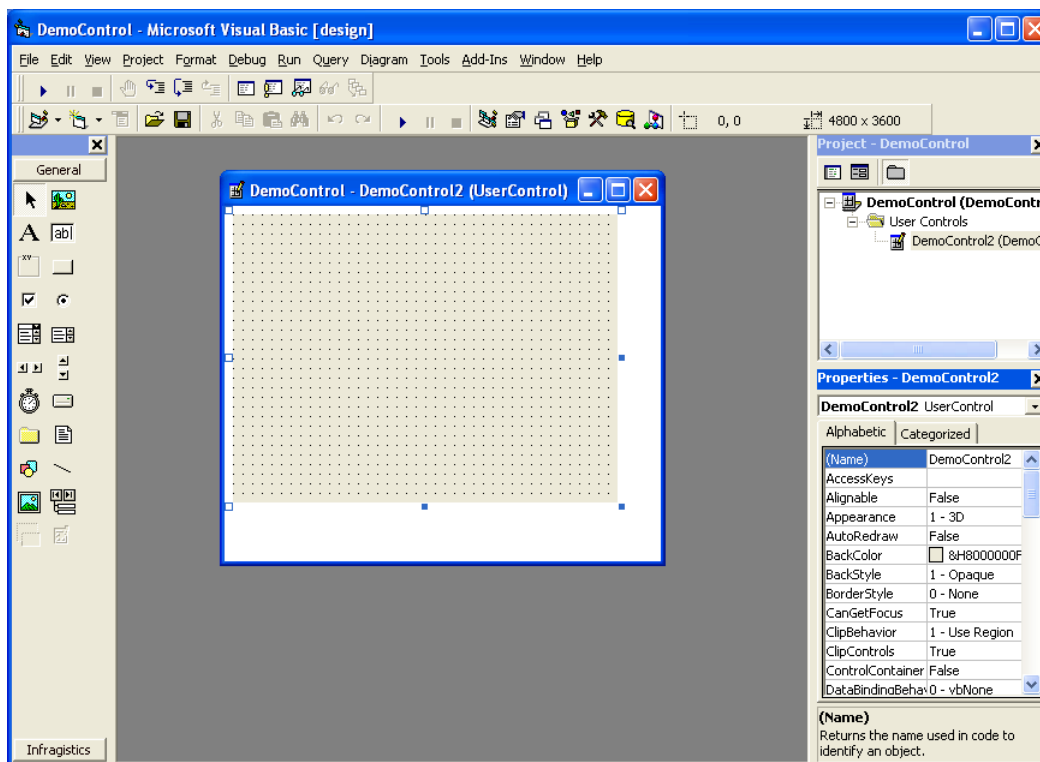


Figure 73-40: Renamed Control

There is one final setting we should change for our project. Select **Project | DemoControl Properties...** from the main menu. Select the **Make** tab and check the **Auto Increment** check box under **Version Number**. Close the dialog by clicking **OK**.

### 73.14.2 Designing the Form

Next, we are going to add a **TextBox** control and four buttons to the form. First, double-click the **TextBox** component on the component palette to add it to the form. Resize the control in the form designer to fill the upper 3/4 of the form. In the properties pane, change its **MultiLine** property to **True**. Add four **CommandButton** controls to the form using the same technique, and arrange them at the bottom of the form as show below:





Figure 73-41: Added TextBox Control and Four Buttons

Now modify the Caption property of each button, from left to right, to **Sync RPC**, **Async RPC**, **Fire Event**, and **Clear**. Double click the rightmost button that is now labeled **Clear** to generate a handler for its click event and complete it as shown below:

```
Private Sub Command4_Click()  
    Text1.Text = ""  
End Sub
```

Figure 73-42: Sample of Completed Handler

### 73.14.3 Accessing the Session Object

*Note:* Visual Basic can have difficulty locating components that are registered using side-by-side versioning. You can find the need to copy the required components to the system directory before adding them as references to the project.

Before we can perform a remote procedure call, we must obtain access to the Session object within the CSS. For Visual Basic to access any COM object, it must first reference that object within its project. To do this, select **Project | References...** from the main menu. In the reference list, find the entry “CIAI Component Support Services” and check it:

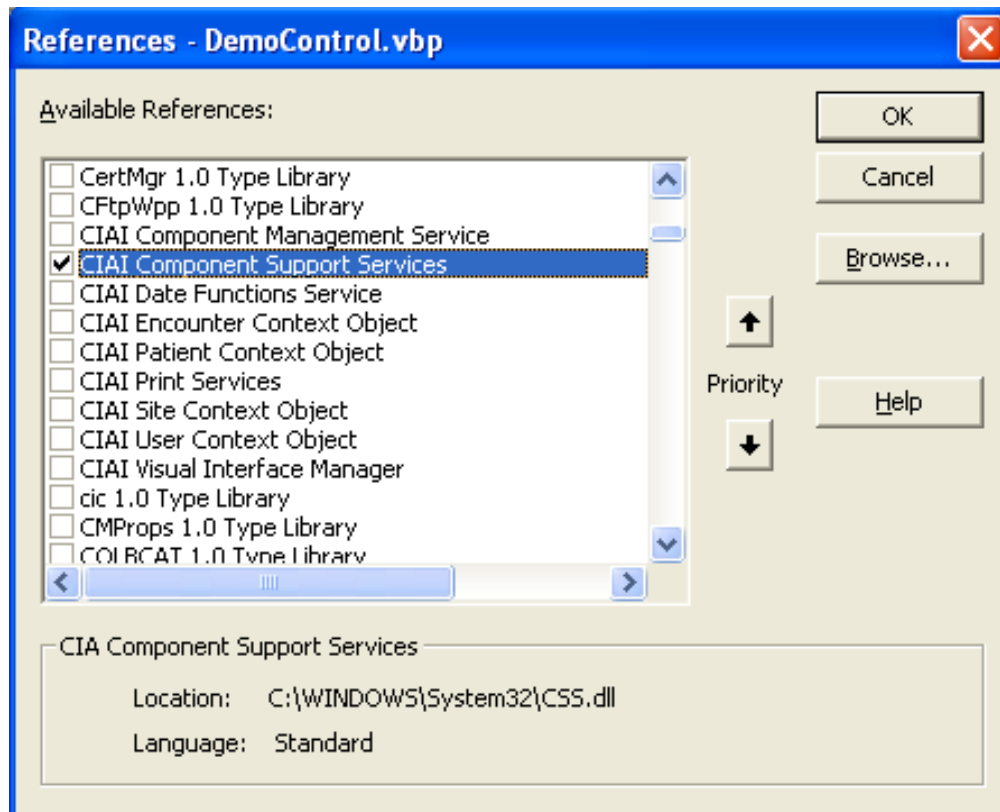


Figure 73-43: Checked Reference

Now click **OK** to close the form. The project now has a reference to the CSS type library and can access objects within it.

Next, we need to declare a global variable to hold our reference to the Session Object. To do this, select (General) in the code editor and add the following code:

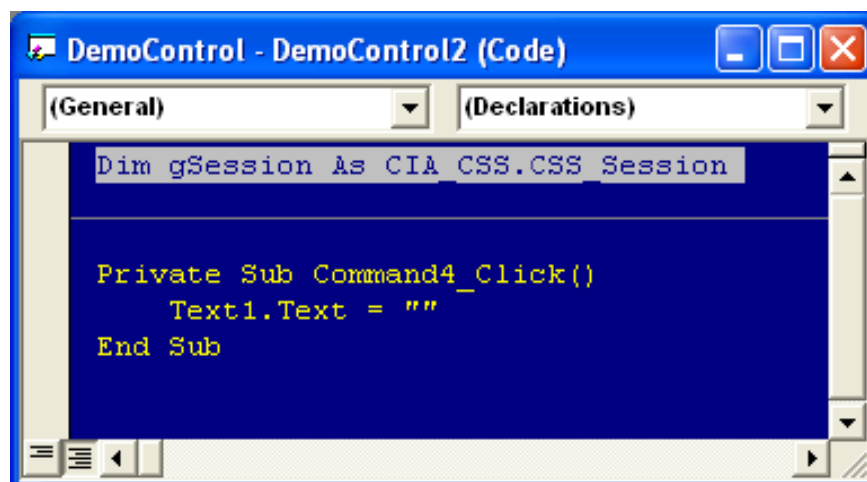
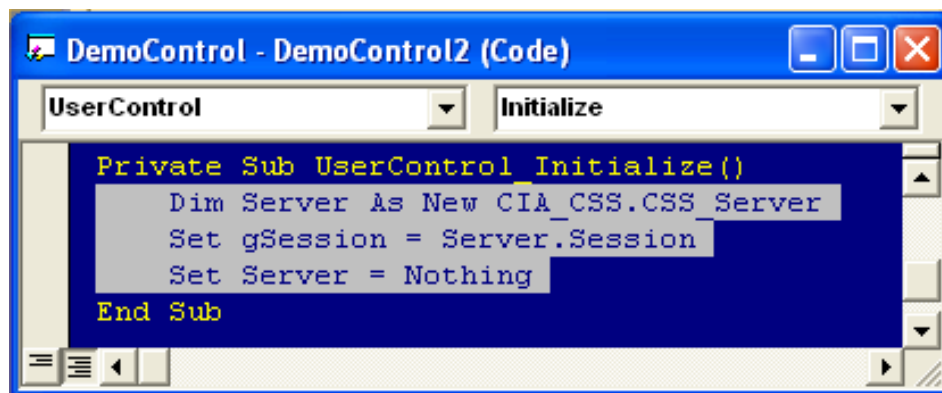


Figure 73-44: Added Code

Now we need to store a reference to the Session object in our global variable, gSession. Select the UserControl class in the left drop-down box of the code editor and

its Initialize method in the right drop-down box. This will create an empty implementation for the Initialize method. Next, complete the implementation by adding the following code:



```
Private Sub UserControl Initialize()  
    Dim Server As New CIA_CSS.CSS_Server  
    Set gSession = Server.Session  
    Set Server = Nothing  
End Sub
```

Figure 73-45: Added Code

This code obtains a temporary reference to the Server object, retrieves a reference to its Session object, and releases the Server object. At this point, you now have a reference to the Session object stored in the global variable, gSession.

## 0.0.2 Accessing the Patient Context Object

Before we can access the Patient Context object, we must add a reference to it to our project in the same manner as the Session object. To do this, select **Project | References...** from the main menu and locate “CIAI Patient Context Object” in the reference list. Check the box next to the entry and close the dialog by clicking **OK**. Next, we need to add a global variable to hold the reference to the Patient Context object, just as we did for the Session object. Return to the (General) section of the code editor and add a global variable declaration as shown:

```
Dim gPatient As CSS.Patient.Patient  
Dim gSession As CIA_CSS.CSS_Session
```

Figure 73-46: Added Global Variable Declaration

Now return to the Initialize method of the UserControl and add the following code to initialize the variable:

```

Private Sub UserControl_Initialize()
    Dim Server As New CIA_CSS.CSS_Server
    Set gSession = Server.Session
    Set Server = Nothing
    Set gPatient = gSession.FindServiceByProgID("CSS PATIENT.PATIENT")
End Sub

```

Figure 73-47: Added Code

### 73.14.4 Calling a Remote Procedure in Synchronous Mode

Now we are ready to add code that will invoke a remote procedure and return its data in the TextBox control. First, we will add a click handler to the first button (labeled **Sync RPC**) to execute a remote procedure that returns detailed information about the currently selected patient and populates the TextBox control with the results. Double-click the left-most button in the form designer and add the following code to its click event handler:

```

Private Sub Command1_Click()
    If gPatient.Handle = 0 Then
        Text1.Text = "No patient is currently selected"
    Else
        Text1.Text = gSession.CallRPCText("BEHOPTCX PTINQ", gPatient.Handle)
    End If
End Sub

```

Figure 73-48: Added Code

This code calls the remote procedure named BEHOPTCX PTINQ, passing it a single parameter corresponding to the patient's internal entry number (DFN), and places the return text in the TextBox control. If the patient context is empty, it displays a message to that effect instead.

### 73.14.5 Testing the Component

Before we proceed to add additional functionality, let us test what we currently have. To do this, we must first compile the project. Select **File | Make DemoControl.ocx...** from the main menu. The project should compile without errors. Next, we need to register our new component to the VueCentric<sup>®</sup> Framework. The easiest way to do this is to use the VueCentric System Management Utility. Run the tool and login to the remote host. Select the Object Registry tab and in the Restrict List To box, check **Local Registry** (this allows us to see local COM objects that aren't yet registered to the Framework). The list of objects will refresh. Now search the list for the programmatic identifier of our newly created component, DemoControl.DemoControl2, and select that entry. In the upper right pane, click **Copy** to copy the local COM registration information into the VueCentric Settings pane. In

that pane, fill in the Name field with the display name for the control. The choice of name is arbitrary. We'll call it Demo Control. Finally fill in the height and width fields with 100 and 200, respectively. Now click **Apply** at the bottom. The display should look something like this:

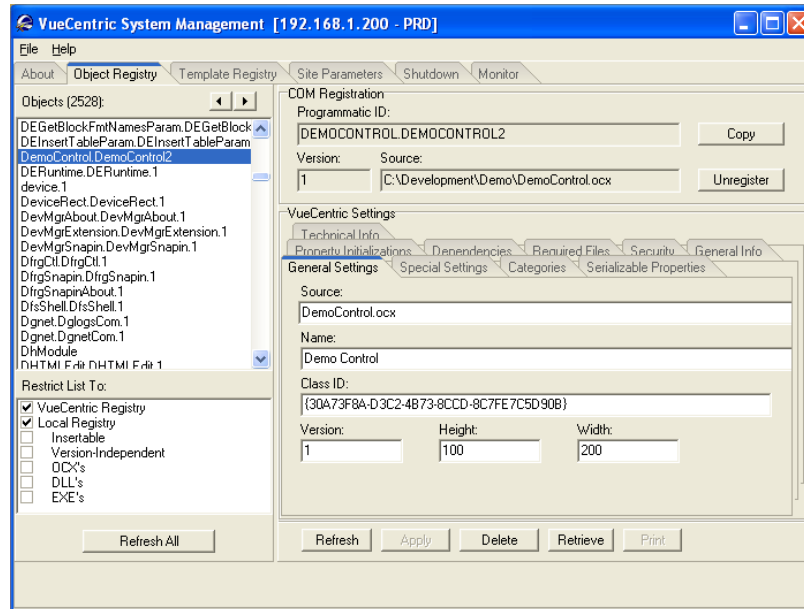


Figure 73-49: Revised Screen

You can now close the utility. To test the component, start the Visual Interface Manager with the following flags:

```
vim.exe /noupdate /blank /trace
```

After logging in, enter design mode, right click the desktop and select **Add Object**. Expand the Name node and locate and add the Patient Identification Header. Top align this control (right-click on it in design mode to do this). Next find and add the Demo Control. Set the alignment of this control to all. Save this as a template named %DEMO for later retrieval. Now exit design mode and click on the **Sync RPC** button. You should see text in the TextBox control. Try clicking on the Patient Identification Header control and changing the patient selection. Note that the contents of the TextBox control is unaffected, because we have not yet subscribed to patient context change events. However, if we click the **Sync RPC** button again, the text that appears in the TextBox control now pertains to the newly selected patient.

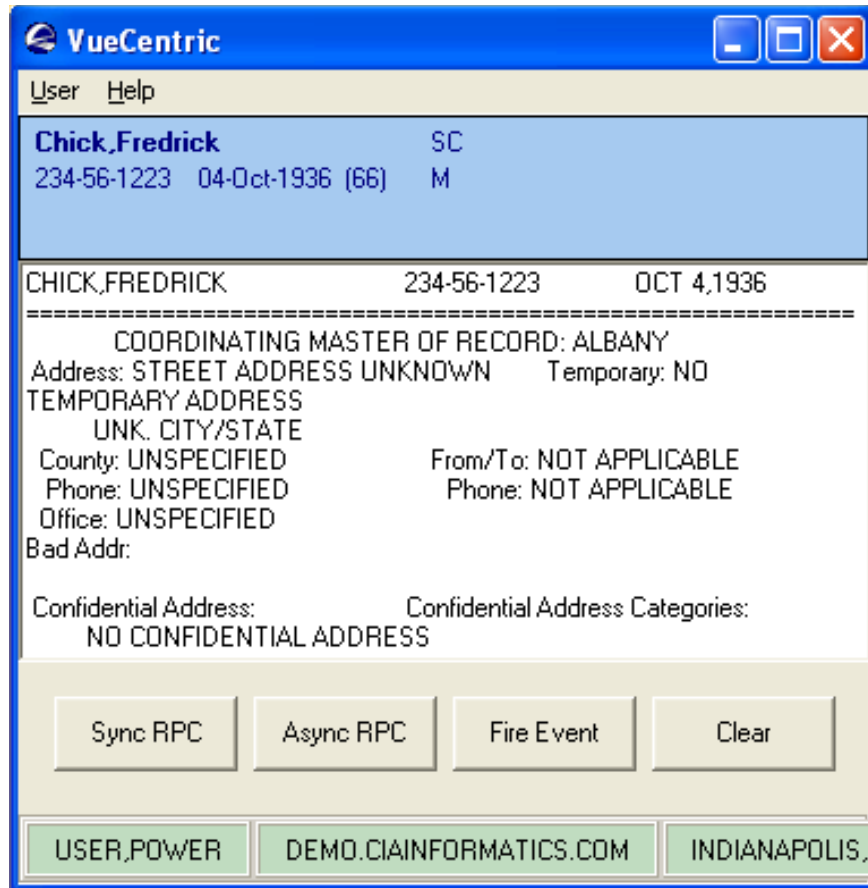


Figure 73-50: Sample Text Pertaining to Patient

Now close the application (note: if you do not close the application now, you will receive an error the next time you try to compile the project because the control's executable image is locked).

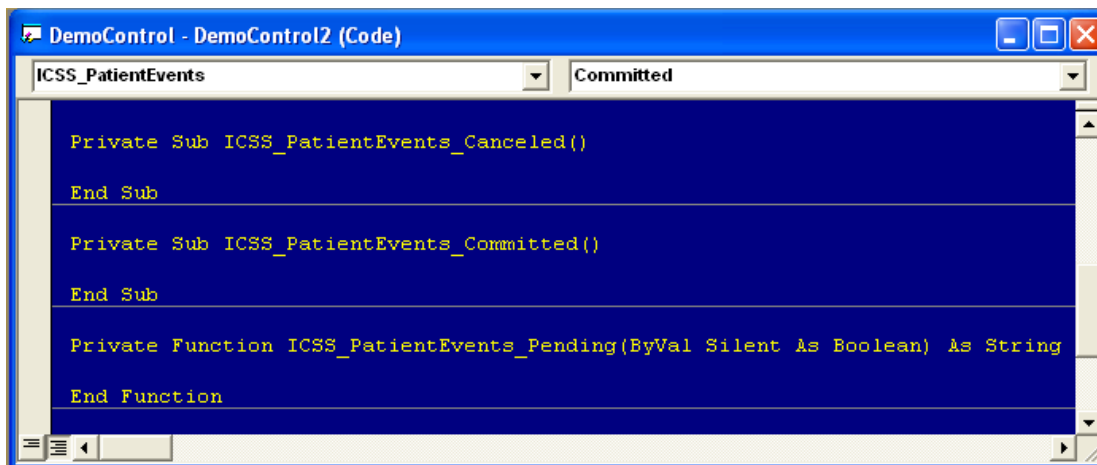
### 73.14.6 Subscribing to Patient Context Changes

As we noticed, our component retrieves information based on the currently selected patient, but it does not respond when the patient selection changes. Let us fix this deficiency by having our component subscribe to patient context changes and modify the TextBox control's contents when the context change occurs. To do this, we need to direct our component to implement the callback interface for patient context changes, `ICSS_PatientEvents`. Return to the (General) section of the code editor and enter the following line of code:

```
Implements ICSS_PatientEvents
Dim gPatient As CSS_Patient.Patient
Dim gSession As CIA_CSS.CSS_Session
```

Figure 73-51: Added Code

From the left drop-down box in the code editor, find and select the entry “ICSS\_PatientEvents.” In the right drop-down box you will see three methods: Pending, Canceled, and Committed. Select each one in turn to generate implementations for each. Your code should look like this:



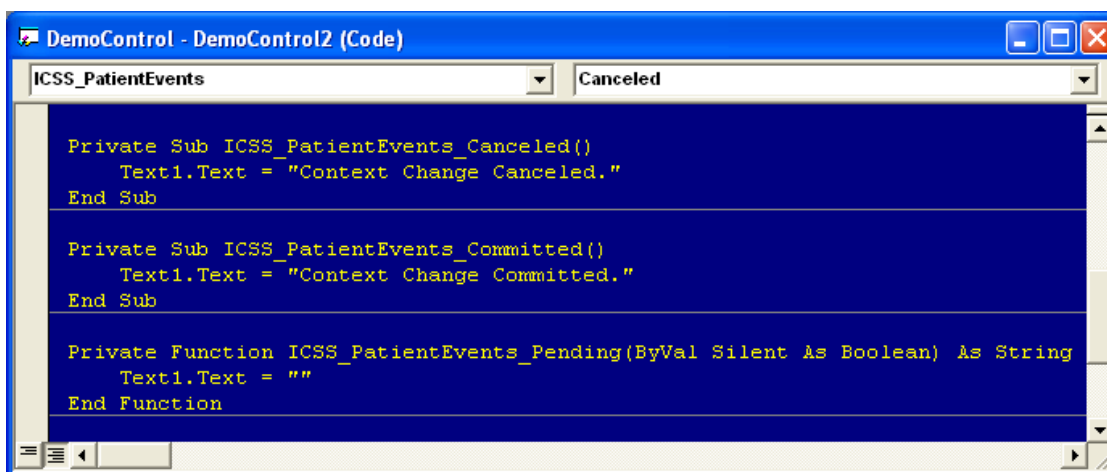
```
Private Sub ICSS_PatientEvents_Canceled()
End Sub

Private Sub ICSS_PatientEvents_Committed()
End Sub

Private Function ICSS_PatientEvents_Pending(ByVal Silent As Boolean) As String
End Function
```

Figure 73-52: New Code

We will add code to each of these implementations to populate the TextBox control with text indicating that each method has been invoked. Complete the implementations with the code shown below:



```
Private Sub ICSS_PatientEvents_Canceled()
    Text1.Text = "Context Change Canceled."
End Sub

Private Sub ICSS_PatientEvents_Committed()
    Text1.Text = "Context Change Committed."
End Sub

Private Function ICSS_PatientEvents_Pending(ByVal Silent As Boolean) As String
    Text1.Text = ""
End Function
```

Figure 73-53: Added Code

When a context change is initiated, the Pending method will be called first. In this method, we simply clear the TextBox control’s contents. Because we are not setting the return value for the Pending method, we are essentially voting YES to the context change. Assuming nothing else cancels the pending change, the Committed method is called next. In that method, we set the TextBox control’s text to indicate that the context change was committed. Now compile the project by selecting File | Make DemoControl.ocx... from the menu. Now restart the Visual Interface Manager, this time with the following command line options:

```
vim.exe /nouupdate /trace /template=%DEMO
```

Once you login, you should see your component much as it looked before. Click the **Sync RPC** button to verify that this still works. Now click on the patient identification header and select a different patient. Notice how the TextBox control's contents has now changed. We have now responded to a context change event.

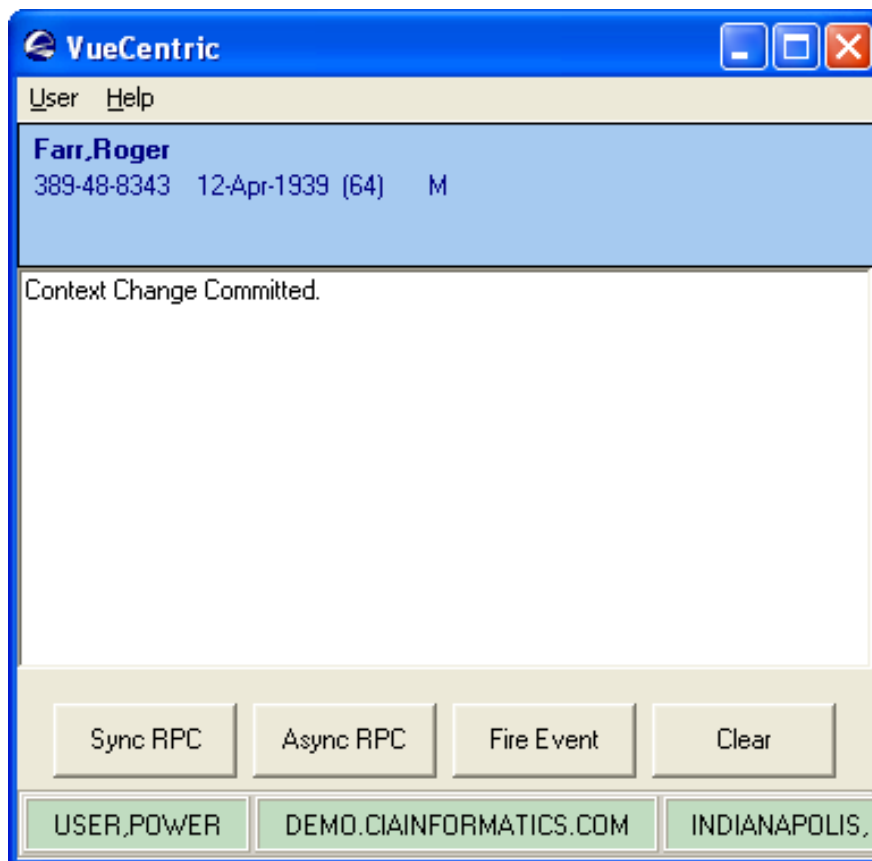


Figure 73-54: Sample Context Change Event

### 73.14.7 Calling a Remote Procedure in Asynchronous Mode

Our next task is to support calling our remote procedure asynchronously. To do this, we need to implement the `ICSS_SessionEvents` interface that is defined in the `CSS` type library. This is done in an identical manner to the `ICSS_PatientEvents` interface we implemented in the previous section. Return again to the (General) section of the code editor and add the following:

```
Implements ICSS_SessionEvents
Implements ICSS_PatientEvents

Dim gPatient As CSS_Patient.Patient
Dim gSession As CIA_CSS.CSS_Session
```

Figure 73-55: Added Code



As we did with the ICSS\_PatientEvents interface, select “ICSS\_SessionEvents” from the left drop-down box and each of its three methods in turn from the right drop-down box. This will create blank implementations for each method.

```
Private Sub ICSS_SessionEvents_EventCallback(ByVal EventType As String, ByVal EventStub As String)
End Sub

Private Sub ICSS_SessionEvents_RPCCallback(ByVal Handle As Long, ByVal Data As String)
End Sub

Private Sub ICSS_SessionEvents_RPCCallbackError(ByVal Handle As Long, ByVal ErrorCode As Long, ByVal ErrorText As String)
End Sub
```

Figure 73-56: Blank Implementations Created

At this point, we will ignore the EventCallback method, but will return to it later. Let us add the following code to the other two methods:

```
Private Sub ICSS_SessionEvents_RPCCallback(ByVal Handle As Long,
    gHandle = 0
    Text1.Text = Data
End Sub

Private Sub ICSS_SessionEvents_RPCCallbackError(ByVal Handle As
    gHandle = 0
    Text1.Text = "An error occurred: " + ErrorText
End Sub
```

Figure 73-57: Added Code

Here, we add the data returned by the asynchronous call to the TextBox control if it completed normally, or the text of the reported error if it did not.

Next, we will add code to the **Async RPC** button to call our remote procedure in asynchronous mode. To do this, double-click on that button in the form designer and complete the click event handler as shown:

```
Private Sub Command2_Click()
    If gHandle <> 0 Then gSession.CallRPCAbort (gHandle)
    Text1.Text = "Asynchronous Remote Procedure Invoked."
    gHandle = gSession.CallRPCAsync("BEHOPTCX PTINQ", gPatient.Handle, Me, True)
End Sub
```

Figure 73-58: Added Code

Note that we are first aborting any asynchronous remote procedure in progress before we call it again.

Now add the global variable declaration to the (General) section of the code editor to store the returned handle:

```
Dim gHandle As Integer
Dim gPatient As CSS_Patient.Patient
Dim gSession As CIA_CSS.CSS_Session
```

Figure 73-59: Added Code

Now, because we are already equipped to respond to patient context changes, let us add code to abort an asynchronous call in progress when a context change occurs. We will add this code to the Committed method:

```
Private Sub ICSS_PatientEvents_Committed()
    Text1.Text = "Context Change Committed."
    If gHandle <> 0 Then
        gSession.CallRPCAbort (gHandle)
        gHandle = 0
    End If
End Sub
```

Figure 73-60: Added Code

Now it is time to test our component again. Recompile the project by selecting **File | Make DemoControl.ocx...** Now restart the Visual Interface Manager as before. This time, click the **Async RPC** button. You should at first see the text “Asynchronous Remote Procedure Invoked.” in the TextBox control. After a small delay, you should see the results of the remote procedure appear. Note that TaskMan must be running to invoke a remote procedure asynchronously, so if you do not receive data from the call, make certain TaskMan is running.

### 73.14.8 Firing an Event

Let us now add the capability of firing an event. We are going to fire a local event called ‘STATUS’. The Visual Interface Manager subscribes to this event and displays the data associated with it in its status bar. Double-click on the **Fire Event** button in the form designer and complete the click event handler as follows:

```
Private Sub Command3 Click()
    gSession.EventFireLocal "STATUS", "Status event fired by DemoControl."
End Sub
```

Figure 73-61: Added Code

Recompile the project and test in the Visual Interface Manager as before. Click the **Fire Event** button and you should see the event data appear in the status bar:

USER,POWER	DEMO.CIAINFORMATICS.COM	INDIANAPOLIS, IN	Status event fired by DemoControl.
------------	-------------------------	------------------	------------------------------------

Figure 73-62: Status Bar

### 73.14.9 Subscribing and Responding to an Event

Finally, we will enable our component to respond to STATUS events. This requires two steps. First we must implement the callback interface for responding to events. Because this is the same interface (ICSS\_SessionEvents) we implemented earlier for responding to asynchronous remote procedures, we have already done this. All we need to do is to fill in the implementation for the EventCallback method. Find this method in your component's implementation section and complete as follows:

```
Private Sub ICSS_SessionEvents_EventCallback(ByVal EventType  
    Text1.Text = EventType + ": " + EventStub  
End Sub
```

Figure 73-63: Added Code

This will display the event name and data in the TextBox control.

Next, we need to subscribe to the STATUS event. To do this, return to the Initialize method and add the following line of code:

```
Private Sub UserControl_Initialize()  
    Dim Server As New CIA_CSS.CSS_Server  
    Set gSession = Server.Session  
    Set Server = Nothing  
    Set gPatient = gSession.FindServiceByProgID("CSS_PATIENT.PATIENT")  
    gSession.EventSubscribe "STATUS", Me  
End Sub
```

Figure 73-64: Added Code

Now recompile and test your component as before. Now, clicking the **Fire Event** button also displays the event data in the TextBox control:

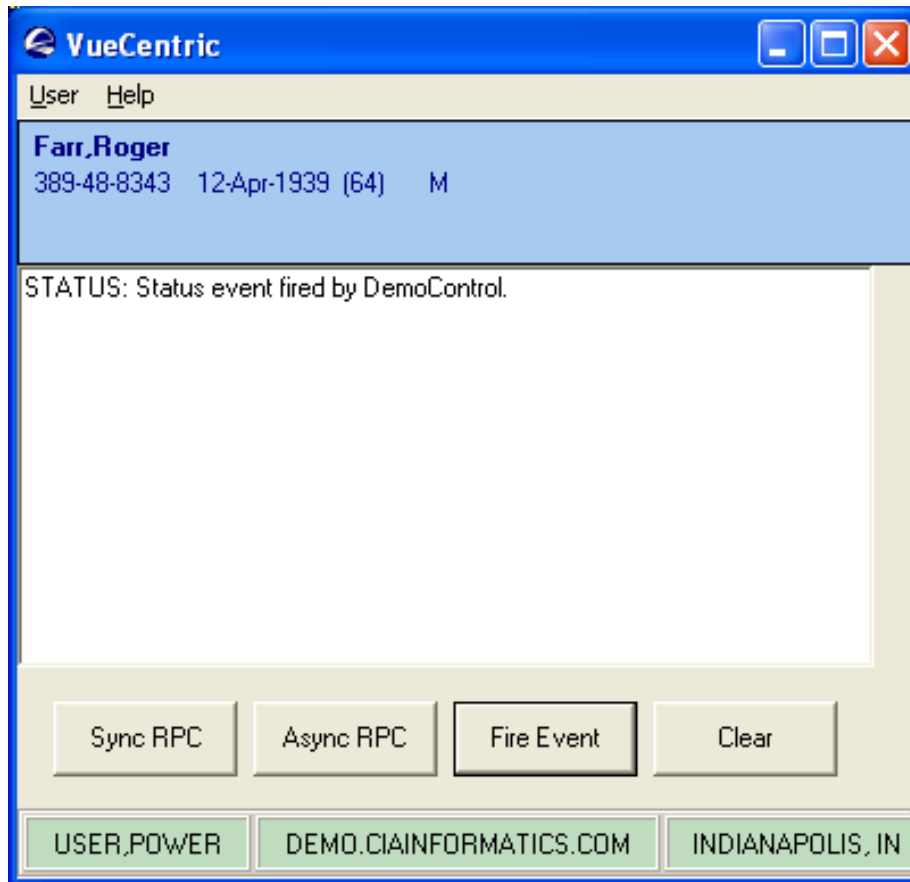


Figure 73-65: Displayed Event Data

### 73.14.10 Summary

You have learned how to create an ActiveX control, reference the Session and Patient Context objects, implement interfaces, call remote procedures synchronously and asynchronously, and fire and receive events. You should now be well prepared to create components on your own.

## 73.15 Creating Visual Components with C#

.NET Window Controls can be used in the VIM through a COM-interoperability feature of the .NET framework known as the COM Callable Wrapper (CCW). This wrapper transparently exposes a .NET Window Control as an ActiveX object and requires little effort on the part of the developer. While any .NET-compatible language can be used, we choose C# for this example.

### 73.15.1 Creating the Windows Control Project

To create an Active Form project, select File | New | Project... This will bring up the project selection dialog as shown below:

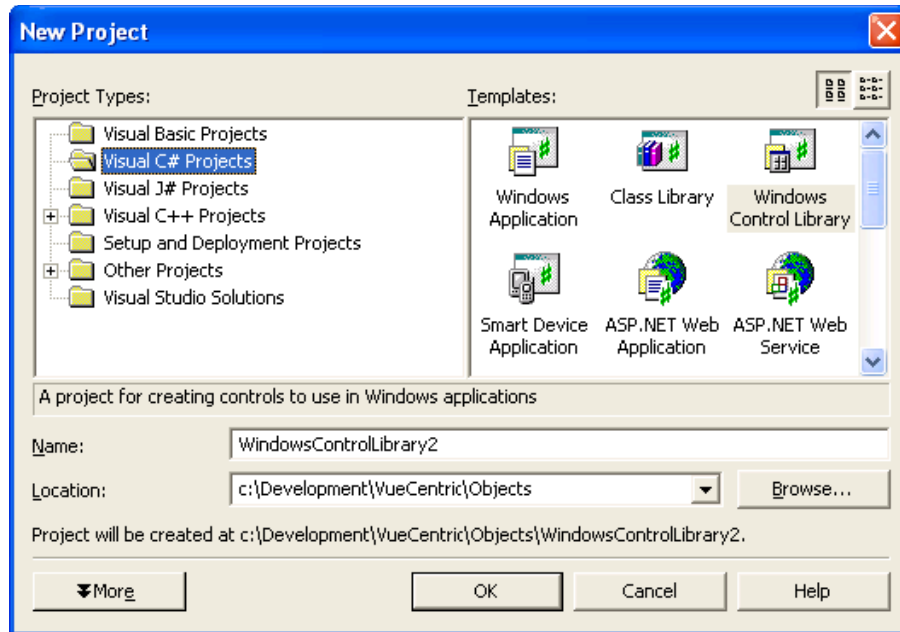


Figure 73-66: Project Selection Dialog

Select Visual C# Projects in the left pane and Windows Control Library in the right pane. Modify the Name to DemoControl and click **OK**.

The form designer will appear:

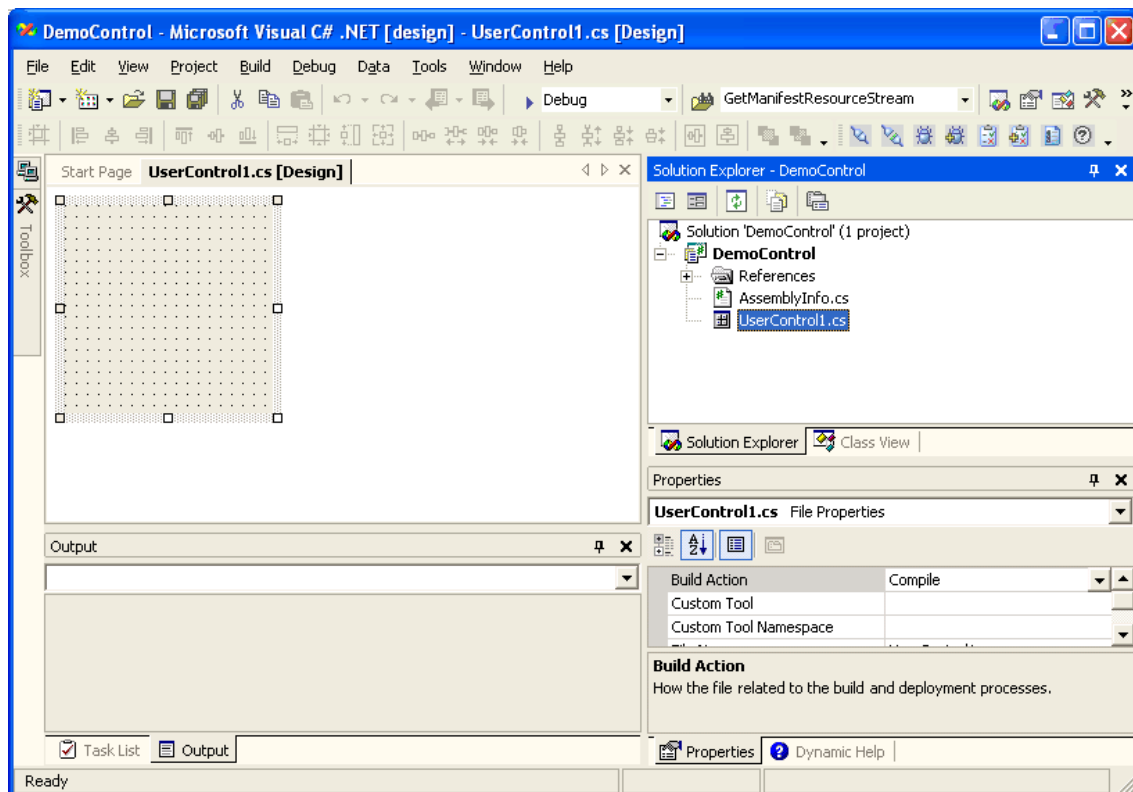


Figure 73-67: Sample Form Designer

Next, right-click the node named “DemoControl” from the Solution Explorer pane (upper right) and select **Properties** from the popup menu. Select **Configuration Properties | Build** from the left pane and change the “Register for COM Interop” property in the right pane to True. Click **OK** to close the dialog.

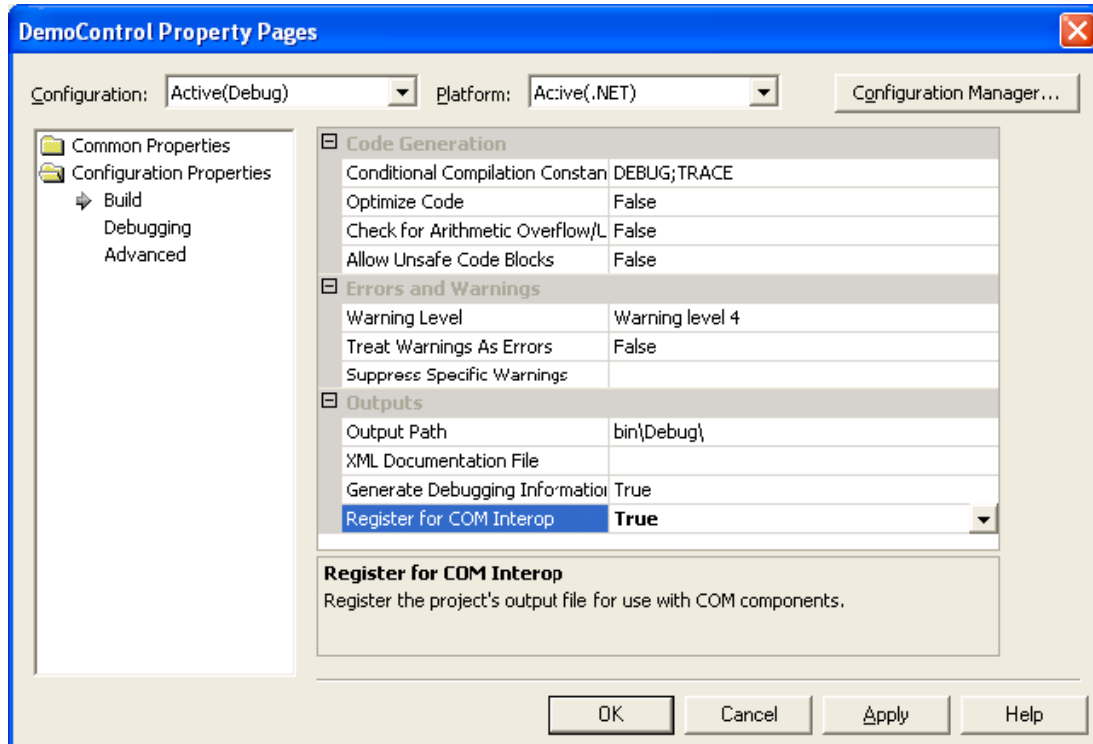


Figure 73-68: Changed Property

Next, rename the component from UserControl1 to DemoControlCS by first clicking the control’s form in the Form Designer (upper left) and then modifying its Name property in the Properties pane (lower right) to DemoControlCS. Also, change the BackColor property to “ControlLight”. (If you do not change the default color, it will appear black in the VIM.) Then resize the form on the form designer to accommodate the controls you will be placing on it by grabbing the lower right sizing grip and dragging it to its desired position. Your project should now look like this:

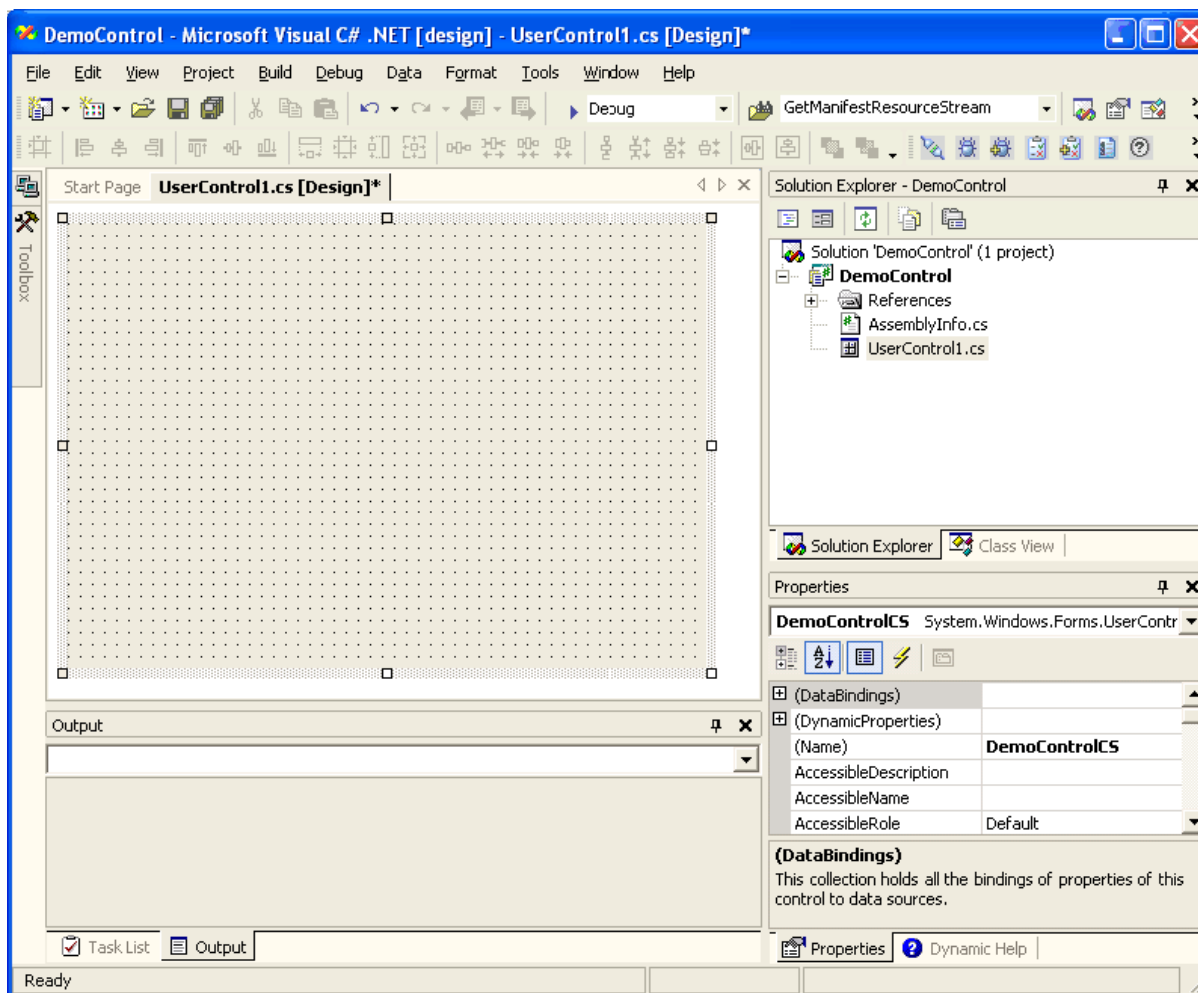


Figure 73-69: Revised Project

Now we are going to add a text box and four buttons to the form. First, open the component palette by hovering the mouse pointer over the ToolBox icon to the left of the Form Designer. Locate the TextBox control (you might need to scroll the component list to find it) and double-click to add it to your form. Next, from the Form Designer, select the newly added TextBox control by clicking on it. In the Properties pane, edit its Dock property to Top and its MultiLine property to True. Then resize it to fill most of the form (allowing space at the bottom for buttons) by grabbing its resizing grip and dragging to the desired position.

Next, add four buttons to the form by opening the component palette as before and double-clicking on the Button control. Repeat this three more times. You will see four buttons on your form. Drag them to the desired positions at the bottom of the form. Set the Anchor property for each button to (Bottom,Left). An easy way to do this is to select the first button by clicking on it, then clicking on the remaining three while holding down the Shift key. This will select all four buttons. Then modify the Anchor property to the desired value. This will change the property values for all four buttons.

While we are here, we are going to add an empty handler for the Load Event. To do this, switch the property editor view to events by clicking on the lightning bolt icon in the toolbar. Find the Load event in the list and double click in the empty box to the right of it. You should now see the code editor with your empty event handler:

```
private void DemoControlCS_Load(object sender, System.EventArgs e)
{
}

```

Figure 73-70: Code Editor

We will complete this event handler in a moment. For now, return to the design view by clicking on the appropriate tab. Your form should now look like this:

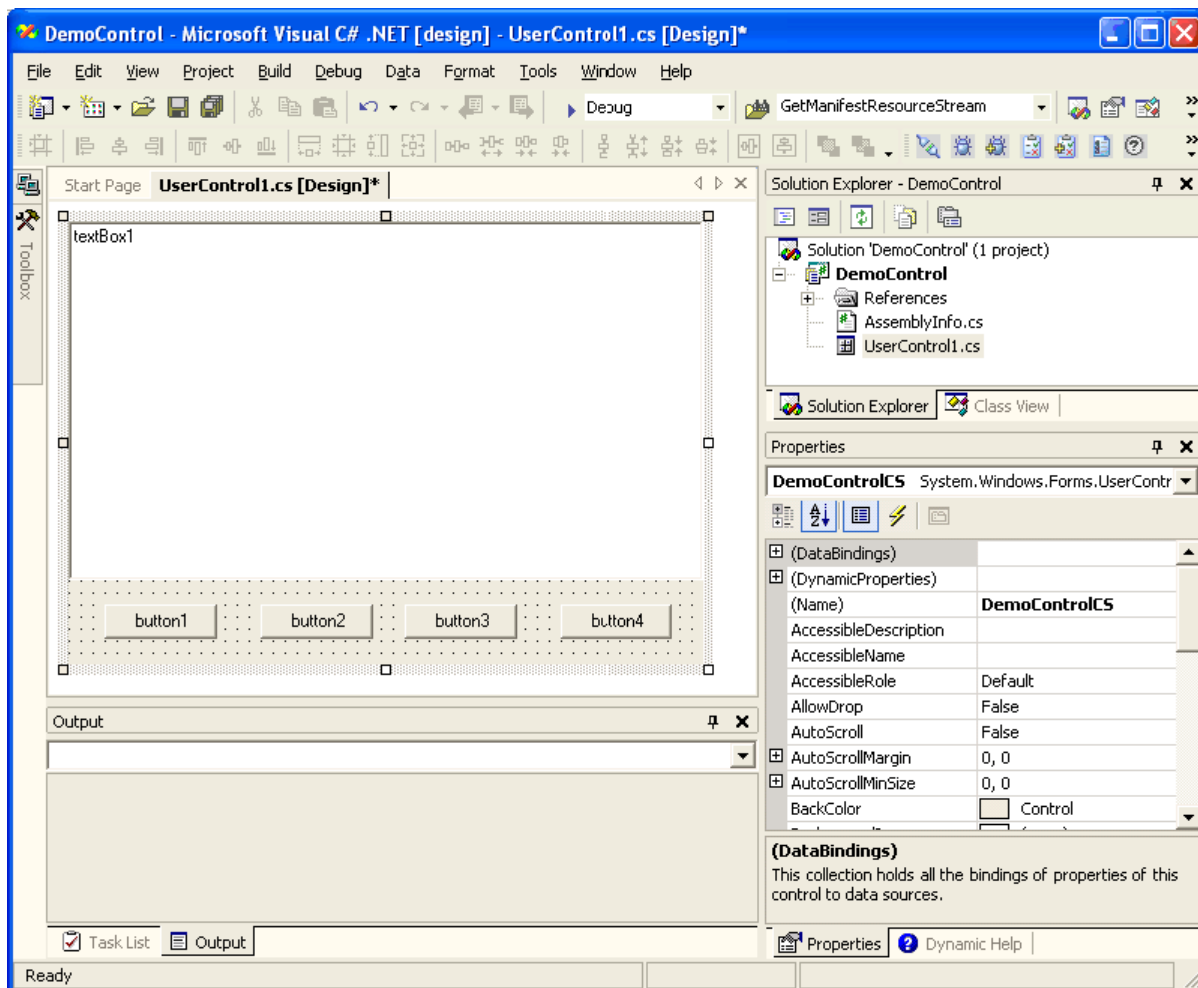


Figure 73-71: Revised Form

Now modify the Text property of each button, from left to right, to **Sync RPC**, **Async RPC**, **Fire Event**, and **Clear**. Double click the right-most button that is now labeled **Clear** to generate a handler for its click event and complete it as shown below:



```
private void button4_Click(object sender, System.EventArgs e)
{
    textBox1.Clear();
}
```

Figure 73-72: Added Code

### 73.15.2 Accessing the Session Object

Before we can perform a remote procedure call, we must obtain access to the Session object within the CSS. For C# to access any COM object, we must first add its reference information to the project. This is done by selecting **Project | Add Reference...** from the menu. Select the **COM** tab and locate the entry named **CIAI Component Support Services** from the list. Select that entry by clicking on it, then click the **Select** button. The list item should now appear in the bottom pane of the dialog as shown:

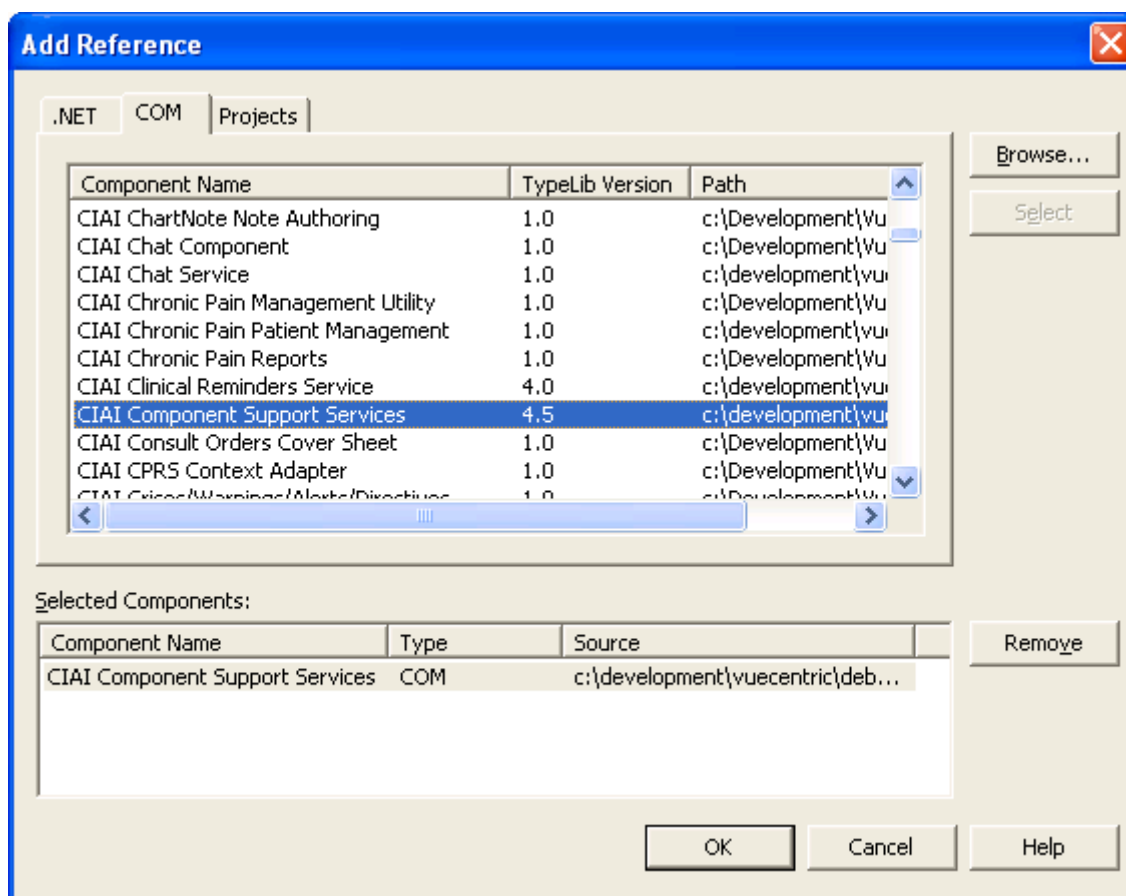


Figure 73-73: Sample Add Reference Dialog

Finally, click **OK** to close the dialog and add the reference to your project. You will now see a new entry in the Solution Explorer pane under the References node named **CIA\_CSS**.

Now that you have added reference information for the CSS, we can add code to access the Session object. First, add an instance variable to the class declaration of your control and name it session. This will be used to hold a reference to the session object.

```
public class DemoControlCS : System.Windows.Forms.UserControl
{
    private CIA_CSS.ICSS_Session session;
    private System.Windows.Forms.TextBox textBox1;
    private System.Windows.Forms.Button button1;
    private System.Windows.Forms.Button button2;
    private System.Windows.Forms.Button button3;
    private System.Windows.Forms.Button button4;
```

Figure 73-74: Added Code

To obtain a reference to the Session object, add the following line of code to the Load event handler we created earlier:

```
private void DemoControlCS_Load(object sender, System.EventArgs e)
{
    session=new CIA_CSS.CSS_ServerClass().Session;
}
```

Figure 73-75: Added Code

We must also be sure to properly release this reference when the object is destroyed. To do this, locate the Dispose method implementation for the control and add the following lines of code:

```
protected override void Dispose( bool disposing )
{
    if(session!=null)
    {
        Marshal.ReleaseComObject(session);
        session = null;
    }

    if( disposing )
    {
        if( components != null )
            components.Dispose();
    }
    base.Dispose( disposing );
}
```

Figure 73-76: Added Code

### 73.15.3 Accessing the Patient Context Object

Before we can access the Patient Context object, its reference information must be added to the project in the same manner as you did with the Session object. From the

Add Reference dialog, double-click the entry “CIAI Patient Context Object” to add it and click **OK** to close the dialog. A new reference should appear in the Solution Explorer pane named CSS\_Patient.

Next, declare an instance variable to hold the reference:

```
public class DemoControlCS : System.Windows.Forms.UserControl
{
    private CIA_CSS.ICSS_Session session;
    private CSS_Patient.ICSS_Patient patient;
    private System.Windows.Forms.TextBox textBox1;
    private System.Windows.Forms.Button button1;
    private System.Windows.Forms.Button button2;
    private System.Windows.Forms.Button button3;
    private System.Windows.Forms.Button button4;
```

Figure 73-77: Added Code

Now, we need to obtain a reference to the Patient Context object. Because this object is just a special kind of service, we use the Session object to retrieve a reference to it. Add the following line of code to the Load event handler to do this:

```
private void DemoControlCS_Load(object sender, System.EventArgs e)
{
    session=new CIA_CSS.CSS_ServerClass().Session;
    patient=session.FindServiceByProgID("CSS_Patient.Patient") as CSS_Patient.ICSS_Patient;
}
```

Figure 73-78: Added Code

The above code will cause the Session to locate (and start if not already started) the indicated service, in this case the Patient Context object. Because this function returns a reference to the default IUnknown interface, it must be cast to the desired interface, in this case ICSS\_Patient.

As before, we must release all object references when the control is destroyed. To do this, add the following lines of code to the Dispose method:

```

protected override void Dispose( bool disposing )
{
    if(session!=null)
    {
        Marshal.ReleaseComObject(session);
        session = null;
    }

    if (patient!=null)
    {
        Marshal.ReleaseComObject(patient);
        patient = null;
    }

    if( disposing )
    {
        if( components != null )
            components.Dispose();
    }
    base.Dispose( disposing );
}

```

Figure 73-79: Added Code

## 73.16 Calling a Remote Procedure in Synchronous Mode

Now we are ready to add code that will invoke a remote procedure and display its data in the TextBox control. First, we will add a click handler to the first button (labeled **Sync RPC**) to execute a remote procedure that returns detailed information about the currently selected patient and populates the TextBox control with the results. Double-click the left-most button in the form designer and add the following code to its click event handler:

```

private void button1_Click(object sender, System.EventArgs e)
{
    if(patient.Handle==0)
        textBox1.Text="No patient is currently selected";
    else
        textBox1.Text=session.CallRPCText("BEHOPTCX PTINQ",patient.Handle);
}

```

Figure 73-80: Added Code

This code calls the remote procedure named BEHOPTCX PTINQ, passing it a single parameter corresponding to the patient's internal entry number (DFN), and displays the return text in the TextBox control. If the patient context is empty, it displays a message to that effect instead.

### 73.16.1 Testing the Component

Before we proceed to add additional functionality, let us test what we currently have. Before compiling the project, we must first assign a unique GUID to our new control.

If we do not do this, the control will be assigned a new GUID every time it is registered, which is not desirable. First, we must add a reference to the COM Interop Services namespace to our project as shown:

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Windows.Forms;
using System.Runtime.InteropServices;
```

Figure 73-81: Added Code

Next, we must obtain a unique GUID to assign to our component. To do this, select **Tools | Create GUID** from the menu. The following dialog appears:

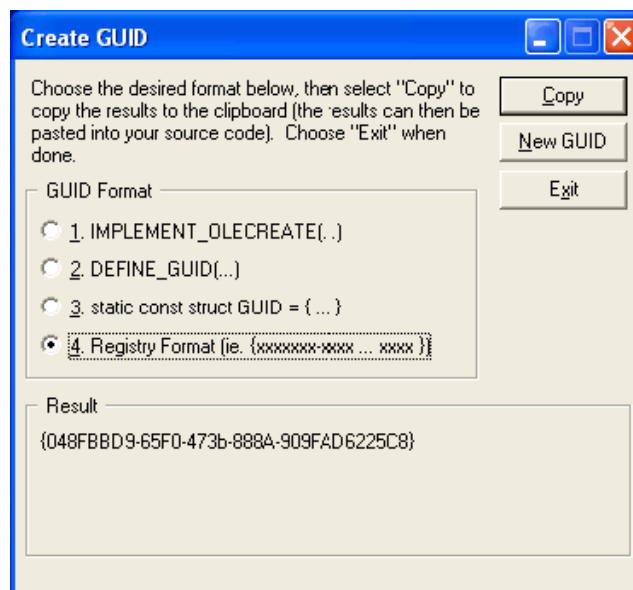


Figure 73-82: Create GUID Dialog

Select the format labeled “Registry Format,” click the **Copy** button to copy the GUID to the clipboard, then click **Exit** to close the dialog. To assign a static GUID to our control, insert the following attribute declaration just prior to the class declaration for the control. Where the GUID appears in the declaration, paste the contents of the clipboard and delete the curly braces.

```
[GuidAttribute("048FBBD9-65F0-473b-888A-909FAD6225C8")]
public class DemoControlCS : System.Windows.Forms.UserControl
{
    private CIA_CSS.ICSS_Session session;
    private CSS_Patient.ICSS_Patient patient;
```

Figure 73-83: Added Code

Now we are ready to compile the project. Select **Build | Build Solution** from the menu.

Finally, we need to register our new component to the **VueCentric®** Framework. The easiest way to do this is to use the **VueCentric®** System Management Utility. Run the tool and login to the remote host. Select the **Object Registry** tab and in the **Restrict List To** box, check **Local Registry** (this allows us to see local COM objects that aren't yet registered to the Framework). The list of objects will refresh. Now search the list for the programmatic identifier of our newly created component, **DemoControl.DemoControlCS**, and select that entry. In the upper right pane, click **Copy** to copy the local COM registration information into the **VueCentric Settings** pane. In that pane, fill in the **Name** field with the display name for the control. The choice of name is arbitrary. We'll call it **C# Demo Control**. Next, fill in the **height** and **width** fields with **100** and **200**, respectively. Finally, switch to the **Special Settings** tab and check the box labelled **.NET Component**. Switch back to the **General Settings** tab and click **Apply** at the bottom. The display should look something like this:

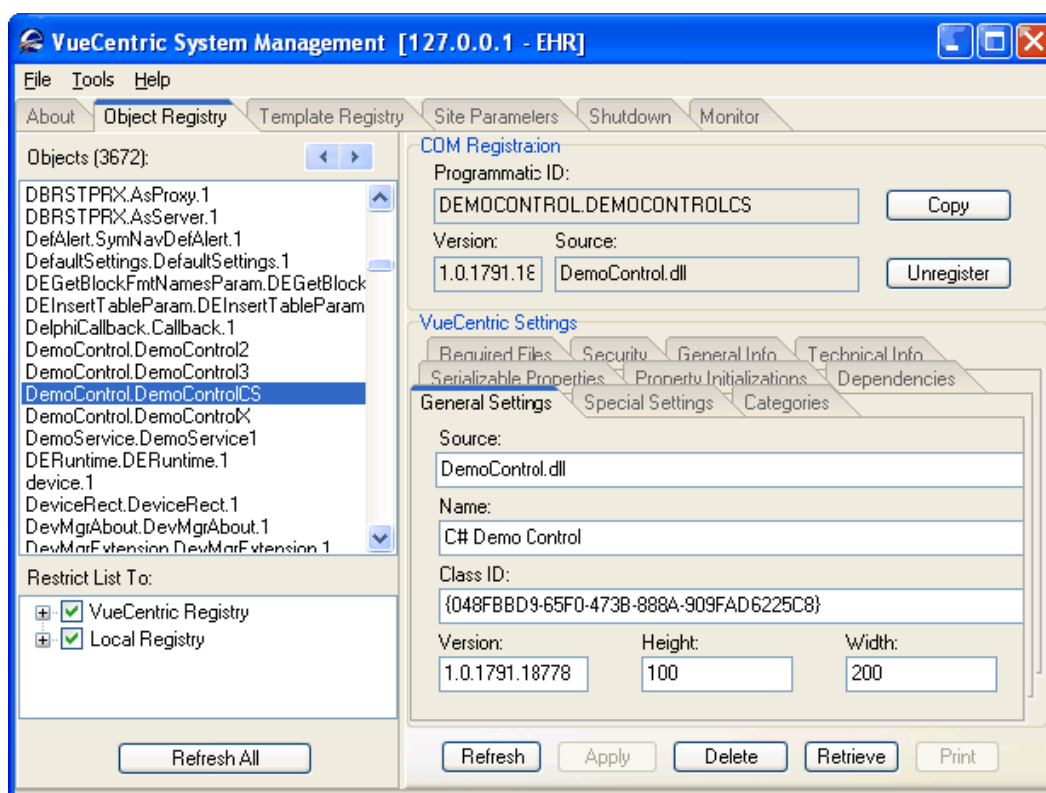


Figure 73-84: Revised Dialog

You can now close the utility. To test the component, start the Visual Interface Manager with the following flags:

```
vim.exe /noupdate /blank /trace
```

After logging in, enter design mode, right click the desktop and select **Add Object**. Expand the **Name** node and locate and add the **Patient Identification Header**. Top align this control (right-click on it in design mode to do this). Next find and add the **Demo Control**. Set the alignment of this control to all. Save this as a template named **%DEMO** for later retrieval. Now exit design mode and click on the **Sync RPC** button.

You should see text in the text box control. Try clicking on the Patient Identification Header control and changing the patient selection. Note that the contents of the text box control is unaffected, because we have not yet subscribed to patient context change events. However, if we click the **Sync RPC** button again, the text that appears in the text box control now pertains to the newly selected patient.

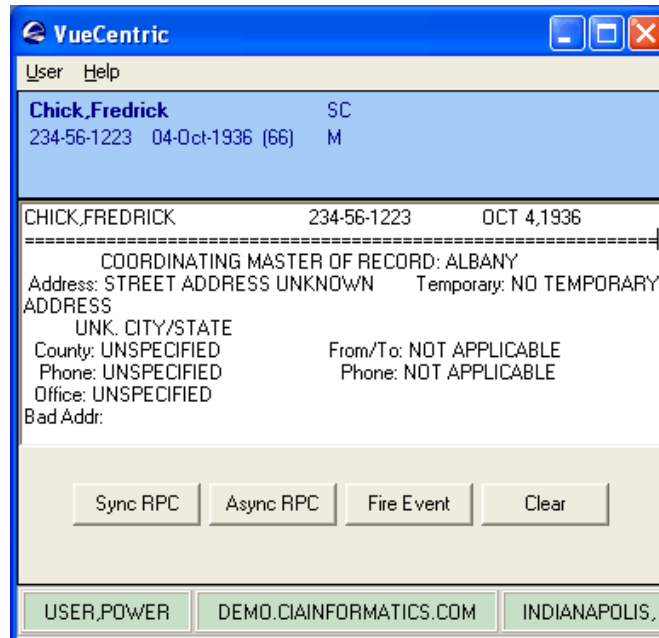


Figure 73-85: Text in Text Box Control

Now close the application (note: if you do not close the application now, you will receive an error the next time you try to compile the project because the control's executable image is locked).

### 73.16.2 Subscribing to Patient Context Changes

As we noticed, our component retrieves information based on the currently selected patient, but it does not respond when the patient selection changes. Let us fix this deficiency by having our component subscribe to patient context changes and modify the text box control's contents when the context change occurs. To do this, we need to add a callback interface to our component by modifying its class declaration:

```
public class DemoControlCS : System.Windows.Forms.UserControl, ICSS_Patient.ICSS_PatientEvents
{
    Press TAB to implement stubs for interface 'ICSS_Patient.ICSS_PatientEvents'
```

Figure 73-86: Added Code

Note that Visual Studio .NET prompts you to press the Tab key to implement method stubs when you add a new interface. This is a convenience feature that can be a real timesaver. Press the tab key now to generate these stubs. To view them, scroll to the bottom of the program code. You should see a collapsed region named "ICSS\_PatientEvents Members." Expand it to reveal the method stubs:

```
#region ICSS_PatientEvents Members

public void Committed()
{
    // TODO: Add DemoControlCS.Committed implementation
}

public string Pending(bool Silent)
{
    // TODO: Add DemoControlCS.Pending implementation
    return null;
}

public void Canceled()
{
    // TODO: Add DemoControlCS.Canceled implementation
}

#endregion
```

Figure 73-87: Expanded Region

We will add code to each of these stub entries to populate the text box control with text indicating that each method has been invoked. Complete the implementations with the code shown below:

```
#region ICSS_PatientEvents Members

public void Committed()
{
    textBox1.Text="Context change committed.";
}

public string Pending(bool Silent)
{
    textBox1.Clear();
    return "";
}

public void Canceled()
{
    textBox1.Text="Context change canceled.";
}

#endregion
```

Figure 73-88: Added Code

When a context change is initiated, the Pending method will be called first. In this method, we simply clear the text box control's contents. Because we are returning a null string for the Pending method, we are essentially voting YES to the context change. Assuming nothing else cancels the pending change, the Committed method is called next. In that method, we set the text box control's text to indicate that the context change was committed. Now compile the project by selecting **Build | Build**



**Solution** from the menu. Now restart the Visual Interface Manager, this time with the following command line options:

```
vim.exe /noupdate /trace /template=%DEMO
```

Once you login, you should see your component much as it looked before. Click the **Sync RPC** button to verify that this still works. Now click on the patient identification header and select a different patient. Notice how the memo control's contents have now changed. We have now responded to a context change event.

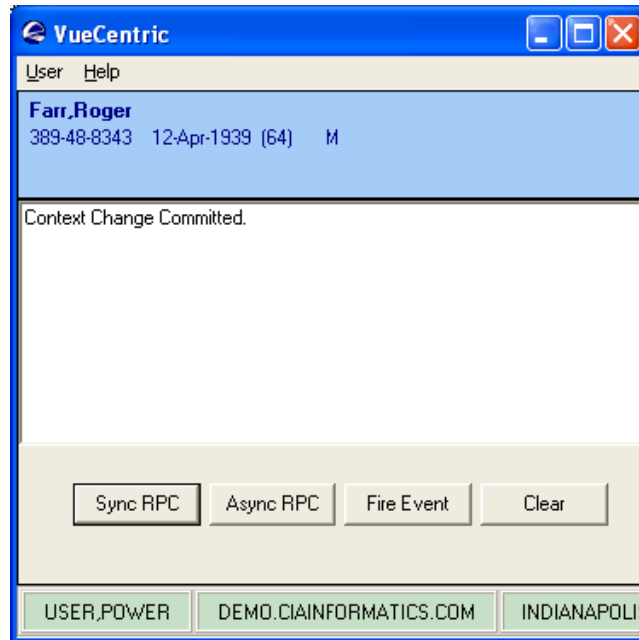


Figure 73-89: Changed Memo Control

### 73.16.3 Calling a Remote Procedure in Asynchronous Mode

Our next task is to support calling our remote procedure asynchronously. To do this, we need to implement the `ICSS_SessionEvents` interface that is defined in the `CSS` type library. This is done in an identical manner to the `ICSS_PatientEvents` interface we implemented in the previous section. First, add the interface to the component's class declaration:

```
, CSS_Patient.ICSS_PatientEvents, CIA_CSS.ICSS_SessionEvents
```

Figure 73-90: Added Code

Be sure the press the Tab key when prompted to create the method stubs:

```

#region ICSS_SessionEvents Members

public void RPCCallback(int Handle, string Data)
{
    // TODO: Add DemoControlCS.RPCCallback implementation
}

public void EventCallback(string EventType, string EventStub)
{
    // TODO: Add DemoControlCS.EventCallback implementation
}

public void RPCCallbackError(int Handle, int ErrorCode, string ErrorText)
{
    // TODO: Add DemoControlCS.RPCCallbackError implementation
}

#endregion

```

Figure 73-91: Creating Method Stubs

At this point, we will ignore the EventCallback method, but will return to it later. Let us add the following code to the other two methods:

```

public void RPCCallback(int Handle, string Data)
{
    rpcHandle=0;
    textBox1.Text=Data;
}

public void RPCCallbackError(int Handle, int ErrorCode, string ErrorText)
{
    rpcHandle=0;
    textBox1.Text="An error occurred: "+ErrorText;
}

```

Figure 73-92: Added Code

Here, we add the data returned by the asynchronous call to the text box control if it completed normally, or the text of the reported error if it did not.

Next, we will add code to the **Async RPC** button to call our remote procedure in asynchronous mode. To do this, double-click on that button in the form designer and complete the click event handler as shown:

```

private void button2_Click(object sender, System.EventArgs e)
{
    if(rpcHandle!=0)
        session.CallRPCAbort(rpcHandle);

    textBox1.Text="Asynchronous Remote Procedure Invoked.";
    rpcHandle=session.CallRPCAsync("BEHOPTCX PTINQ",patient.Handle,this,true);
}

```

Figure 73-93: Added Code

Note that we are first aborting any asynchronous remote procedure in progress before we call it again.

Now add an instance variable declaration to the component's class for the `rpcHandle` variable used to stored the returned handle:

```
private CIA_CSS.ICSS_Session session;
private CSS_Patient.ICSS_Patient patient;
private int rpcHandle;
private System.Windows.Forms.TextBox textBox1;
```

Figure 73-94: Added Code

Now, because we are already equipped to respond to patient context changes, let us add code to abort an asynchronous call in progress when a context change occurs. We will add this code to the `Committed` method:

```
public void Committed()
{
    textBox1.Text="Context change committed.";

    if(rpcHandle!=0)
    {
        session.CallRPCAbort(rpcHandle);
        rpcHandle=0;
    }
}
```

Figure 73-95: Added Code

Now it is time to test our component again. Recompile the project by selecting **Build | Build Solution**. Now restart the Visual Interface Manager as before. This time, click the **Async RPC** button. You should at first see the text "Asynchronous Remote Procedure Invoked" in the memo control. After a small delay, you should see the results of the remote procedure appear. Note that TaskMan must be running to invoke a remote procedure asynchronously, so if you do not receive data from the call, make certain TaskMan is running.

#### 73.16.4 Firing an Event

Let us now add the capability of firing an event. We are going to fire a local event called 'STATUS'. The Visual Interface Manager subscribes to this event and displays the data associated with it in its status bar. Double-click on the **Fire Event** button in the form designer and complete the click event handler as follows:

```
private void button3_Click(object sender, System.EventArgs e)
{
    session.EventFireLocal("STATUS","Status event fired by DemoControl.");
}
```

Figure 73-96: Added Code

Recompile the project and test in the Visual Interface Manager as before. Click the **Fire Event** button and you should see the event data appear in the status bar:



Figure 73-97: Revised Status Bar

### 73.16.5 Subscribing and Responding to an Event

Finally, we will enable our component to respond to STATUS events. This requires two steps. First we must implement the callback interface for responding to events. Because this is the same interface (ICSS\_SessionEvents) we implemented earlier for responding to asynchronous remote procedures, we have already done this. All we need to do is to fill in the implementation for the EventCallback method. Find this method in your component's implementation section and complete as follows:

```
public void EventCallback(string EventType, string EventStub)
{
    textBox1.Text=EventType+": "+EventStub;
}
```

Figure 73-98: Added Code

This will display the event name and data in the text box control.

Next, we need to subscribe to the STATUS event. To do this, return to the Load event handler and add the following line of code:

```
private void DemoControlCS_Load(object sender, System.EventArgs e)
{
    session=new CIA_CSS.CSS_ServerClass().Session;
    patient=session.FindServiceByProgID("CSS_PATIENT.PATIENT") as CSS_Patient.ICSS_Patient;
    session.EventSubscribe("STATUS",this);
}
```

Figure 73-99: Added Code

Now recompile and test your component as before. Now, clicking the **Fire Event** button also displays the event data in the memo control:

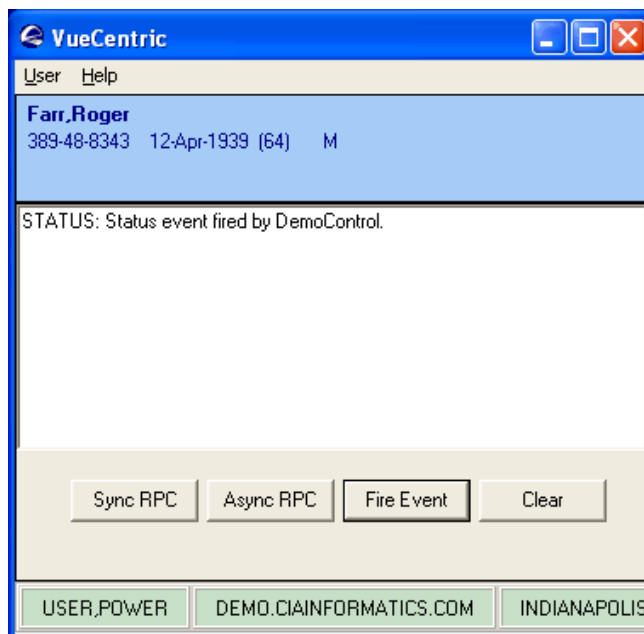


Figure 73-100: Revised Memo Control

### 73.16.6 Summary

You have learned how to create a .NET Windows Control, expose it as an ActiveX object, reference the Session and Patient Context objects, call remote procedures synchronously and asynchronously, and fire and receive events. You should now be well prepared to create components on your own.

## 73.17 Creating Services

Services, like visual components, are COM objects. Unlike visual components, they are not ActiveX controls, they cannot be manipulated at design time nor do they manifest themselves visually in their baseline state. Despite these differences, the programming techniques are very similar.

In the Delphi and Visual Basic examples that follow, we will be creating a simple service that implements a function to return information about the remote host. We will modify the visual component we created in the previous tutorial to use this service.

## 73.18 Creating Services with Delphi

This tutorial will demonstrate how to use Delphi to create a simple service object that implements a single method and how to access that service from within another component. Delphi offers several project types that can be used to create COM objects. We will create an automation-compatible COM object, which will afford the most flexibility for use in different settings.

### 73.18.1 Creating the Project

Creating a project to produce an automation-compatible COM object requires two steps. First we will create an ActiveX Library project by selecting **File | New | Other...** from the main menu. From the dialog that appears, select the ActiveX tab as shown below.

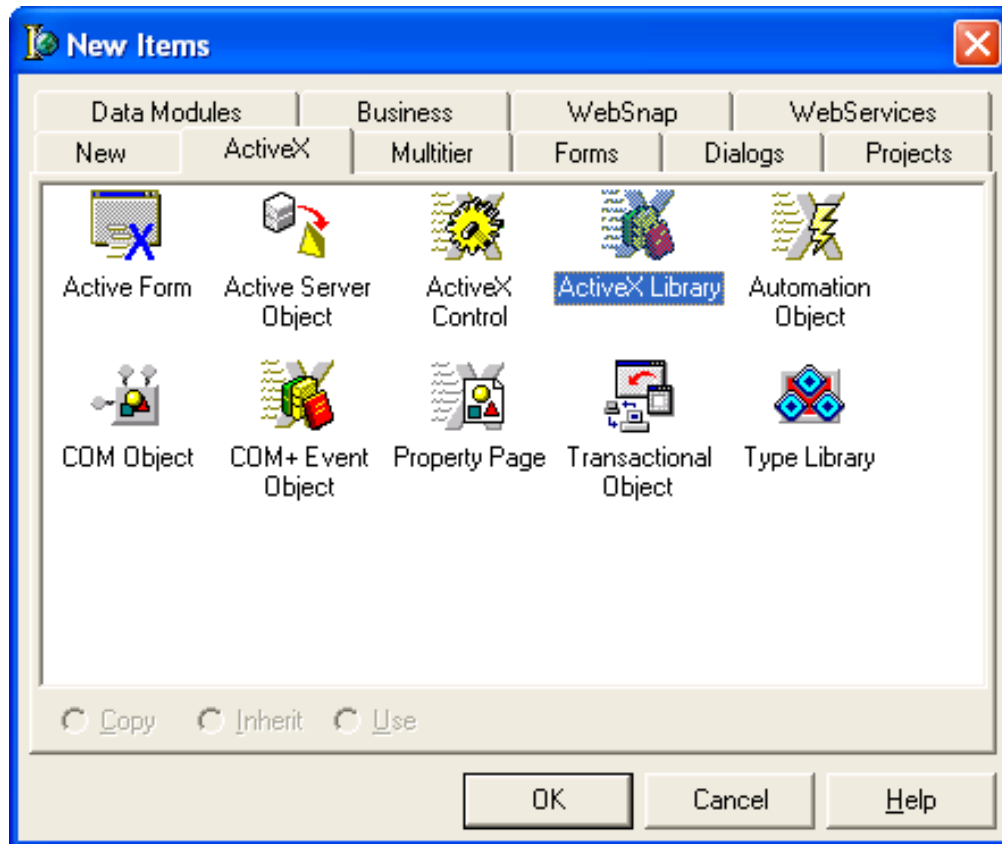


Figure 73-101: Selecting ActiveX Tab

Select **ActiveX Library** and click **OK**.

### 73.18.2 Creating the Service Object

Next, we need to add a COM object that will be our service to our newly created project. To do this, return to the project type dialog by selecting **File | New | Other...**, select the ActiveX tab again, but this time select **Automation Object** and click **OK**. In the Automation Object Wizard that appears, fill in the CoClass Name as shown below. Leave the other settings as they are.

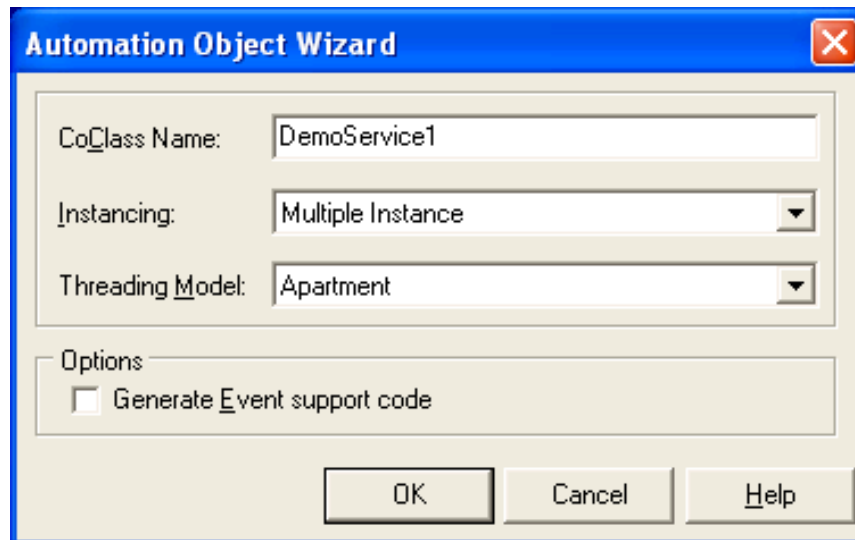


Figure 73-102: Automation Object Wizard Entries

Click OK and you should see the type library editor appear with our newly created type library and its automation object, DemoService1, with its default interface, IDemoService1:

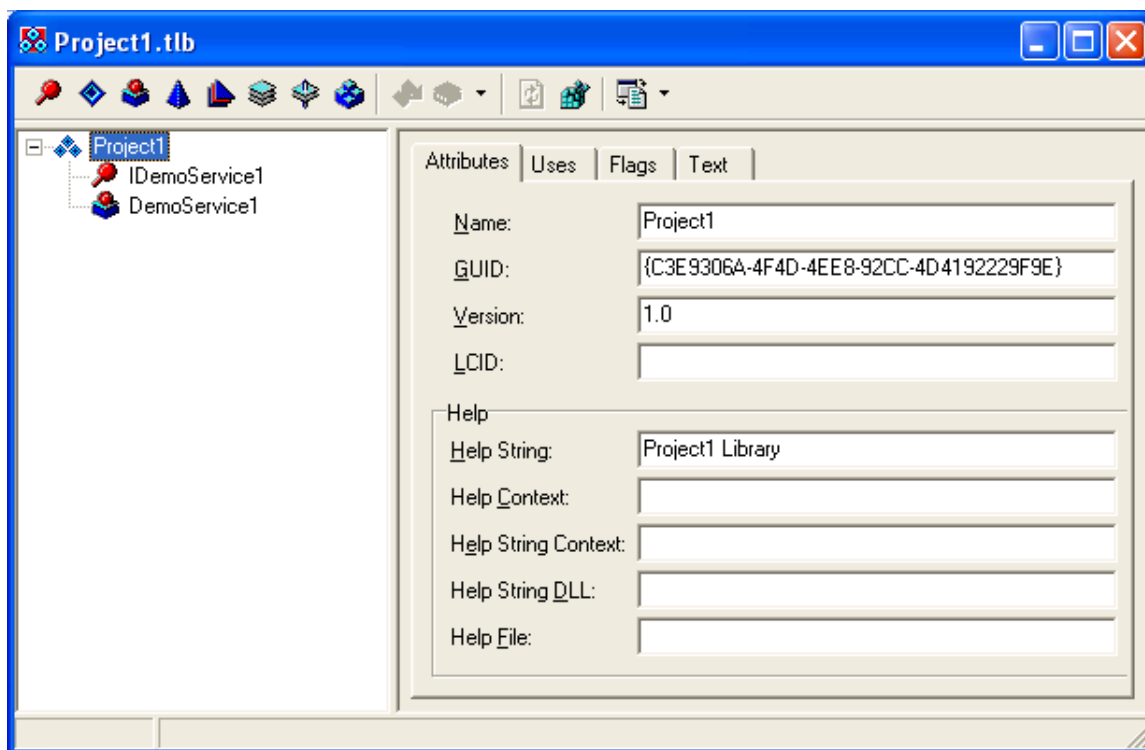



Figure 73-103: Sample Library Editor

Select the top node labeled Project1 and rename the type library to DemoService by changing the entry in the **Name** edit box in the right pane. Also change the entry in the **Help String** edit box to DemoService as well. Click the  button in the toolbar to synchronize your program code with the change. Now save the project to a folder of

your choice, naming the project file DemoService1 (this will give us an executable filename of DemoService1.dll when we are done).

### 73.18.3 Accessing the Session Object

Because we will be making a remote procedure call, we need to access the Session object. This is done in the exact same manner as we did with the visual component we created earlier. Because we have already imported the type library for the CSS, we do not have to repeat this step (if you did not do this in the tutorial on creating visual components, review the section [“Accessing the Session Object” on page 493](#) to see how this is done). We do need to add a reference to the CSS type declaration unit to the uses clause of the unit containing the Session object. In the code editor, select the Unit1 unit and modify the uses clause as shown:

```
uses
  ComObj, ActiveX, Project1_TLB, StdVcl, CIA_CSS_TLB;
```

Figure 73-104: Added Code

Now create a private section for the service object’s class and add a declaration for the variable that will hold a reference to the Session object:

```
type
  TDemoService1 = class(TAutoObject, IDemoService1)
  private
    FSession: ICSS_Session;
  protected
    ( Protected declarations )
  end;
```

Figure 73-105: Added Code

As with our visual component created earlier, we will initialize the FSession variable in the Initialize method. Because the Automation Object Wizard doesn’t automatically generate this for us, we will need to add it manually. Create a public section and add an override declaration for the Initialize method:

```
type
  TDemoService1 = class(TAutoObject, IDemoService1)
  private
    FSession: ICSS_Session;
  protected
    ( Protected declarations )
  public
    procedure Initialize; override;
  end;
```

Figure 73-106: Added Code



To generate an implementation for the Initialize method, we will use a Delphi shortcut. Right-click on the service object's class name (TDemoService1) and select **Complete class at cursor** from the popup menu. This will produce a default implementation for the Initialize method:

```
procedure TDemoService1.Initialize;
begin
    inherited;
end;
```

Figure 73-107: Default Implementation for Initialize Method


Now add the code to initialize the FSession variable:


```
procedure TDemoService1.Initialize;
begin
    inherited;
    FSession := CoCSS Server.Create.Session;
end;
```

Figure 73-108: Added Code

#### 73.18.4 Modifying the Interface

Now we want to add a method to the interface of our service object. This will be a function that will receive one argument that specifies the type of information requested and will return the requested information as a string. To modify the interface, we will use the type library editor. If it is not already visible, make it visible by selecting **View | Type Library** from the main menu. Select the node labeled IDemoService1. This corresponds to the default interface for the service object. It currently has no methods

or properties associated with it. To create a method, click the  button on the toolbar. This will generate a method with the default name of Method1. Rename the method to GetInfo by either changing the label associated with the node or by changing the Name edit box in the right pane. Now switch to the Parameters tab. From the **Return Type** drop-down box, select either WideString or BSTR (the list content depends upon whether you have configured the editor to display Delphi or C syntax). We also need to add the parameter that will contain the type of information we are requesting. At the bottom of the Parameters tab, click **Add**. This will create a parameter with a default name of Param1 and a datatype of Integer. Change the name of the parameter to InfoType by selecting the name in its cell and modifying. Leave the datatype as it is.

Now click the  button on the toolbar to synchronize your program code with these changes. At this point, the type library editor should look like this:

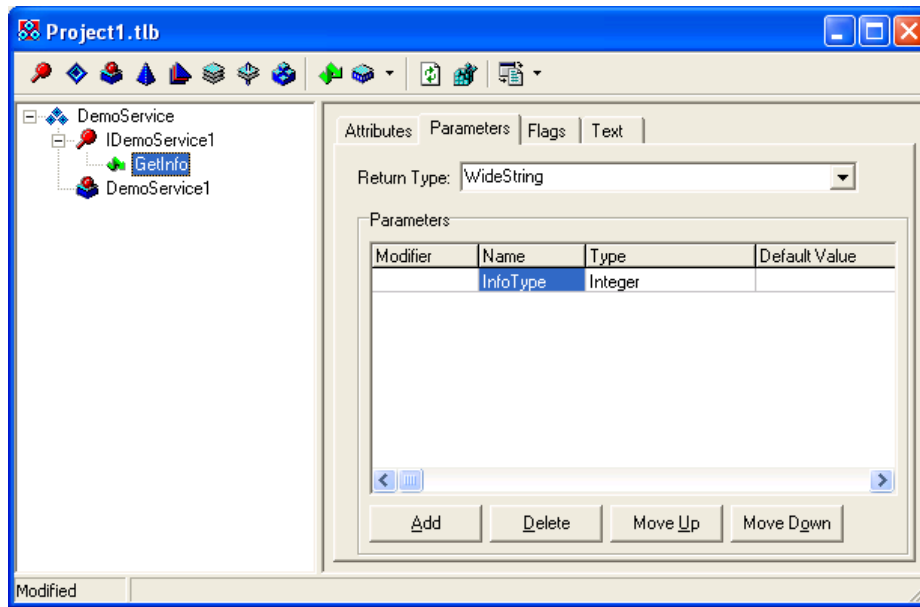


Figure 73-109: Type Library Editor

Your program code should now contain a new method:

```

type
  TDemoService1 = class(TAutoObject, IDemoService1)
  private
    FSession: ICSS_Session;
  protected
    function GetInfo(InfoType: Integer): WideString; safecall;
    ( Protected declarations )
  public
    procedure Initialize; override;
  end;

```

Figure 73-110: Added Code

### 73.18.5 Providing the Implementation

Now that we have created our method, we need to provide its implementation. Our method will perform a synchronous remote procedure call, passing its single parameter to indicate the type of information we are requesting. It will return the result of the call in its return value. To do this, complete the implementation for the GetInfo method as shown:

```

function TDemoService1.GetInfo(InfoType: Integer): WideString;
begin
  Result := FSession.CallRPCText('CIANBRPC GETINFO', InfoType);
end;

```

Figure 73-111: Added Code

### 73.18.6 Registering the Service

Now we need to register the service. First, let us compile the project by selecting **Project | Build DemoService1** from the menu. It should compile without errors. Now, register the type library by selecting **Run | Register ActiveX Server** from the menu. You should receive confirmation of successful registration.

Next, we need to register our service to the **VueCentric®** Framework. This is done in the same way we registered our visual component - using the **VueCentric®** System Management Utility. Run the tool and login to the remote host. Select the **Object Registry** tab and in the **Restrict List To** box, check **Local Registry** (this allows us to see local COM objects that aren't yet registered to the Framework). The list of objects will refresh. Now search the list for the programmatic identifier of our newly created service, **DemoService.DemoService1**, and select that entry. In the upper right pane, click **Copy** to copy the local COM registration information into the **VueCentric Settings** pane. In that pane, fill in the **Name** field with the display name for the control. The choice of name is arbitrary. We'll call it "Demo Service 1." Now switch to the **special settings** tab and check the box labeled **Service**. Now click **Apply** at the bottom. This completes the Framework registration process.

Finally, let's import the service's type library so that it is available to be used by other projects. Select **Project | Import Type Library...** from the main menu. Find the entry "DemoService (Version 1.0)" and select it. Make sure **Create Component Wrapper** is unchecked and click **Create Unit**. You have now created a type declaration unit in the **Delphi Imports** folder (if you receive a message that the unit is already in the project, just ignore it and continue).

### 73.18.7 Accessing the Service

Next, we need to provide a means to test the **GetInfo** method of our service object. To do this, we will modify the **DemoControl** we created in the earlier tutorial. Open the **DemoControl** project now (be sure to save any changes to the existing one). Modify the **uses** clause in the **DemoControl1** unit to add a reference to our service's type library declaration unit.

```
controls, Forms, Dialogs,  
, CIA_CSS_TLB, CSS_Patient_TLB, DemoService_TLB;
```

Figure 73-112: Added Code

Next, add a declaration for the variable that is to hold the reference to our service in the **private** section of our **TDemoControlX** class.

```
private
  { Private declarations }
  FService: IDemoService1;
  FHandle: Integer;
  FPatient: ICSS_Patient;
```

Figure 73-113: Added Code

Add the code to initialize the FService variable to the Initialize method:

```
procedure TDemoControlX.Initialize;
begin
  inherited Initialize;
  OnActivate := ActivateEvent;
  OnClick := ClickEvent;
  OnCreate := CreateEvent;
  OnDbClick := DbClickEvent;
  OnDeactivate := DeactivateEvent;
  OnDestroy := DestroyEvent;
  OnKeyPress := KeyPressEvent;
  OnPaint := PaintEvent;
  FSession := CoCSS_Server.Create.Session;
  FPatient := FSession.FindServiceByCLSID(CLASS_Patient) as ICSS_Patient;
  FSession.EventSubscribe('STATUS',self);
  FService := FSession.FindServiceByProgID('DemoService.DemoService1') as IDemoService1;
end;
```

Figure 73-114: Added Code

Note that we are using the FindServiceByProgID here. We could just as easily have used the FindServiceByCLSID and passed the GUID for the service object.

Finally, add a fifth button to the form and position it as desired. Rename the caption to “Demo Service” and set its Anchors property to [akLeft,akBottom]. Double-click the button and complete its Click event handler as shown:

```
procedure TDemoControlX.Button5Click(Sender: TObject);
begin
  Memo1.Lines.Text := FService.GetInfo(2);
end;
```

Figure 73-115: Added Code

Compile the project by selecting Project | Build DemoControl and load it in the Visual Interface Manager as before. Now click the **Demo Service** button. You should see something similar to this:

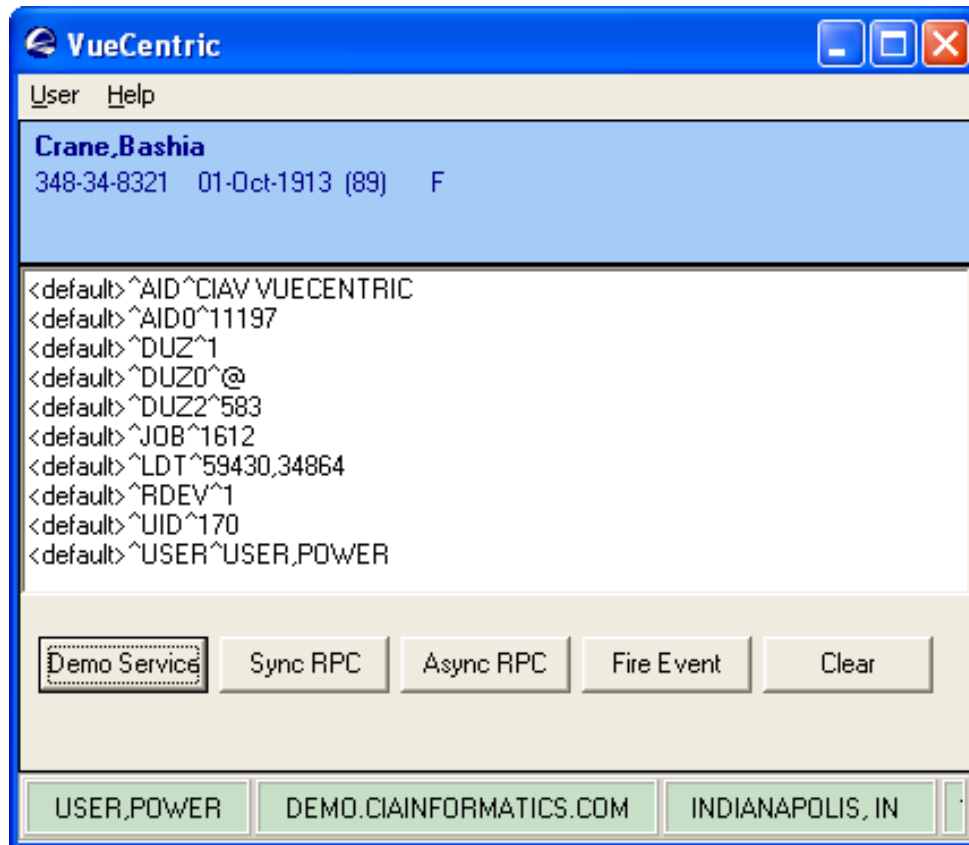


Figure 73-116: Revised Screen

### 73.18.8 Summary

You have learned how to create a service object and access that service within another component. The techniques for performing other programming tasks, such as making synchronous and asynchronous remote procedure calls and firing and receiving events, are identical to those used in creating visual components.

## 73.19 Creating Services with Visual Basic

This tutorial will demonstrate how to use Visual Basic to create a simple service object that implements a single method and how to access that service from within another component.

### 73.19.1 Creating the Project

To create the project for our service component, select **File | New Project** from Visual Basic's main menu. Select the **ActiveX DLL** project type and click **Open**.

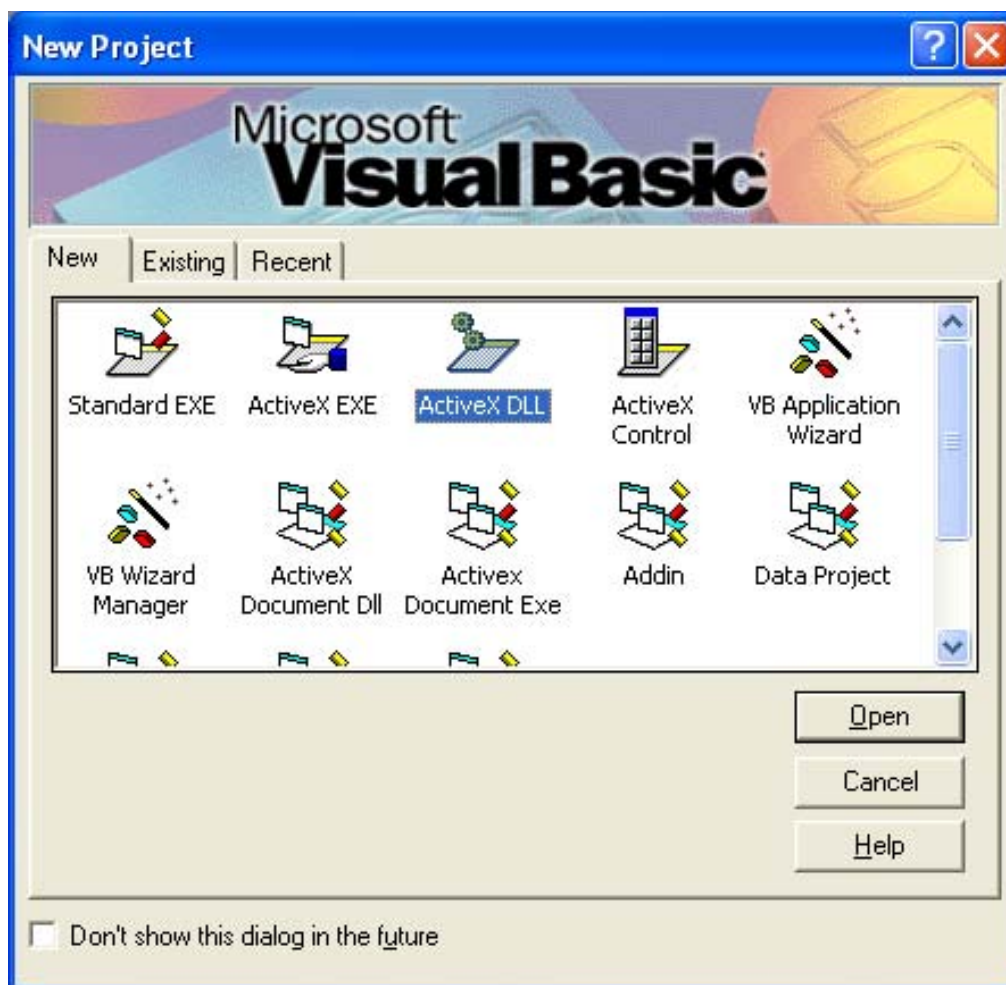


Figure 73-117: New Project Dialog

This creates a project containing a single automation-compatible COM object named Class1 by default. Before continuing, let us rename our project and object to better reflect their function. In the project pane, select the project node and change its name in the property pane from Project1 to DemoService. Returning to the project pane, select the object node labeled Class1 and change its name in the property pane to DemoService2. This will provide a programmatic identifier for our service object of DemoService.DemoService2. Now save the project to a folder of your choice.

### 73.19.2 Accessing the Session Object

Because we will be making a remote procedure call, we need to access the Session object. This is done in the exact same manner as we did with the visual component we created earlier. To do this, select **Project | References...** from the main menu, locate and check the entry “CIA Component Support Services,” and click **OK** to close the dialog.

Next, we need to declare a global variable to hold our reference to the Session Object. To do this, select (General) in the code editor and add the following code:

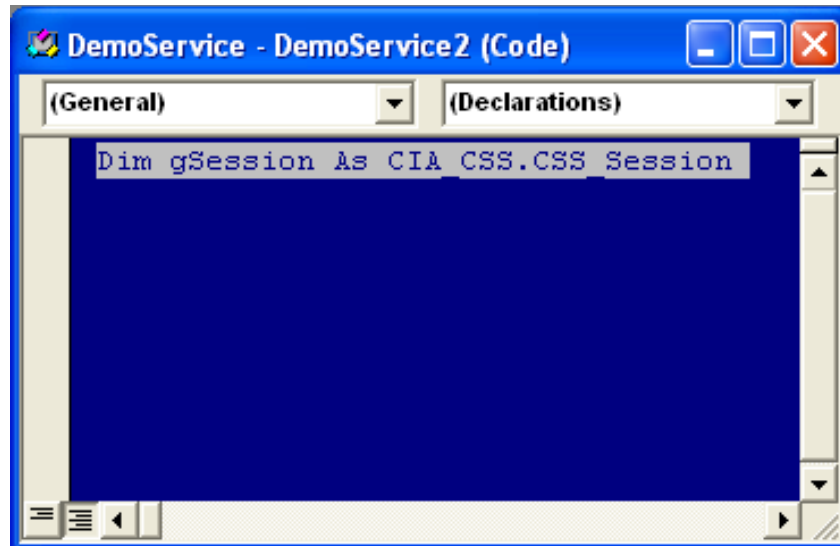


Figure 73-118: Added Code

Now we need to store a reference to the Session object in our global variable, gSession. Select the Class entry in the left drop-down box of the code editor and its Initialize method in the right drop-down box. This will create an empty implementation for the Initialize method. Next, complete the implementation by adding the following code:

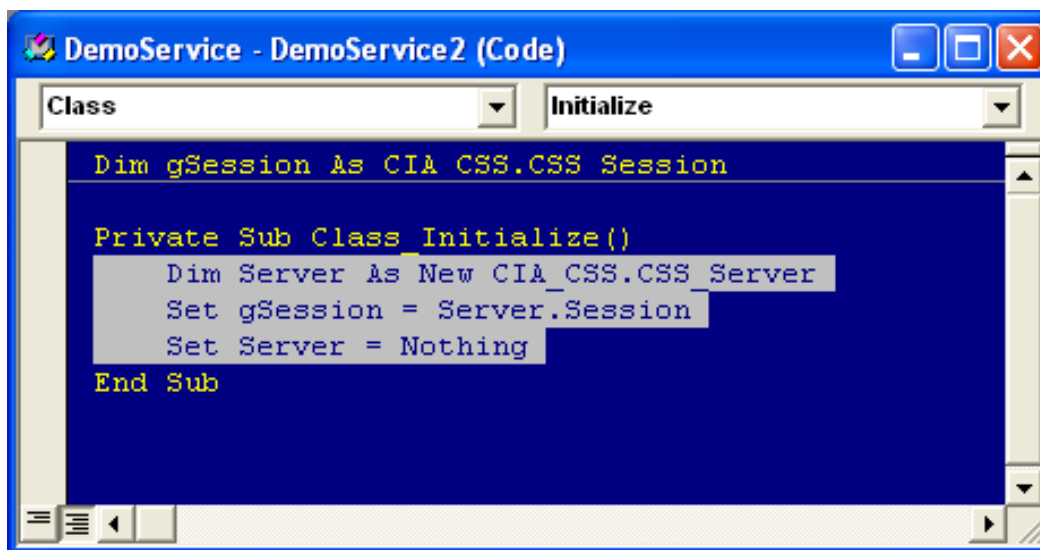
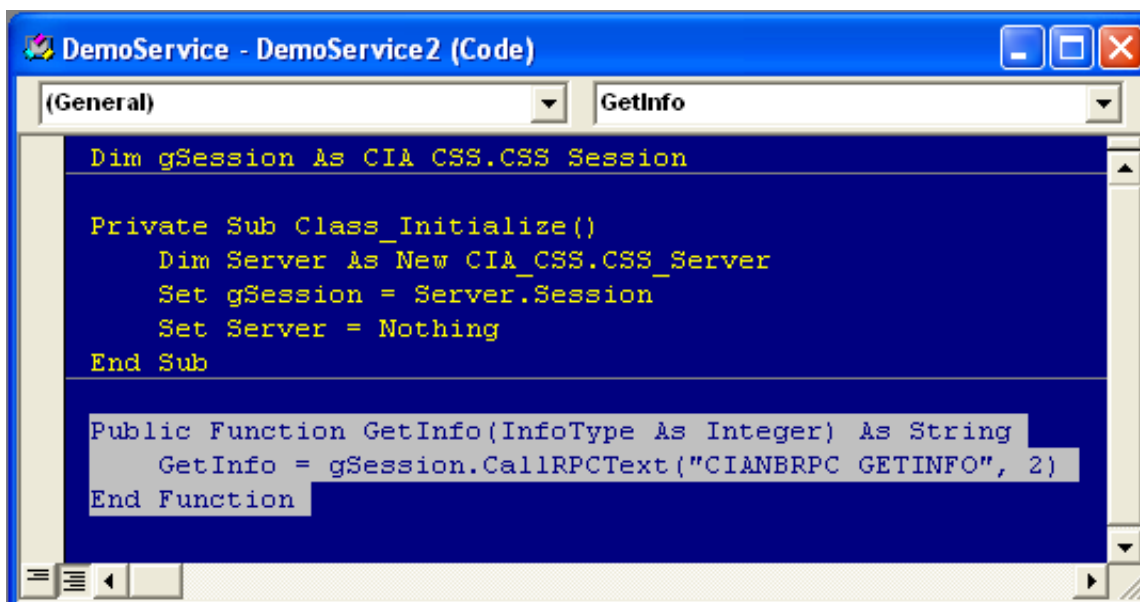


Figure 73-119: Added Code

This code obtains a temporary reference to the Server object, retrieves a reference to its Session object, and releases the Server object. At this point, you now have a reference to the Session object stored in the global variable, gSession.

### 73.19.3 Modifying the Interface

Now we want to add a method to the interface of our service object. This will be a function that will receive one argument that specifies the type of information requested and will return the requested information as a string. Return to the code editor and add the following public function at the end as shown:



```
Dim gSession As CIA.CSS.CSS.Session

Private Sub Class_Initialize()
    Dim Server As New CIA.CSS.CSS.Server
    Set gSession = Server.Session
    Set Server = Nothing
End Sub

Public Function GetInfo(InfoType As Integer) As String
    GetInfo = gSession.CallRPC("CIANRPC GETINFO", 2)
End Function
```

Figure 73-120: Added Code

### 73.19.4 Registering the Service

Now we need to register the service. First, let us compile the project by selecting **File | Make DemoService.dll** from the menu. It should compile without errors. Next, we need to register our service to the **VueCentric®** Framework. This is done in the same way we registered our visual component - using the **VueCentric®** System Management Utility. Run the tool and login to the remote host. Select the **Object Registry** tab and in the **Restrict List To** box, check **Local Registry** (this allows us to see local COM objects that aren't yet registered to the Framework). The list of objects will refresh. Now search the list for the programmatic identifier of our newly created service, **DemoService.DemoService2**, and select that entry. In the upper right pane, click **Copy** to copy the local COM registration information into the **VueCentric Settings** pane. In that pane, fill in the **Name** field with the display name for the control. The choice of name is arbitrary. We'll call it "Demo Service 2." Now switch to the **special settings** tab and check the box labeled **Service**. Now click **Apply** at the bottom. This completes the Framework registration process.

### 73.19.5 Accessing the Service

Now we need to provide a means to test the **GetInfo** method of our service object. To do this, we will modify the **DemoControl** we created in the earlier tutorial. Open the **DemoControl** project now (be sure to save any changes to the existing one). Add a



reference to our service object by selecting **Project | References...** from the main menu. Find the entry “DemoService” in the list, check it, and click **OK** to close the dialog.

Next, create a global variable to hold the reference to our service object in the (General) section as shown:

```
Dim gService As DemoService2
Dim gHandle As Integer
Dim gPatient As CSS_Patient.Patient
Dim gSession As CIA_CSS.CSS Session
```

Figure 73-121: Added Code

Now initialize the gService variable in the Initialize method of the UserControl:

```
Private Sub UserControl_Initialize()
    Dim Server As New CIA_CSS.CSS_Server
    Set gSession = Server.Session
    Set Server = Nothing
    Set gPatient = gSession.FindServiceByProgID("CSS_PATIENT.PATIENT")
    gSession.EventSubscribe "STATUS", Me
    Set gService = gSession.FindServiceByProgID("DemoService.DemoService2")
End Sub
```

Figure 73-122: Added Code

Finally, add a fifth button to the form and position it as desired. Rename the caption to “Demo Service.” Double-click the button and complete its Click event handler as shown:

```
Private Sub Command5_Click()
    Text1.Text = gService.GetInfo(2)
End Sub
```

Figure 73-123: Added Code

Compile the project by selecting **File | Make DemoControl.ocx...** and load it in the Visual Interface Manager as before. Now click the **Demo Service** button. You should see something similar to this:

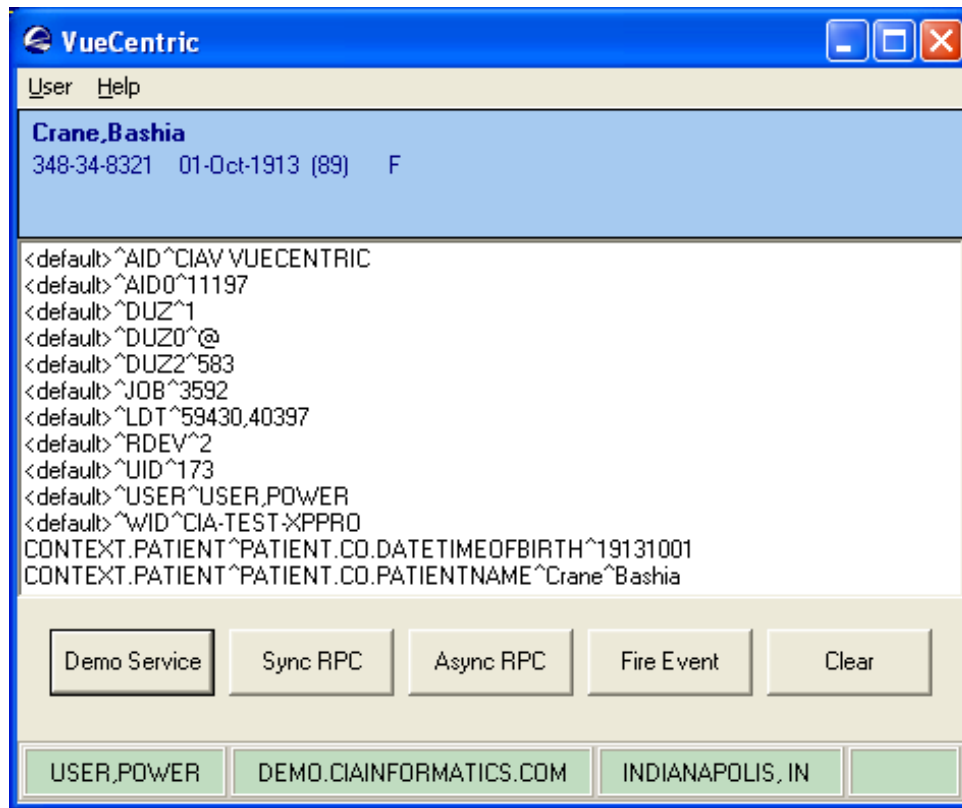


Figure 73-124: Revised Dialog

### 73.19.6 Summary

You have learned how to create a service object and access that service within another component. The techniques for performing other programming tasks, such as making synchronous and asynchronous remote procedure calls and firing and receiving events, are identical to those used in creating visual components.

## 73.20 Deploying Components

Successful deployment of a component relies on a thorough understanding of version control and dependency control as implemented by the VueCentric® Framework. In addition, the VueCentric® SDK can greatly facilitate the task of creating server-side builds for your components.

### 73.21 Version Control

Proper version control is essential to ensuring that your components are updated properly and that the correct version of your component is loaded at run-time. The VueCentric® Framework (specifically, the CMS) automatically recognizes that a newer version of a requested component is available and retrieves and installs it (so-called, just-in-time update). For this to work properly, a basic understanding of how this is done is necessary.

### 73.21.1 Version Numbers

Version numbers consist of up to four numbers separated by periods. These numbers represent, from left to right, the major, minor, release, and build versions. These numbers have a hierarchical relationship, so whenever a number is incremented, all numbers below it (i.e., to the right) should be reset to 0. Build numbers should be incremented every time a component is recompiled (most compilers can be configured to do this automatically). The guidelines for when to increment the major, minor, and release numbers are less clear. We typically increment the release number whenever we add or change functionality, but do not break compatibility with a previous version. We increment the minor version number when we break compatibility with a previous version. We increment the major version number when there is a major change to the functionality.

### 73.21.2 Which Version?

COM allows associating version numbers with the type library and each imbedded interface and object. While this has certain advantages, COM makes no provision for supporting multiple versions of an object on a given machine. In addition, COM versioning has no applicability to files that are not COM objects. Therefore, a version control mechanism independent of COM is necessary. The CMS uses two different mechanisms for file versioning. For binary files that have an imbedded version resource, the CMS uses this information to determine the file version. Version resources are a standard means for imbedding version information within binary files. You can examine this resource in Windows by viewing a binary file's properties. If the binary file has a version resource, you will see a tab labeled Version and on that tab you will find the file version. Most compilers can be configured to include version information.

For non-binary files, the CMS uses the file's modification timestamp to generate a pseudo-version number. Like standard version numbers, this version number consists of four hierarchically arranged numbers. From left to right, these numbers are the year, month, day, and time. Because the timestamp uses universal time format, the version number generated in this manner is not sensitive to varying time zones.

### 73.21.3 Registering Version Information

When a component is requested, the CMS searches the application directory of the local machine to determine the version of the component currently residing there. It does this by directly examining the component's imbedded version resource. It then compares the local version number to the version number specified for the component in its VUECENTRIC OBJECT REGISTRY file entry. If the local version number is less than the one specified in the VUECENTRIC OBJECT REGISTRY file, or if the component was not found in the application directory, it is retrieved from the object repository and installed in the application directory. Note that the CMS does not directly examine the file in the object repository to determine its version. This means that the version entered in the VUECENTRIC OBJECT REGISTRY file must truly

reflect the version residing in the object repository for the update mechanism to work. The reason the CMS does not directly examine files in the object repository is twofold. First, if the repository resides on a remote share, examining a file's version resource can be very time consuming (in fact, Windows makes a local copy of the file to do this). Second, if the repository resides on a web or ftp server, there is no means to examine the file directly without downloading it first. Therefore, be certain that you correctly update the version information for your components.

#### 73.21.4 What is Side-by-Side Versioning?

Side-by-side versioning refers to the ability to have multiple versions of the same component residing on a machine at the same time. When most COM objects are registered, the full path to the executable is included in the Windows registry. Because this registration overwrites any previous entry, it is not possible to register more than one version at a time. Thus, the default behavior of COM is to share a single copy of an object across all applications. This is rarely desirable, especially in the situation where an object is not backward-compatible with previous versions. To overcome this limitation, Microsoft has offered three possible solutions:

- Store only the object's file name, not its path information, in the Windows registry. In the absence of path information, Windows will search for the file in the application directory first, then in the System and Windows directories. In this manner, one can partition different object versions in the respective application directories and be certain that the correct one is loaded. Modifying the default COM registration behavior can be done in one of two ways. Because COM objects register themselves through the `DLLRegisterServer` method, one can override the default implementation of this method and modify the registration process. A second option is to set the `SIDE-BY-SIDE` field of the corresponding `VUECENTRIC OBJECT REGISTRY` file entry to true. When this field is true, the CMS inspects the Windows registry entry for the COM object and removes path information if it exists. This allows forcing side-by-side versioning on an object-by-object basis.
- Microsoft offers a second solution to the versioning problem. If the COM subsystem detects a file with the same name as the main application with `“.local”` appended to it, it will ignore all path information in the Windows registry. For example, if the file `“VIM.exe.local”` is placed in the application directory, all objects loaded by the VIM application will be treated as if they had been registered without path information. This option has three disadvantages. First, it is all-or-nothing as far as the application is concerned. Second, this capability only applies to versions of Windows including and subsequent to Windows 98 SE. Third, in the presence of an application manifest (see below), this feature is disabled.
- The third, and most recent, solution to versioning is in the form of an application manifest, which can either be imbedded in the application or present in the form of a file with the same name as the application with a `“.manifest”` appended to it. The purpose of the manifest is to allow the application devel-

oper to specify which version of a particular component is to be used. In contrast to the “.local” file described above which affects the search behavior for all components, the manifest affects search behavior only for those components listed. Furthermore, the use of a manifest precludes the use of the “.local” file (except in Windows 2000 environments where the manifest is not supported). The EHR uses a manifest to enable theme support for the application.

The choice of versioning techniques depends on many factors. Because of the multiple run-time environments supported by the EHR, all three techniques are utilized to ensure the correct components are used for a given application instance.

For a detailed treatment of this subject, see the MSDN article entitled “*Implementing Side-by-Side Component Sharing in Applications*” (<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnsetup/html/sidebyside.asp>”).

### 73.21.5 Imbedding Version Information

Both Delphi and Visual Basic can imbed version information in the binary files they generate.

To include version information in Delphi projects, choose the Project|Options menu. From the Project Options dialog that appears, select the Version Info tab. Check the boxes labeled “Include version information in project” and “Auto-increment build number.” You can optionally fill in other information located at the bottom of the dialog.

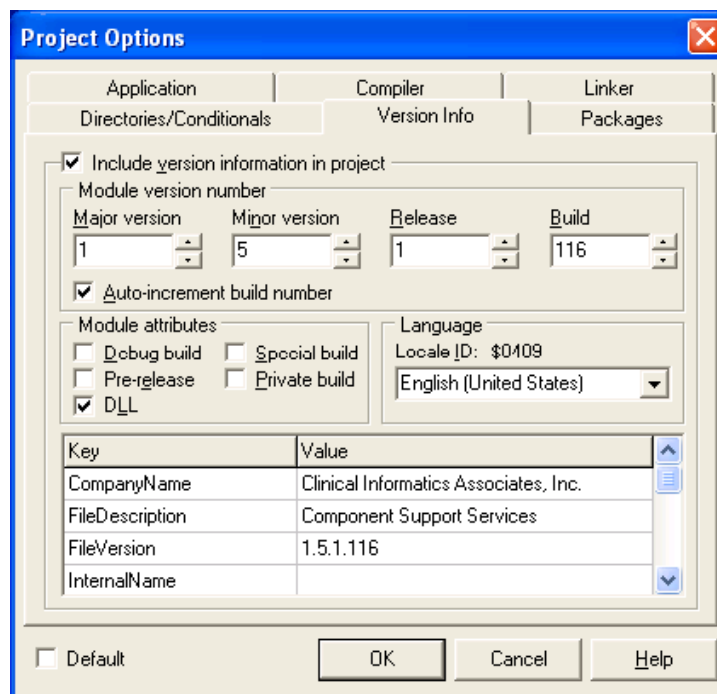


Figure 73-125: Checkboxes to Check

To imbed version information using Visual Basic, select the **Project|Properties** menu. From the Project Properties dialog that appears, select the **Make** tab. Check the box labeled “Auto Increment.” You can optionally fill in other version information as desired.

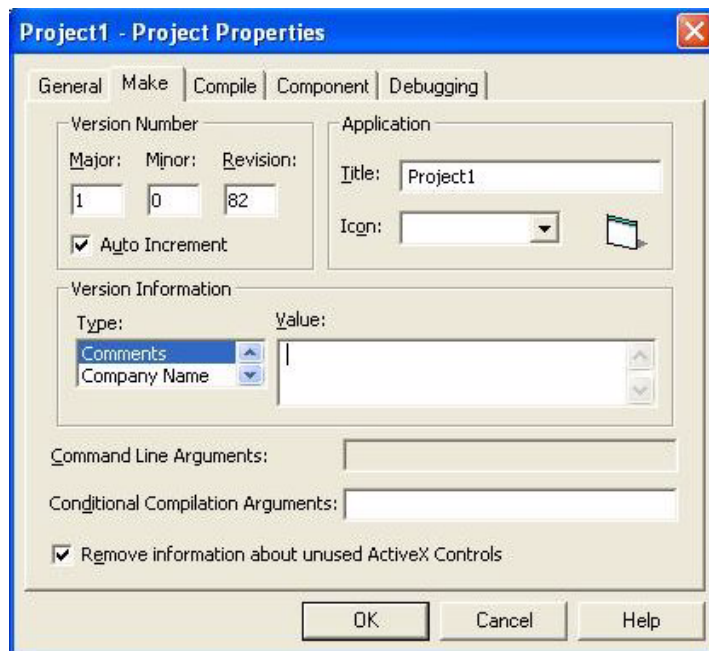


Figure 73-126: Checkbox to Check

## 73.22 Handling Dependencies

It is not at all uncommon that a component depends upon the presence of other components or files to function. So how does one insure that all of the necessary pieces are in place when a component is executed? There are several possible approaches to this problem.

If a given file is required by many components, it might make sense to deploy that file using the same mechanism to deploy the core Framework. VueCentric<sup>®</sup> provides a configurable installer utility that installs and updates core components from a shared directory, independent of the Framework. Alternatively, if the core files are installed using a third-party installer, one could add the required file(s) to the installation.

The VUECENTRIC OBJECT REGISTRY file provides two options for handling dependent files. Using the VueCentric System Management Utility, one can register dependencies between entries in this file in the Dependencies section of the Object Registry tab. When an object is requested, all listed dependencies are requested automatically. This method has the advantage of applying version control over each dependency. In addition, if a dependent file is marked as a service, the service is automatically started. Dependencies are recursive so that any dependencies listed for a dependent file are also requested.

The Required Files section of the Object Registry tab provides an alternate means for declaring dependencies. Whenever an object is installed or updated, any required files listed here are also retrieved. This has the advantage of not requiring that these dependent files be entered into the VUECENTRIC OBJECT REGISTRY file. The main disadvantage is that no version control is applied to these files. They are only retrieved when the object needs to be installed or updated.

## 73.23 Generating KIDS Builds

All components require some kind of installation on the remote host. At a minimum, a component requires registration information to be entered into the VUECENTRIC OBJECT REGISTRY file. Most will also require supporting code and remote procedure declarations. Some will also define events and parameters. To facilitate the delivery of these required elements to the remote host, the VueCentric® SDK augments the KIDS system by providing a means to easily package these elements into the build and also provides support for common tasks such as registering remote procedures.

The VueCentric® SDK is delivered as a KIDS build. Once installed, the SDK provides a template for creating component builds and a configuration file for controlling the behavior of the build. The template is a KIDS build called VUECENTRIC DUMMY. Do not modify this build directly. Rather, create a copy of the build and modify it. This build is delivered with pre-installation, post-installation, and pre-transportation methods that should be modified. In addition, it is also configured to deliver entries from several key files based on information in the configuration file. You can add additional files to the build, but you should not remove the existing entries. You can also add any additional elements that are to be delivered (e.g., remote procedures, options, etc.).

The configuration file delivered with the SDK is called the VUECENTRIC DISTRIBUTION file. By associating an entry with your component build created from the VUECENTRIC DUMMY template, you can control the elements delivered with your build. The VUECENTRIC DISTRIBUTION file has the following fields:

Property	Datatype	Access	Description
NAME	.01	Text	This is typically the display name given to the component.
BUILD	.5	Pointer (#9.6)	This is the build with which this entry is to be associated.
OBJECT (multiple)	1	Pointer (#19930. 2)	Any VUECENTRIC OBJECT REGISTRY entries to be included in the build.
PARAMETER DEFINITION (multiple)	2	Pointer (#8989.5 1)	Any parameter definitions to be included in the build. The subfile also has fields for initializing parameter values.

Property	Datatype	Access	Description
PARAMETER TEMPLATE (multiple)	3	Pointer (#989.52)	Any parameter templates to be included in the build.
TEMPLATE (multiple)	4	Pointer (#19930.3)	Any VUECENTRIC TEMPLATE REGISTRY entries to be included in the build.
EVENT (multiple)	5	Pointer (#19941.21)	Any CIA EVENT TYPE entries to be included in the build.
PREINIT CODE	50	M Code	Any code to be executed during the pre-installation phase.
POSTINIT CODE	51	M Code	Any code to be executed during the post-installation phase.
PRETRANS CODE	52	M Code	Any additional pretransportation code to be executed when generating the build.
COMMENTS	99	Word Processing	Used for documentation purposes.

When you generate your build, you will see the message “Target distribution: <name>” where <name> is the name of the associated entry in the VUECENTRIC DISTRIBUTION file. If more than one entry is associated with the build, you will be prompted to select the one to use.

## 73.24 Pitfalls and Special Techniques

This section discusses potential pitfalls that the component developer can encounter and special techniques that will facilitate component development.

### 73.24.1 Component Initialization

Both Delphi and Visual Basic define an Initialize method for its ActiveX controls. This method is executed when an object is first created and should be used to perform any necessary initializations. In Delphi, this is preferred over overriding the object’s constructor because ActiveX objects typically have more than one constructor and the Initialize method is guaranteed to be executed regardless of which constructor is invoked.

### 73.24.2 Component Destruction

Under Delphi, accessing a component’s COM interface within the destructor can cause a stack overflow. This happens because the destructor has been invoked because the object’s reference count has reached zero. When the object’s COM interface is referenced, the reference count increments and then decrements back to zero, causing re-entry of the destructor. If there is a need to access an object’s COM interface in its destructor, first make a call to the object’s `_AddRef` method. This increments the



reference count and insures that it will not return to zero. Do not call the `_Release` method because the object is already in the process of being released and doing so would cause the reference count to return to zero.

### 73.24.3 Other Containers

Because visual components are ActiveX-compliant, any visual component can potentially be hosted in any ActiveX-compliant container (e.g., Internet Explorer). Any component used in this manner should register itself with the CSS by calling the Session object's `RegisterObject` method. If this is not done, the component will not be notified of context changes nor will it be accessible to other components.

### 73.24.4 Focus Issues

ActiveX controls created with Visual Basic have one anomaly associated with control focus. The first mouse click on a Visual Basic ActiveX control sets focus to the control. Once the control has focus, subsequent mouse clicks perform as expected. This means, for example, that if you click on a button on a Visual Basic ActiveX control that does not yet have focus, the click will not generate a button press event. Subsequent clicks will.

One solution to this anomaly is to implement the `MouseMove` method for the `UserControl` and make a call to `SetFocus` in the implementation. This causes the control to automatically receive focus when the mouse moves over it.

### 73.24.5 Deferring Data Fetches

Retrieving data from the remote host can be time consuming and hamper application performance. This burden can be lessened by devising intelligent strategies for deferring data fetches until the data is actually needed. One common strategy is for a component to defer populating its display until it becomes visible. For example, it makes little sense to retrieve a patient's problem list data into a component immediately after a patient context change if the user never views that component. A useful technique for deferring data fetches in this scenario would be to set a flag indicating that a data fetch is required when the context change occurs and then invalidate the component's main window. In the component's painting logic, one could check for this flag and perform the fetch if it is set. By invalidating the component's main window, one insures that if the component is already visible, the fetch will occur immediately. Otherwise, the fetch will occur only if the component becomes visible.

### 73.24.6 Intercomponent Communication

Intercomponent communication refers to the technique of one component communicating information to a second component in the application. This can be

very useful if the action of one component affects another or if one component wishes to make use of a service provided by another.

The VueCentric<sup>®</sup> Framework provides two methods for implementing intercomponent communication. The first uses local events to send information to local subscribers. This method is useful when a component wishes to communicate information to multiple targets. It has the disadvantage of being unidirectional and anonymous.

A second method provides a much more tightly coupled communication link between components. This method utilizes a technique called dynamic discovery to locate other components in the environment and establish an interface reference to them. The Session object provides a number of methods that support dynamic discovery. They are divided into two groups, one for discovering visual components and another for discovering services. Each returns a reference to the IUnknown interface of the requested object which can then be cast to the desired interface. Using this reference, an object can then invoke any method or property on the interface. The methods are:

**FindObjectByCLSID** – This function searches the list of registered objects to find one that implements the class identified by CLSID. If the Last parameter is not nil (Nothing), the search begins following that object's entry in the list. In this manner, one can iterate through multiple object instances of the same class.

**FindObjectByIID** – This function searches the list of registered objects to find one that implements the interface identified by IID. If the Last parameter is not nil (Nothing), the search begins following that object's entry in the list. In this manner, one can iterate through all objects implementing a particular interface.

**FindObjectByProgID** – This function searches the list of registered objects to find one that possesses the programmatic identifier specified by ProgID. If the Last parameter is not nil (Nothing), the search begins following that object's entry in the list. In this manner, one can iterate multiple object instances of the same class.

**FindServiceByCLSID** – Request a reference to the service identified by CLSID. If the service is not already running, the CSS starts the service. If the service is not located, a nil (Nothing) value is returned. Otherwise, the return value is a reference to the service's default interface.

**FindServiceByProgID** – Request a reference to the service identified by ProgID. If the service is not already running, the CSS starts the service. If the service is not located, a nil (Nothing) value is returned. Otherwise, the return value is a reference to the service's default interface.

## 73.24.7 Creating Trace Log Entries

Component authors can create entries in the trace log for debugging purposes using the API suite provided by the session object of the CSS. This suite consists of the following:

Method or Property	Return Type	Description
TraceMode	Boolean	This property indicates whether or not trace mode is active. If trace mode is not active, calling any of the trace methods will have no effect. While it is not required to check the state of trace mode before invoking one of the trace methods, it is generally advisable to do so to avoid the overhead of sending trace information when trace mode is inactive.
TraceBegin(TraceClass,TraceType)	Integer	Call this method to initiate a trace log entry. TraceClass and TraceType determine how the entry is displayed in the trace log viewer. TraceClass defines an overall category for the log entry and TraceType defines a subcategory within the TraceClass. This method returns a handle that uniquely identifies the log entry being created.
TraceAdd(Handle,Value,IsHeader)	none	Call this method to add to the newly created log entry. Handle refers to the unique handle returned by the TraceBegin method. Value is the text to be added to the entry. If IsHeader is true, this indicates that the Value represents header information. When displayed in the log viewer, headers are centered, underlined, and red in color.
TraceEnd(Handle)	none	Call this method to complete the entry. Once this is done, the CSS fires a TRACE event which is intercepted by the log viewer and entered into the trace log.

Consider the following code example:

Delphi
<pre> var   Handle: Integer; begin   if vcSession.TraceMode   then begin     Handle := vcSession.TraceBegin('RPC', 'BEHOPTCX LAST');     vcSession.TraceAdd(Handle, 'Parameters', True);     vcSession.TraceAdd(Handle, '#1 : 733', False);     vcSession.TraceAdd(Handle, 'Results', True);     vcSession.TraceAdd(Handle, '733', False);     vcSession.TraceEnd(Handle);   end; </pre>

This code would generate a trace log entry that would appear as follows in the trace log viewer:

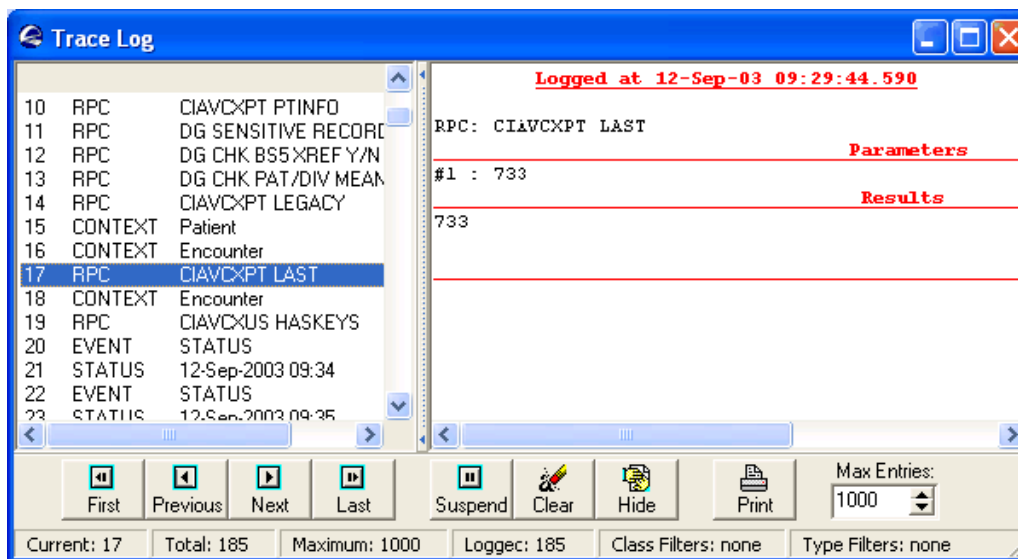


Figure 73-127: Sample Trace Log Entry

### 73.24.8 Embedding Licensed Controls

The use of licensed third party controls in the development of a component sometimes produces an unauthorized license exception when it is imbedded within another component. This is a problem that has been reported with ActiveX controls produced in Visual Basic that contain licensed controls on the primary form. It is unclear whether this is a “feature” or a “bug,” but it has received much discussion in the technical newsgroups. It is not a container issue, but rather a problem with the way Visual Basic passes license information to the imbedded control. Evidently, for the primary form this does not occur. Interestingly, for secondary forms it does. Therefore, imbedding a licensed component on a secondary form causes no problems. This phenomenon lends itself to an interesting, albeit somewhat inelegant, workaround:

Create a dummy form in your Visual Basic project. Place one copy of each licensed component that you will be imbedding in your primary form onto the dummy form. Load the dummy form in the Initialize method of your control. This registers the license information for the controls. Unload the dummy form in the very next statement (it is no longer needed). The licensed components will now function properly on the primary form.

### 73.24.9 Forced Context Changes

While context changes are a mostly democratic process, there are two situations where a context change can be forced even when one of the participants rejects the request. The first situation occurs during a forced shutdown of the application when the context of each context object is cleared. The second situation occurs when a CCOW client requests a context change and elects to override a participant’s objection. In either event, the programmer should be prepared to react to a forced context change and not assume that a rejection of the request will always abort the change.

### 73.24.10 Integrating Help Content

The Visual Interface Manager interrogates each visual component as it is loaded to determine which ones provide on-line help. It then provides access to the help documentation by means of the Help | Help On menu. The VIM attempts to acquire three pieces of information when it interrogates a component: the help file name, the display name of the component, and the help context identifier. It first examines the type library to determine if a Help File attribute is defined for the type library and Help String and Help Context attributes are defined for the component's default interface. If these attribute values are found, they are used for the help file name, display name, and help context identifier, respectively.

For example, the type library shown below defines a Help File attribute of “vcPatientID.chm”, a Help String attribute of “Patient Selection,” and a Help Context attribute of 124.

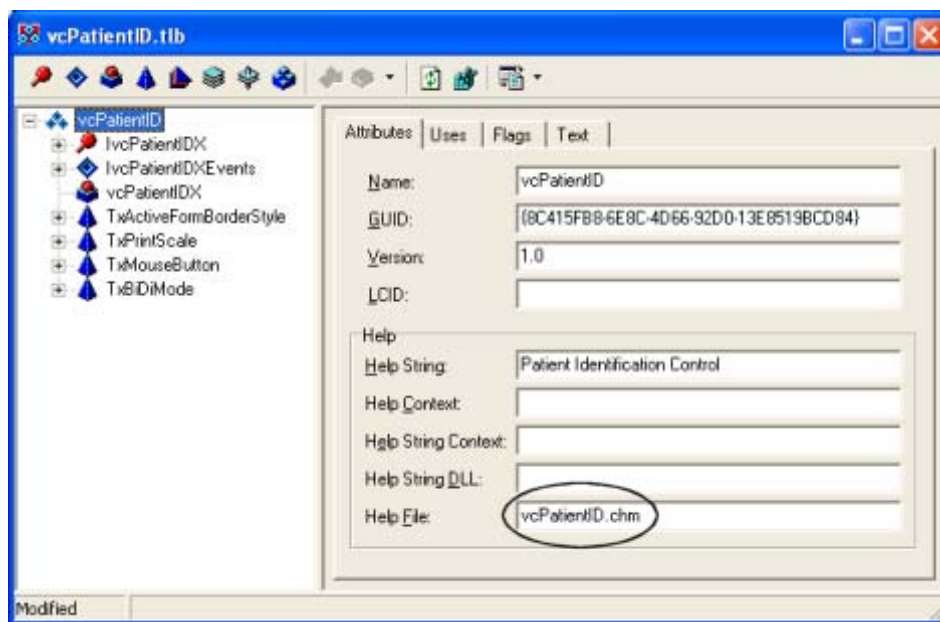


Figure 73-128: Help File Attribute

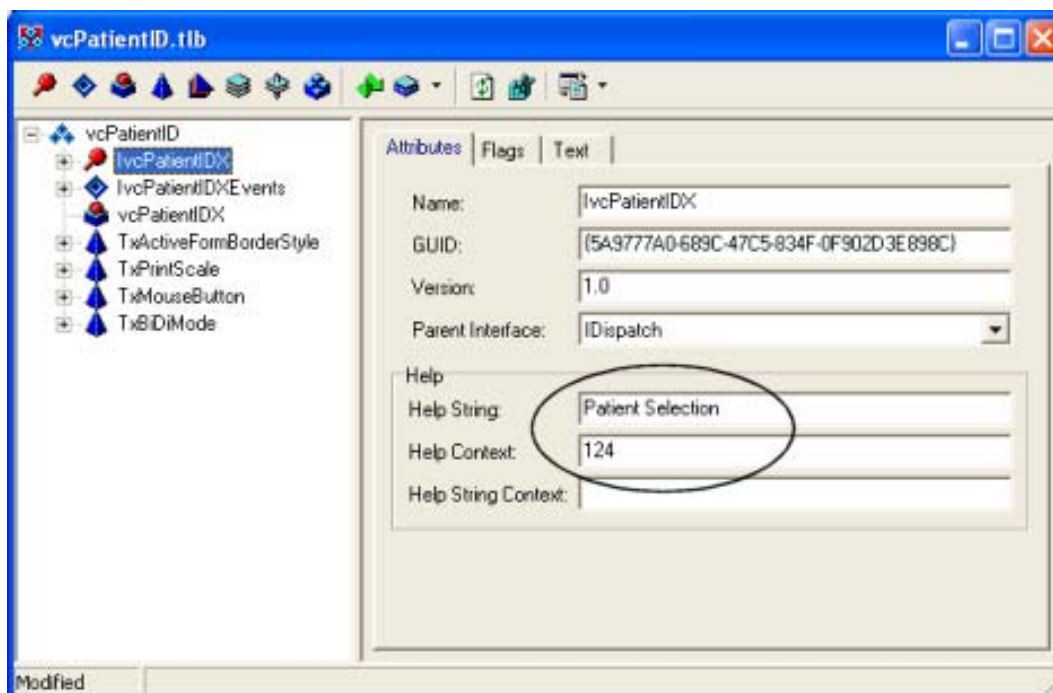


Figure 73-129: Sample Help String and Help Context Information

If the VIM does not obtain the required information from the type library, it then examines the default interface of the component for a `HelpFile` and `HelpContext` property. If it finds these, it uses those values for the help file name and help context identifier, respectively, and uses the component's `Name` property from the VUECENTRIC OBJECT REGISTRY file as the display name.

If the VIM can obtain a value for the component's help file name by one of these methods, that component's display name will appear as a submenu under the **Help | Help On** menu. Invoking the submenu will load the specified help file (both Windows and HTML help formats are supported) at the point referenced by the context identifier (if found).

## 74.0 Contact Information

If you have any questions or comments regarding this distribution, please contact the OIT User Support (IHS) by:

**Phone:** (505) 248-4371 or  
(888) 830-7280

**Fax:** (505) 248-4299

**Web:** <http://www.ihs.gov/General Web/HelpCenter/Helpdesk/index.cfm>

**Email:** [support@ihs.gov](mailto:support@ihs.gov)