RESOURCE AND PATIENT MANAGEMENT SYSTEM

Electronic Health Record

(EHR)

Technical Manual Volume 1

Version 1.1
September 2020

Office of Information Technology
Division of Information Technology

Table of Contents

Document Revision History

Version	Date	Author	Description	Sections
v1.1	9/2007	Doug Martin	v1.1 Release	
v1.1, p17	2/2016	Phillip Salmon	v1.1 Patch 17 Release	
v1.1, p25	8/2019	Phillip Salmon	v1.1 Patch 25 Release	• Preface • 1.0 • 2.3.1.4.4–2.3.1.4.5 • 2.3.1.5 • 2.3.2–2.3.3 • 2.4 • 2.5.1 • 2.8 • 2.12 • 2.16.5.1.16 • 4.15.13 • 5.12.6 • 6.4 • 6.5.5 • 6.10 • 6.16.1.5 • 9.9 • 10.10

Preface

This guide provides information regarding technical aspects of the Indian Health Service Resource and Patient Management Electronic Health Record (RPMS-EHR) v1.1 software. Its target audience is local and regional information technology support personnel who may be called upon to configure or troubleshoot the application. This guide has been broken into four separate files of which this is the first.

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 80 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
- Remote Procedure Call (RPC) Broker
- Context Objects and Related Components
- PCC-based Components
  - Asthma Control Component
  - Chart Review Component
  - Patient Goals Component
- Problem List
- TIU-based Components
- Order Entry
- General Reporting
- Notifications
- Reminders
- Electronic Signature
- Electronic Prescribing
- 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 may 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 Clinical Context Object Workgroup (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 ensures the interoperability of components developed in accordance with the specification. The key constituents of the VueCentric framework are described in the following paragraphs.

The 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 CSS comprise the middle tier and provide shared resources that all components may 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 may 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 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 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. Updating of the software, application and object, requires write access to the Windows Registry. This can be accomplished via local user settings or use of the vcUpdater Service which is described in the separate vcUpdater Service documentation found in the doc folder of the repository.

In addition to the architectural elements described previously, 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 may 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 may 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.

Figure 2-1 summarizes the architecture of the VueCentric Framework.

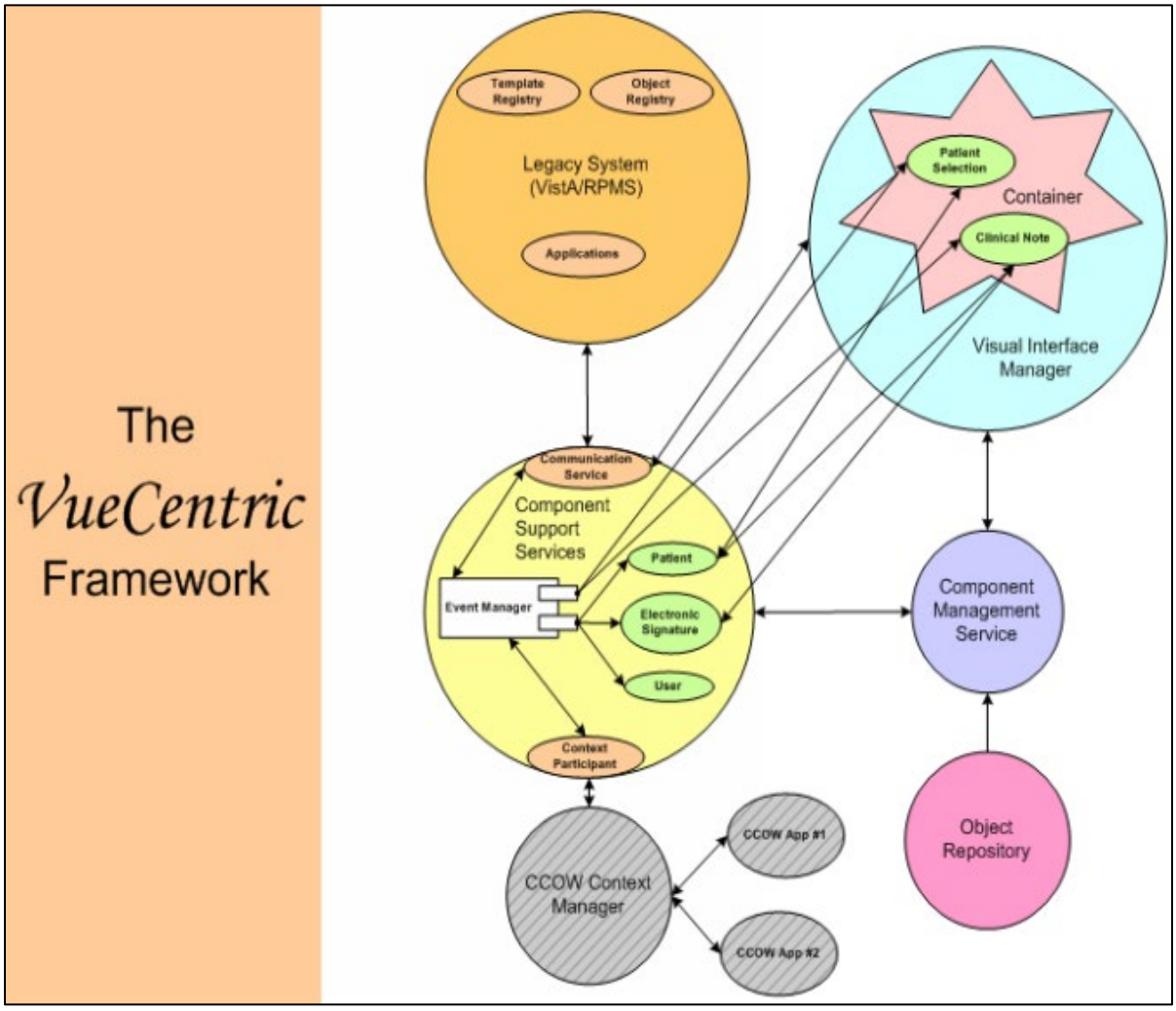


Figure 2-1: VueCentric Framework

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 Utility

2.3.1.1 Introduction

The **VueCentric System Management** (vcManager) **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 may 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 may be preceded by a server selection dialog, depending on how the connection service is configured. If this dialog displays, the user must first select the server with which to establish a connection.

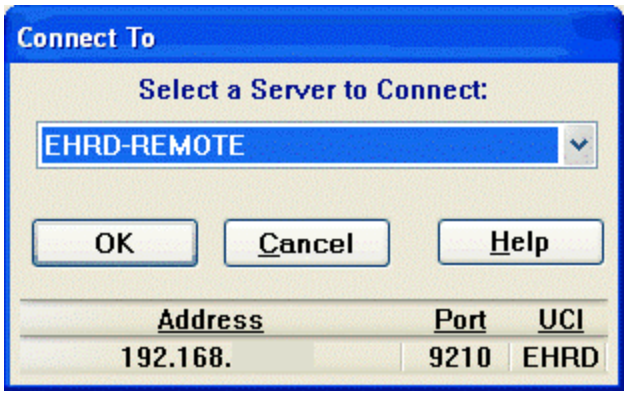


Figure 2-2: Logon Screen

In either case, the logon dialog displays:

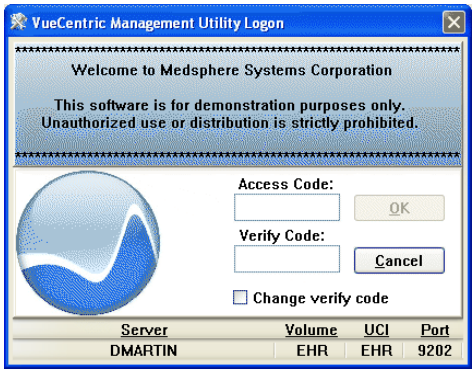


Figure 2-3: Logon dialog

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.
File	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.
File	Logout	Logs off the current host but does not terminate the application.
File	Exit	Logs off and terminates the application.

Menu	Menu Item	Effect
Tools	User Configurable	User configurable by adding to the [tools] section of the vcManager.ini file. Use NotePad to create a text file called vcManager.ini in the same folder as the vcManager application and format according to the example that follows. The following example creates a single menu item labeled Registry Editor under the Tools menu, which invokes the Windows registry editor when clicked. [tools] Registry Editor=regedit.exe
Help	Contents	Displays the help file table of contents.
Help	About	Displays the About tab.

2.3.1.4 Object Registry Tab

The **Object Registry** tab permits the registration of objects that may then be accessed by the VueCentric framework. Objects may be visual components that are available in the *Add Object* dialog of the VueCentric VIM or services that visual components may 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.

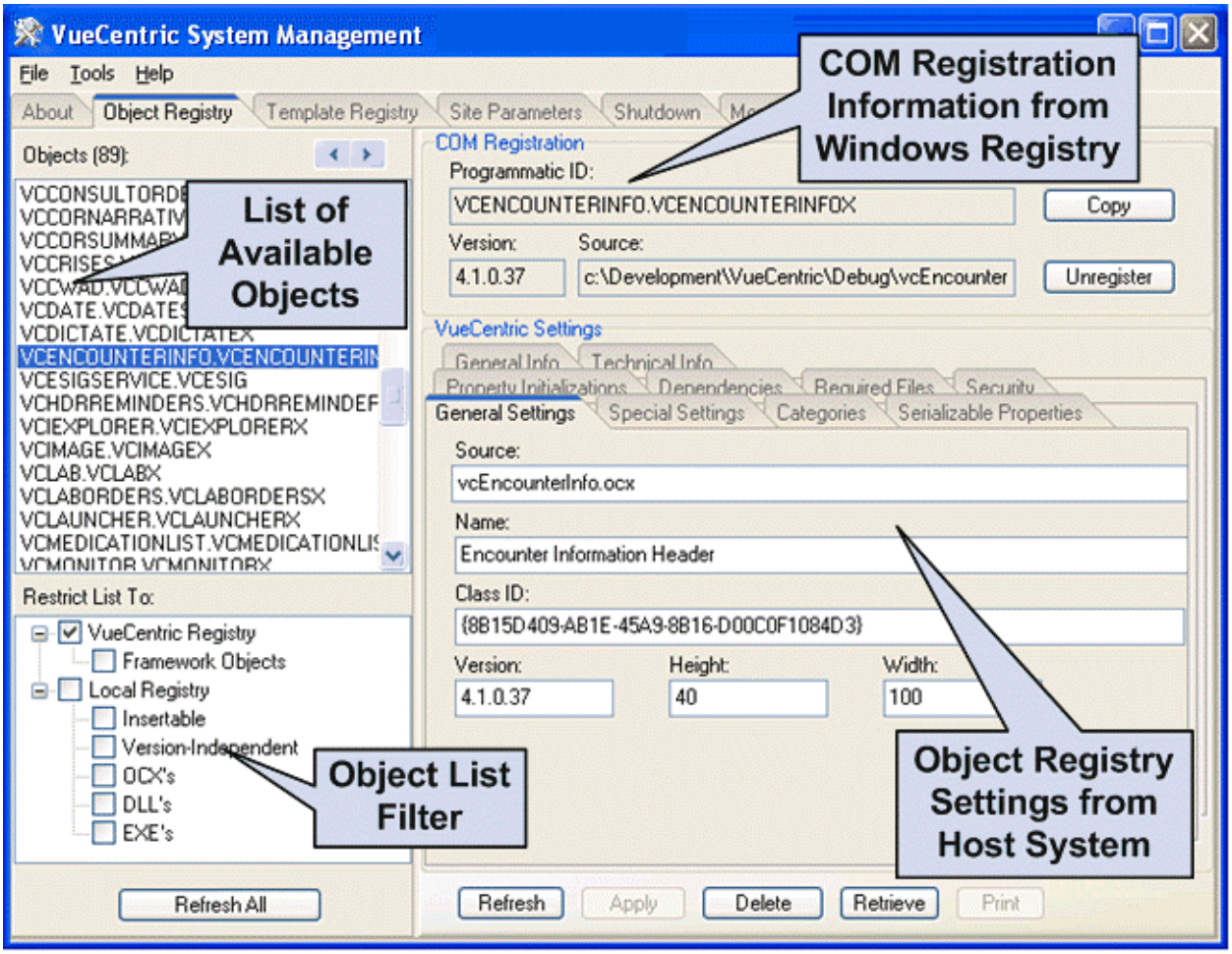


Figure 2-4: VueCentric System Management Window

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.

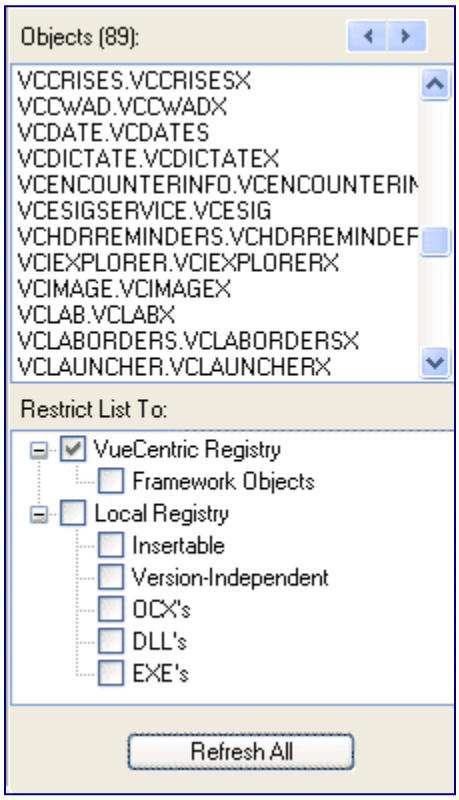


Figure 2-5: Object List Pane

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 (RPMS) 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 the following list.

Filter Item	Effect
VueCentric Registry	Includes entries from the VueCentric Object Registry file located on the host system.
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 may be further restricted using one or more of the filter items that follow:
Insertable	Restricts list to local registry entries marked with the insertable attribute.
Version-Independent	Includes version-independent programmatic identifiers. COM objects may 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 an .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 may be blank. These fields are display only and cannot be modified.

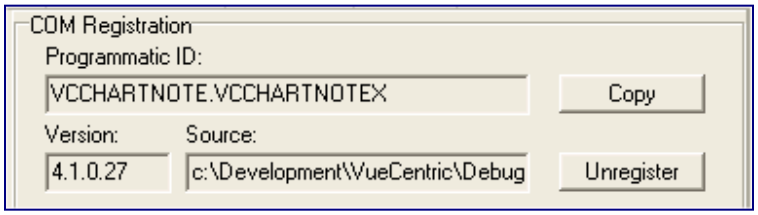


Figure 2-6: COM Registration Pane

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 may be modified by the user.

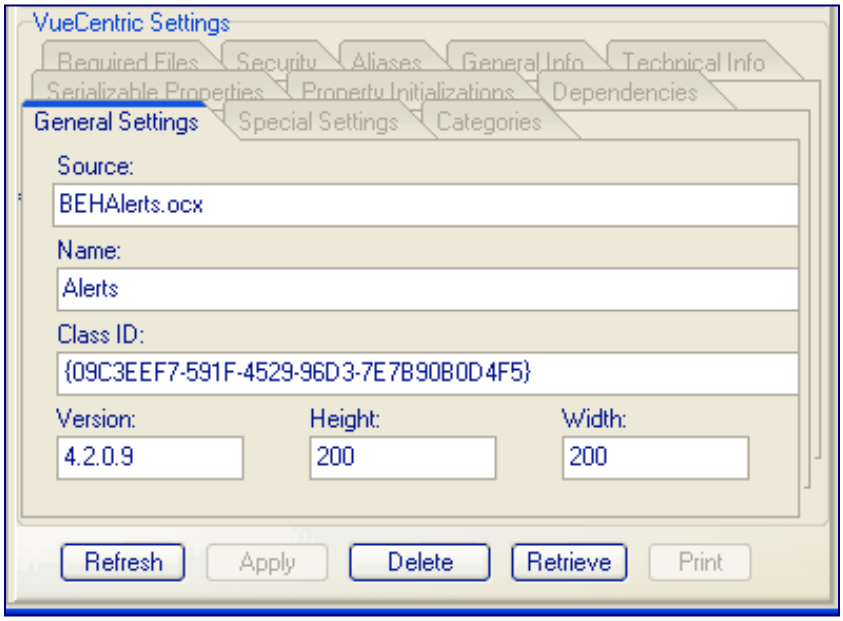


Figure 2-7: VueCentric Settings Pane

The pane consists of 11 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:

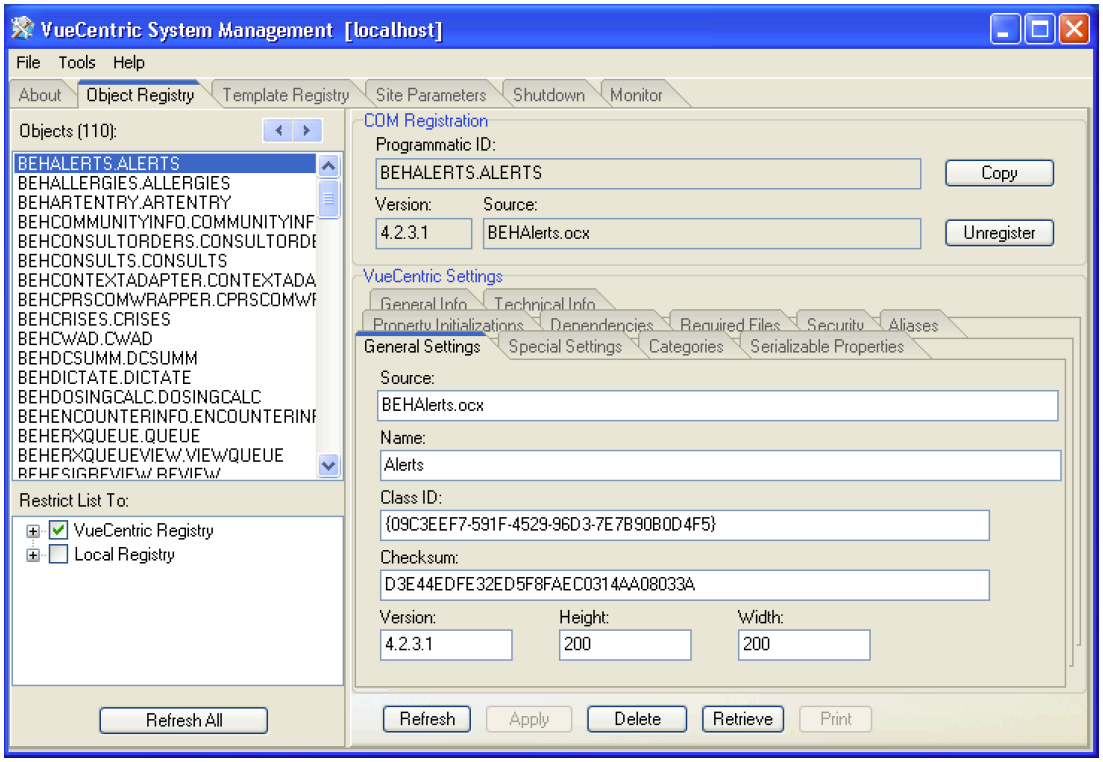


Figure 2-8: General Settings Tab

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 VIM. 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 as defined by Microsoft. 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 been 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 VIM handles an object.

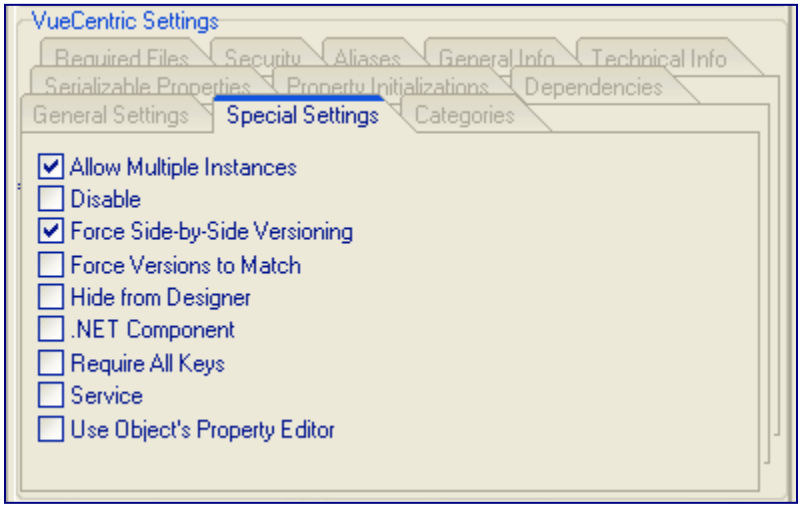


Figure 2-9: Special Settings Tab

The table that follows describes the function of each of these attributes.

Attribute	Effect
Allow Multiple Instances	If checked, multiple instances of the object may exist within the current configuration template. By default, only one instance of an object may 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 VIM, the object's icon will be grayed out 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	Side-by-side versioning refers to the ability to have multiple versions of the same component residing on a machine at the same time. 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. 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 ensure that the client versions are appropriate for the target host. 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. Unless an object is clearly independent of the target host system, it is recommended that it be registered with side-by-side versioning enabled.
Force Versions to Match	Normally, the Component Management Service performs an automatic update only if the version of the master copy of the object is newer than the local copy (or a local copy does not yet exist). If this attribute is checked, an automatic update occurs anytime the versions of the master and local copies are not the same. This feature is useful if a newer object version is problematic and there is a need to revert to a previous version.
Hide from Designer	Checking this attribute prevents the object from appearing in the Add Object and Required Services dialogs of the VIM. This setting is appropriate for any supporting files that are not ActiveX components.
.NET Component	If checked, this object represents a .NET component. This setting is necessary to ensure successful creation of a .NET component with the VIM.
Require All Keys	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.
Service	If checked, the object represents a service that may 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 Component Support Services layer. Marking an object as a service ensures that it does not appear in the Add Object dialog, and that it is automatically loaded and started when needed.

Attribute	Effect
Use Object's Property Editor	The VIM 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 VIM to use the object's internal property editor instead of the generic one. For more information about the generic property editor, see the discussion of the Serializable Properties tab.

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 VIM. They can also help organize nonvisual components into functional categories as well.

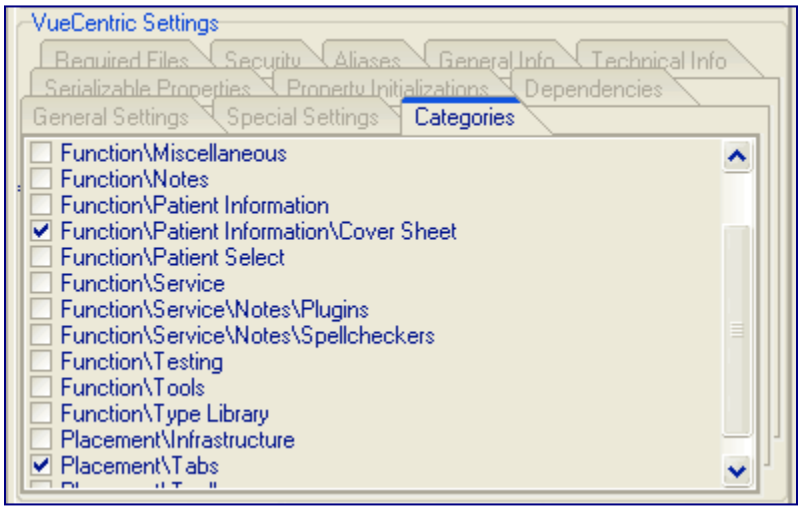


Figure 2-10: Categories Tab

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: 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 VIM 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.

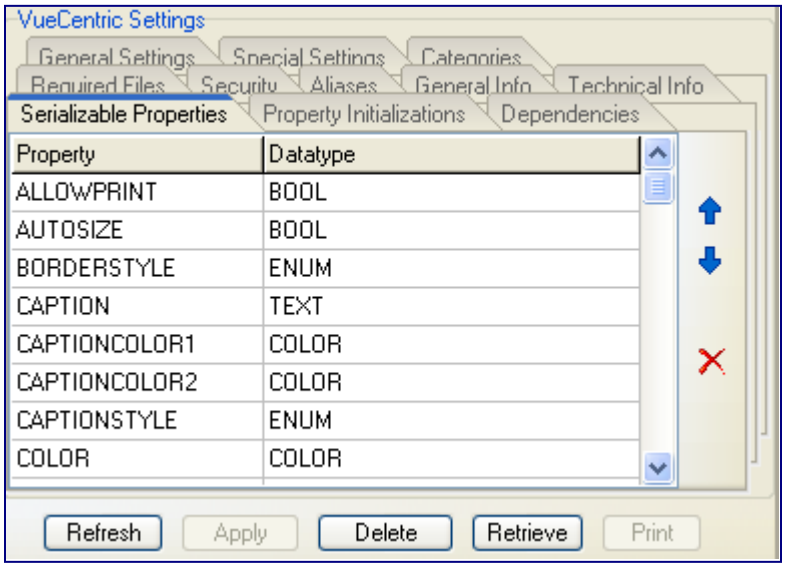
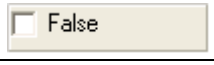


Figure 2-11:Serializable Properties Tab

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 following table.

To delete an entry, click 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.	
COLOR	The property is a color.	
ENUM	The property is one of several fixed values.	

Datatype	Description	Property Page Appearance
FILE	The property is a filename.	C:\Splash.jpg ...
FLAG	Similar to an enumeration, but selections are not mutually exclusive and may overlap (i.e., selecting one may affect the selection of others).	Top; Left
FONT	The property is a font.	MS Sans Serif ...
HIDDEN	The property is not displayed in the property editor, but its value is saved in the template.	N/A
ICON IMAGE	The property is an icon or graphic image file.	C:\Splash.jpg ...
INT	The property is an integer value.	600
TEXT	The property is simple text.	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.

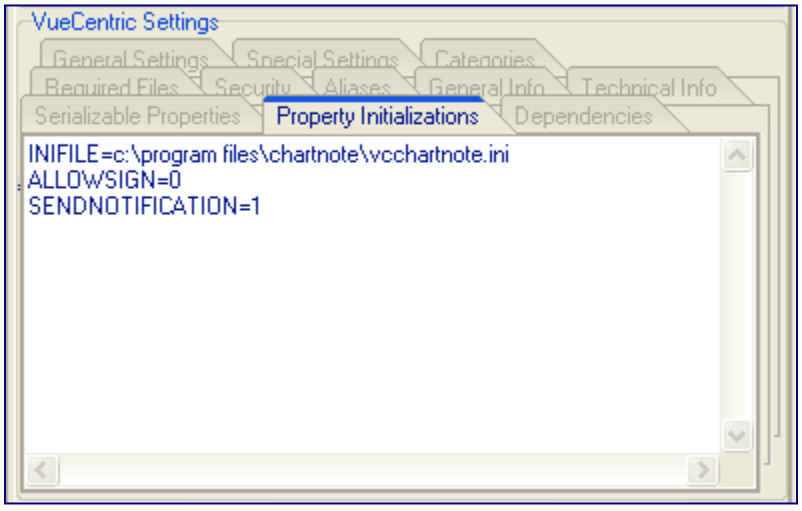


Figure 2-12: Property Initializations Tab

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.

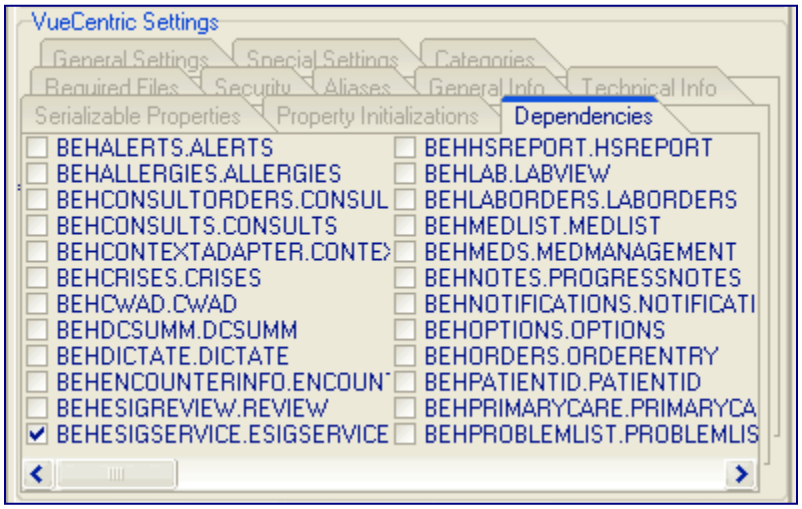


Figure 2-13: Dependencies Tab

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 may 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.

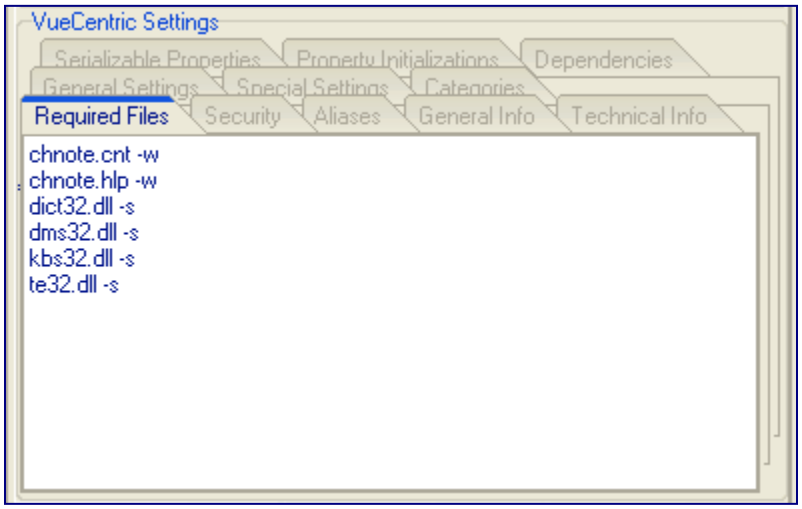


Figure 2-14: Required Files Tab

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 following table.

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.

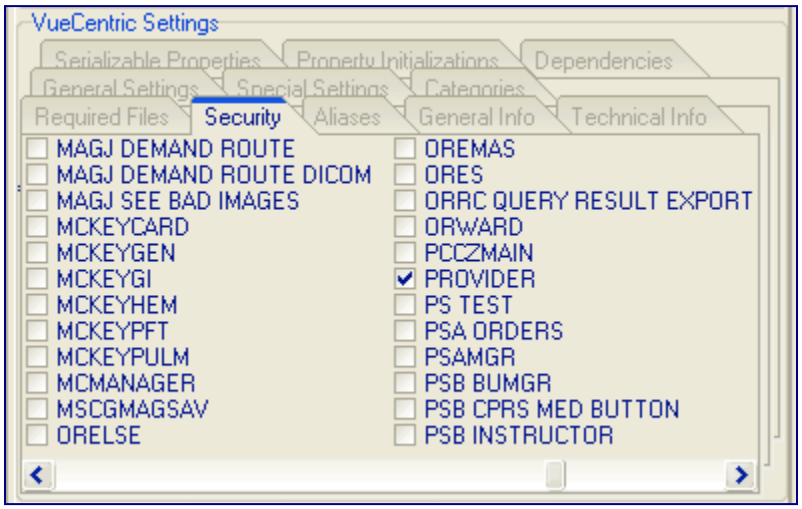


Figure 2-15: Security Tab

> **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.

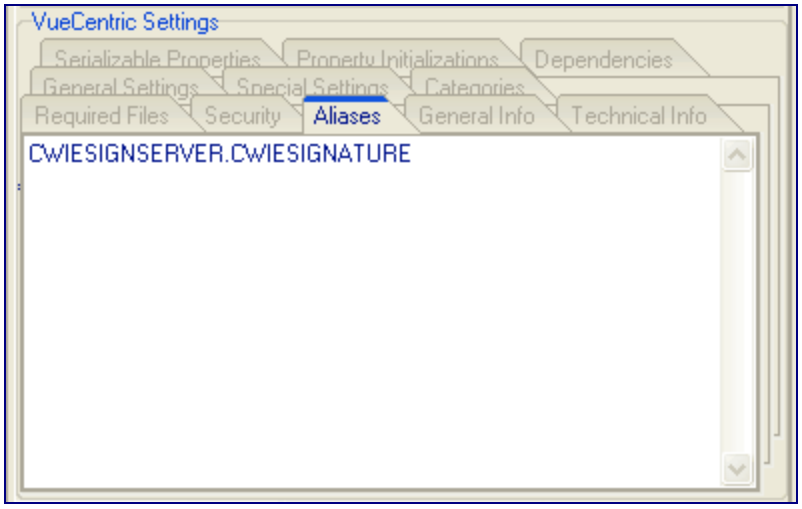


Figure 2-16: Aliases Tab

2.3.1.4.14 General Info Tab

This tab displays general information about the object that describes its purpose and function.

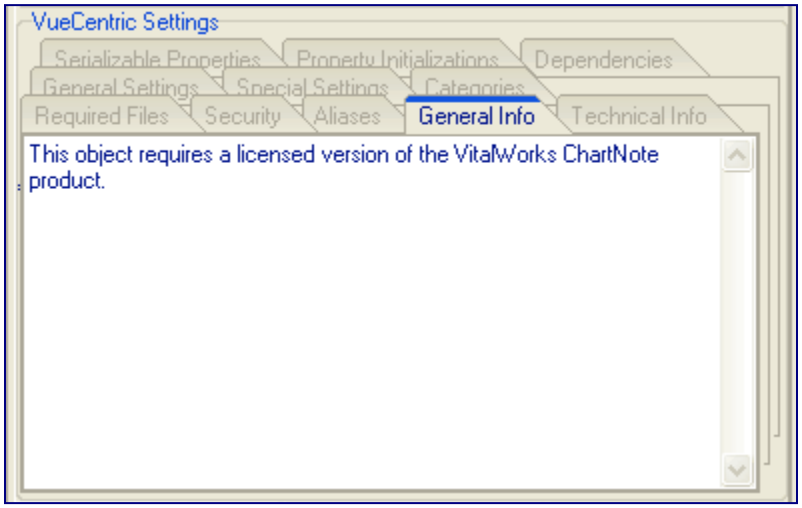


Figure 2-17: General Info Tab

2.3.1.4.15 Technical Info Tab

This field is to be used to provide technical information about an object such as a detailed description of its properties and methods.

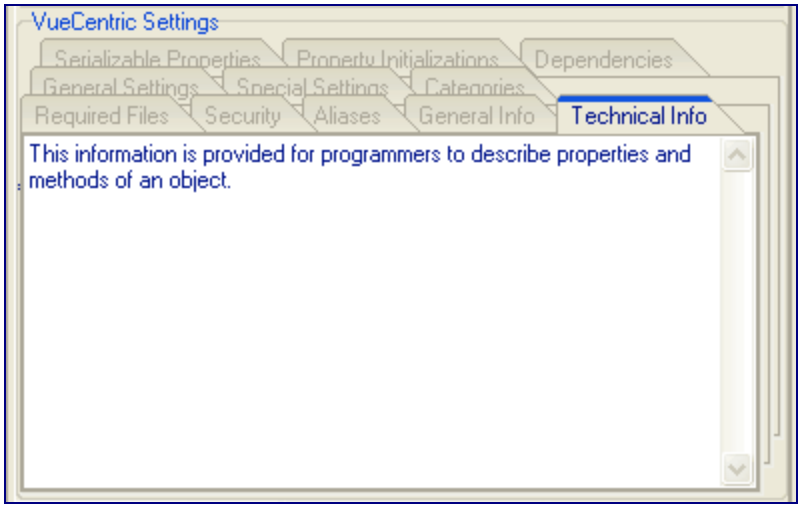


Figure 2-18: Technical Info Tab

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 display in the Add Object dialog of the VIM.

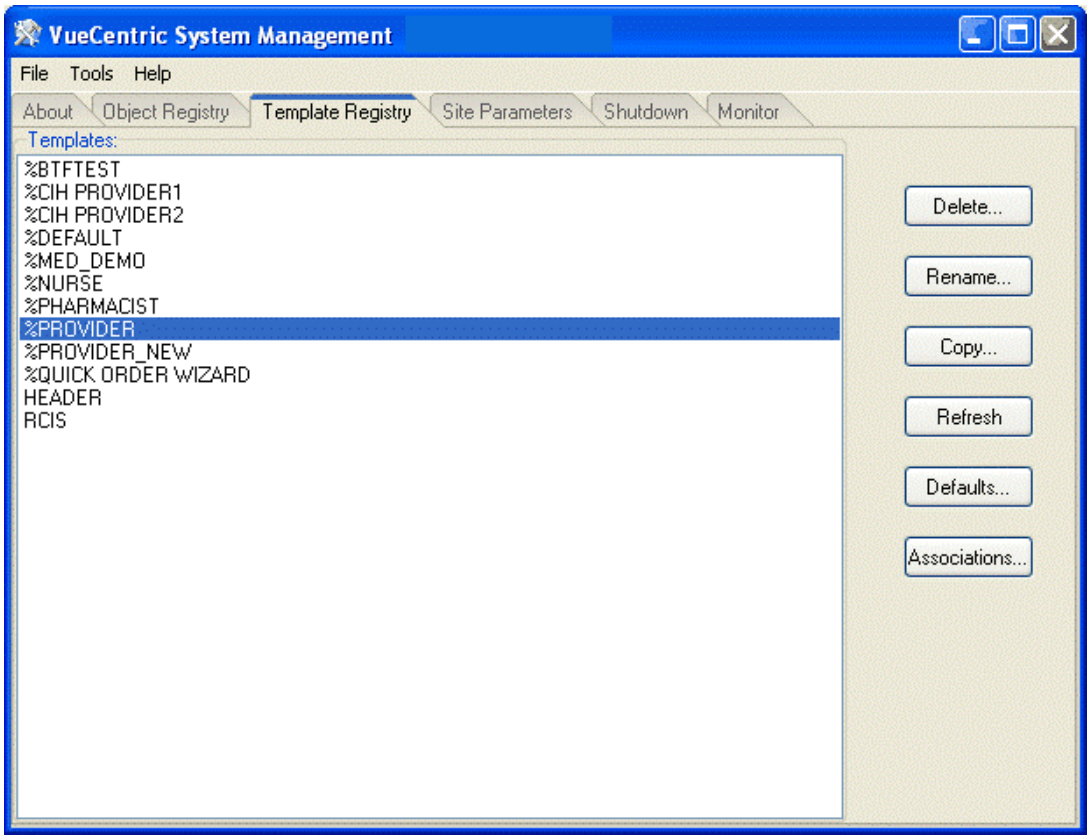


Figure 2-19: Template Registry Tab

The buttons on this window operate as follows:

- **Delete** – Deletes the selected template. Any associations with the deleted template are also deleted.
- **Rename** – Prompts for a new template name and renames the selected template to that name.
- **Copy** – Prompts for a new template name and creates an exact copy of the template under that name.
- **Refresh** – 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.
- **Defaults** – Displays the Default Templates dialog. This dialog enables changing the default template for various entity types.
- **Associations** – 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 follows:

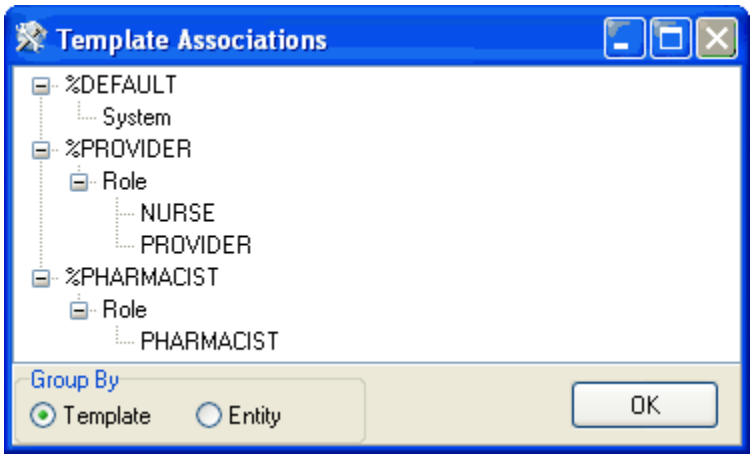


Figure 2-20: Template Associations Dialog Sorted by Template

or by entity as follows:

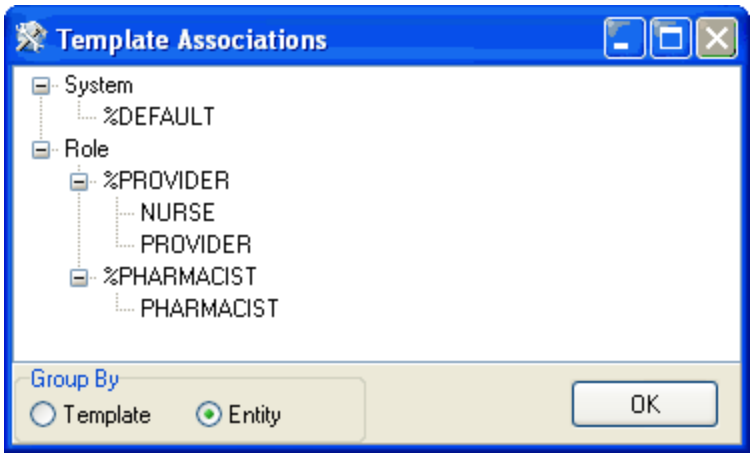


Figure 2-21: Template Associations Dialog Sorted by Entity

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 follows:

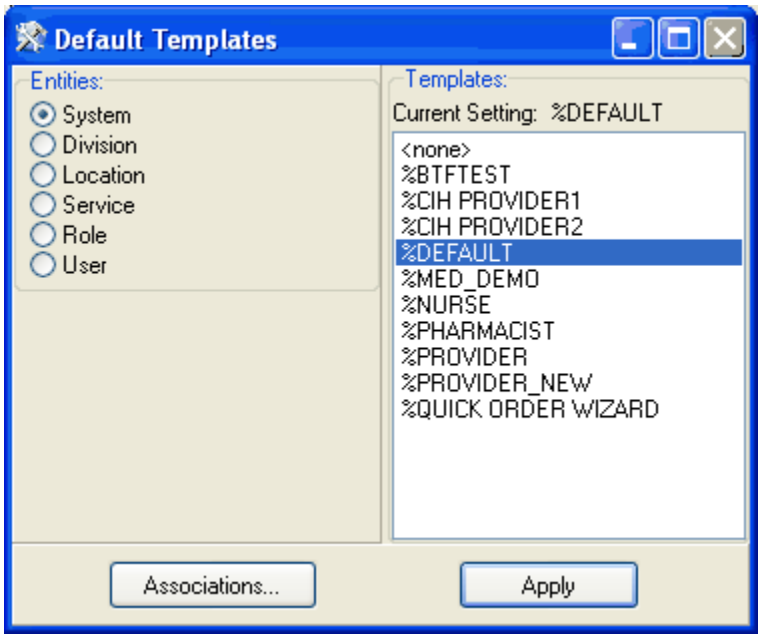


Figure 2-22: Default Templates dialog

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 display as follows:

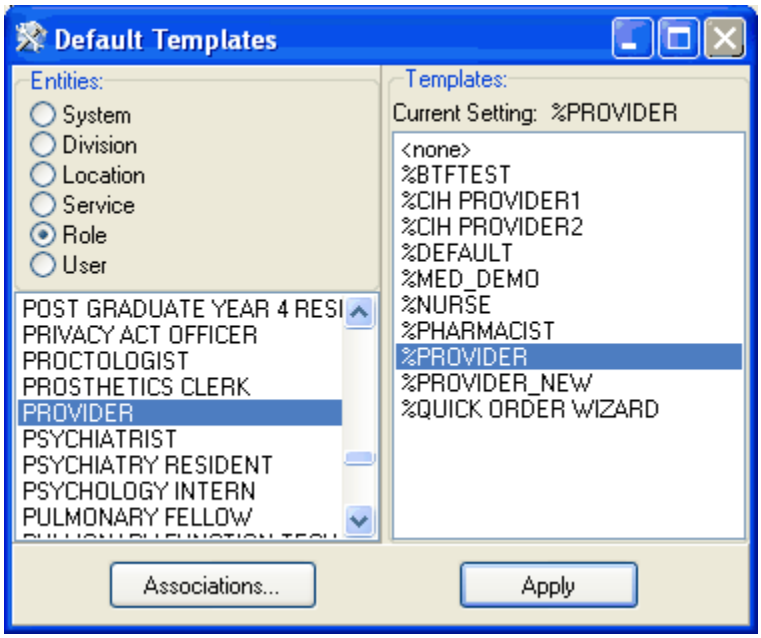


Figure 2-23: Default Templates dialog with User Roles defined

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 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 may differ from that described here. The package manager may elect to add or remove additional parameter elements depending on local needs.

The table that follows describes each of the parameters and their function.

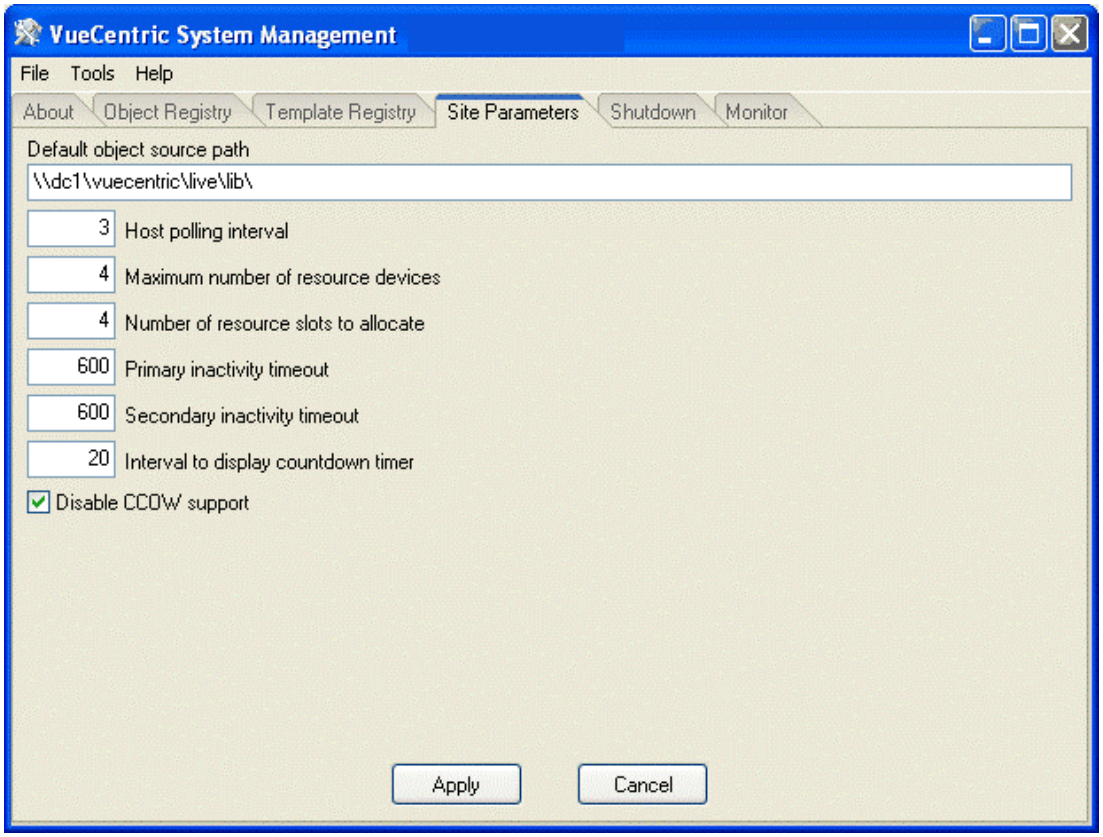


Figure 2-24: Site Parameters tab

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 of a load on the host system but reduces responsiveness. **Note**: Changing this value has an immediate effect on all running VueCentric applications.
Maximum number of resource devices	VueCentric 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 may 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.
Primary inactivity timeout	This is the inactivity timeout interval, in seconds, after which the VIM 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 VIM will terminate the application. If this value is 0, the secondary timeout function is disabled.
Interval to display countdown timer	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 VueCentric 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 VueCentric Framework, check this parameter.

2.3.1.7 Shutdown Tab

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

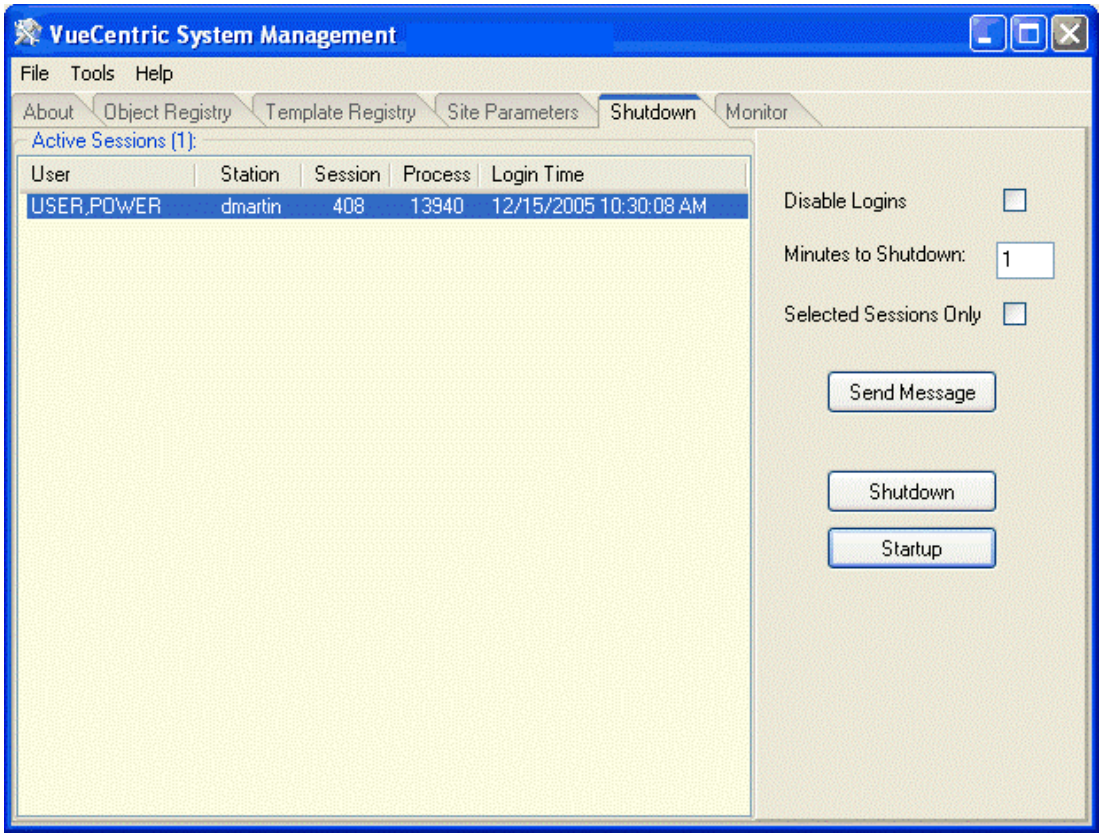


Figure 2-25: Shutdown Tab

The pane on the left displays all currently running VueCentric sessions. It is updated dynamically as sessions come and go or may 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 in the following paragraphs 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 proceeding). 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 may 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.

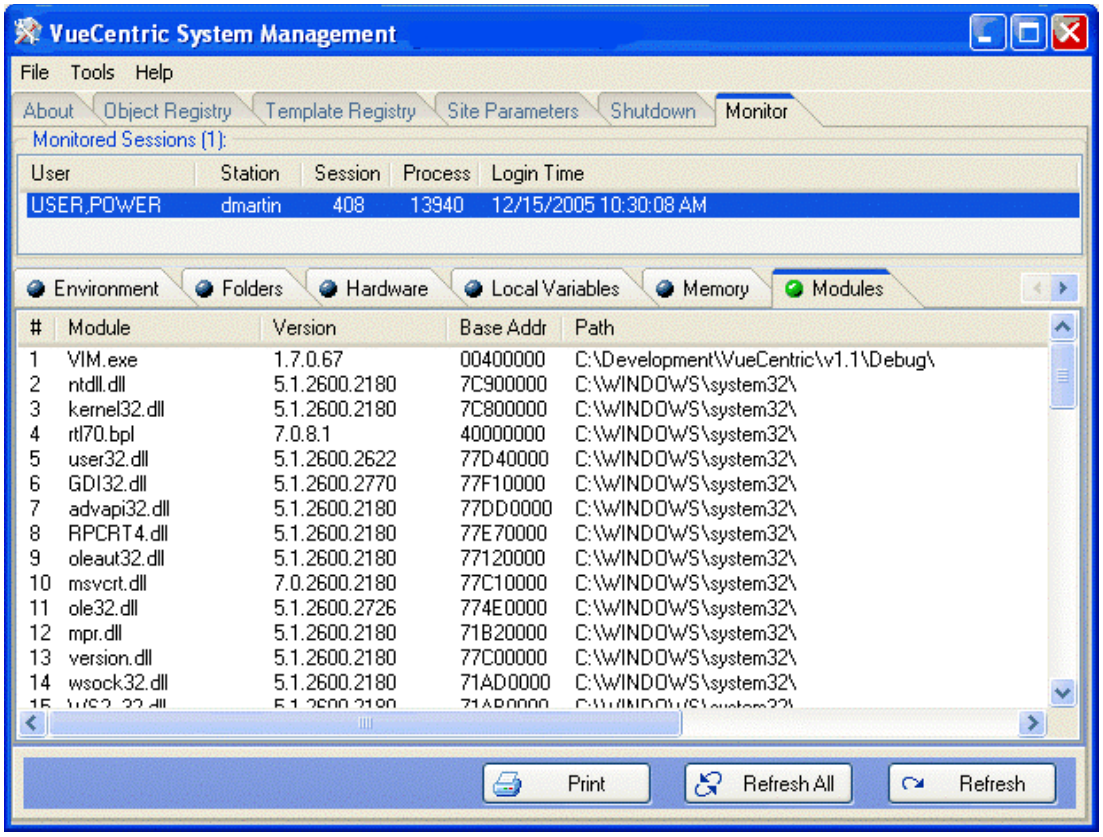


Figure 2-26: Monitor Tab

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 the following paragraph), 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 VIM. 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 may 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 VIM.

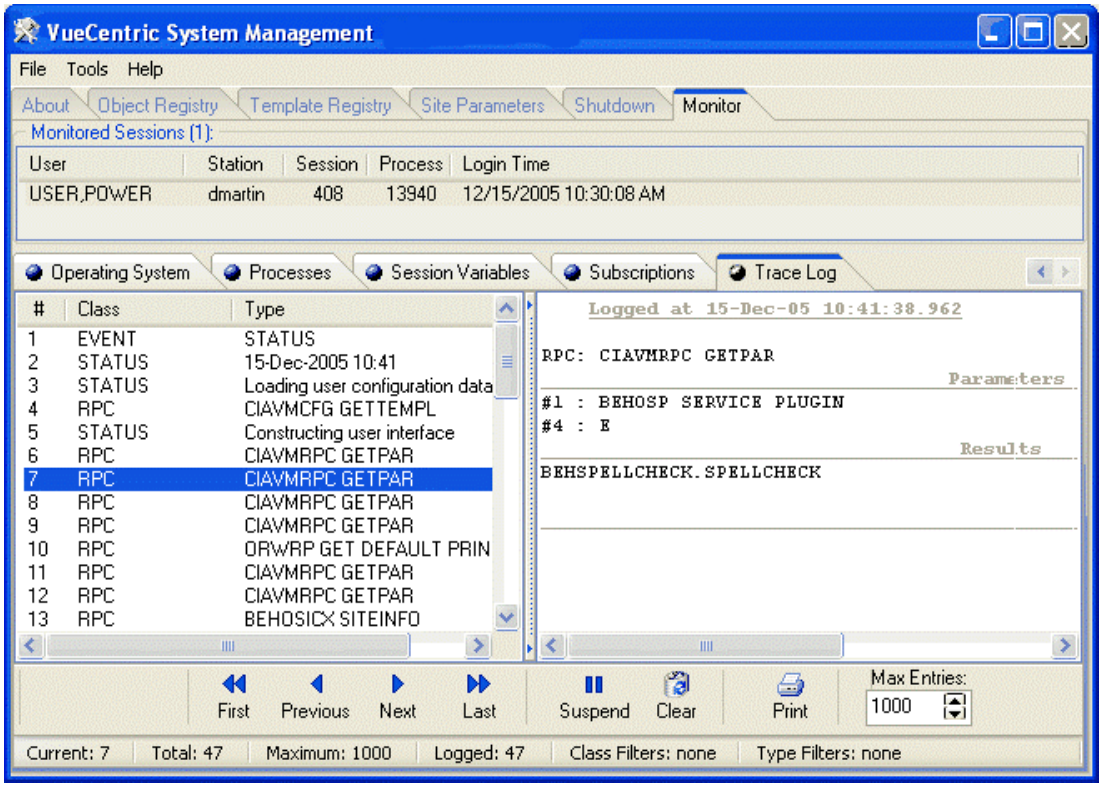


Figure 2-27: Trace Log Tab

2.3.2 Ini Configuration Utility

The Ini Configuration (vcIniConfig) utility may 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 may 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).

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:

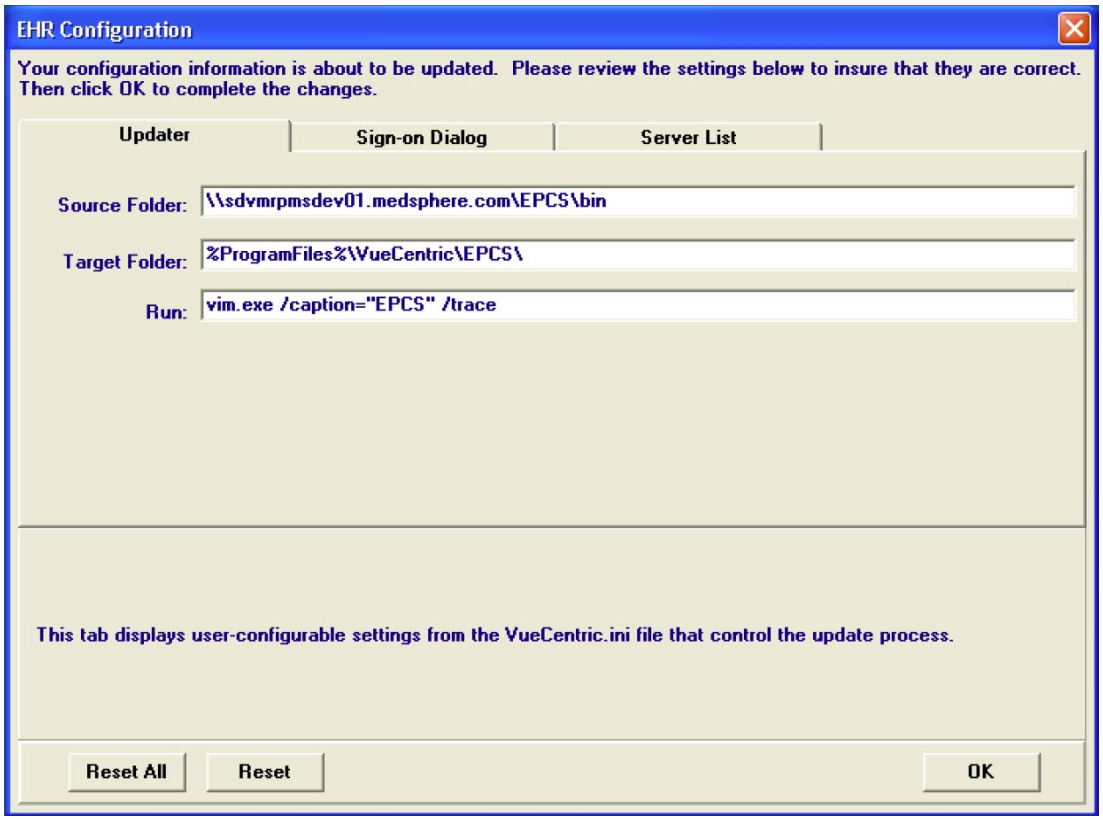


Figure 2-28: EHR Configuration Dialog

Three key settings from the VueCentric.ini file are displayed and may 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 may be found in the RPMS-EHR v1.1 Installation Guide. The various parameters supported by the run attribute can be found in Visual Interface Manager section of this document.

The Sign-on Dialog and Server List tabs allow editing of the vcBroker.ini properties.

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.

2.3.3 Repository Check Utility

2.3.3.1 Introduction

The Repository Check Utility found the in the utl folder of the repository allows the repository manager to quickly compare the content with a control file containing the version and checksums for each released file.

2.3.3.2 Program

Launching the RPMSEHRRepositoryCheckUtility.exe application will present the user with the following screen.

The application will derive the Application and Object repository folder paths based on where the utility was launched. The user will need to select **the Environment Check File** appropriate for the patch level of the repository and then click **Start Scan**.

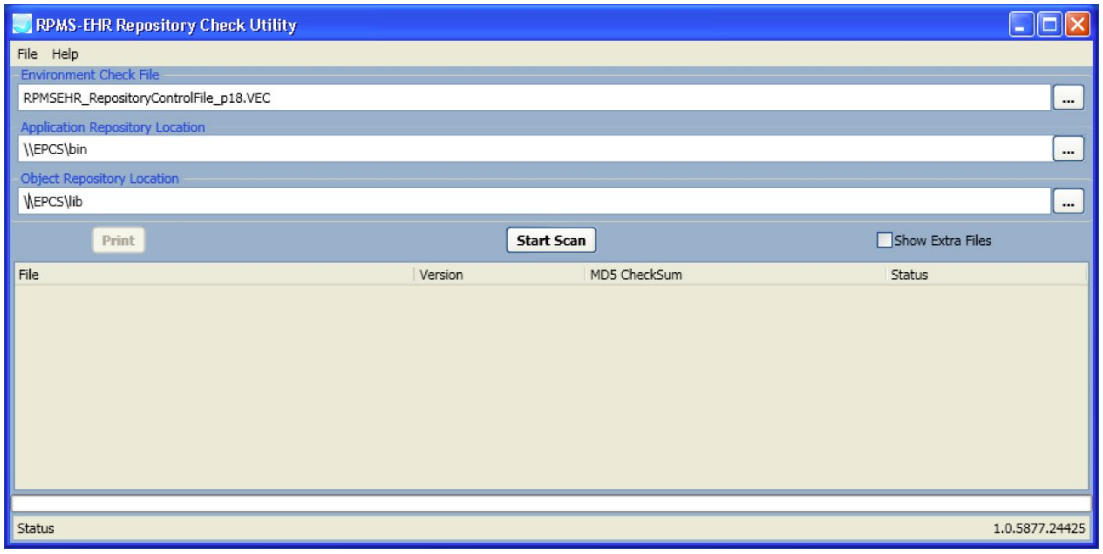


Figure 2-29: RPMS EHR Repository Check Utility

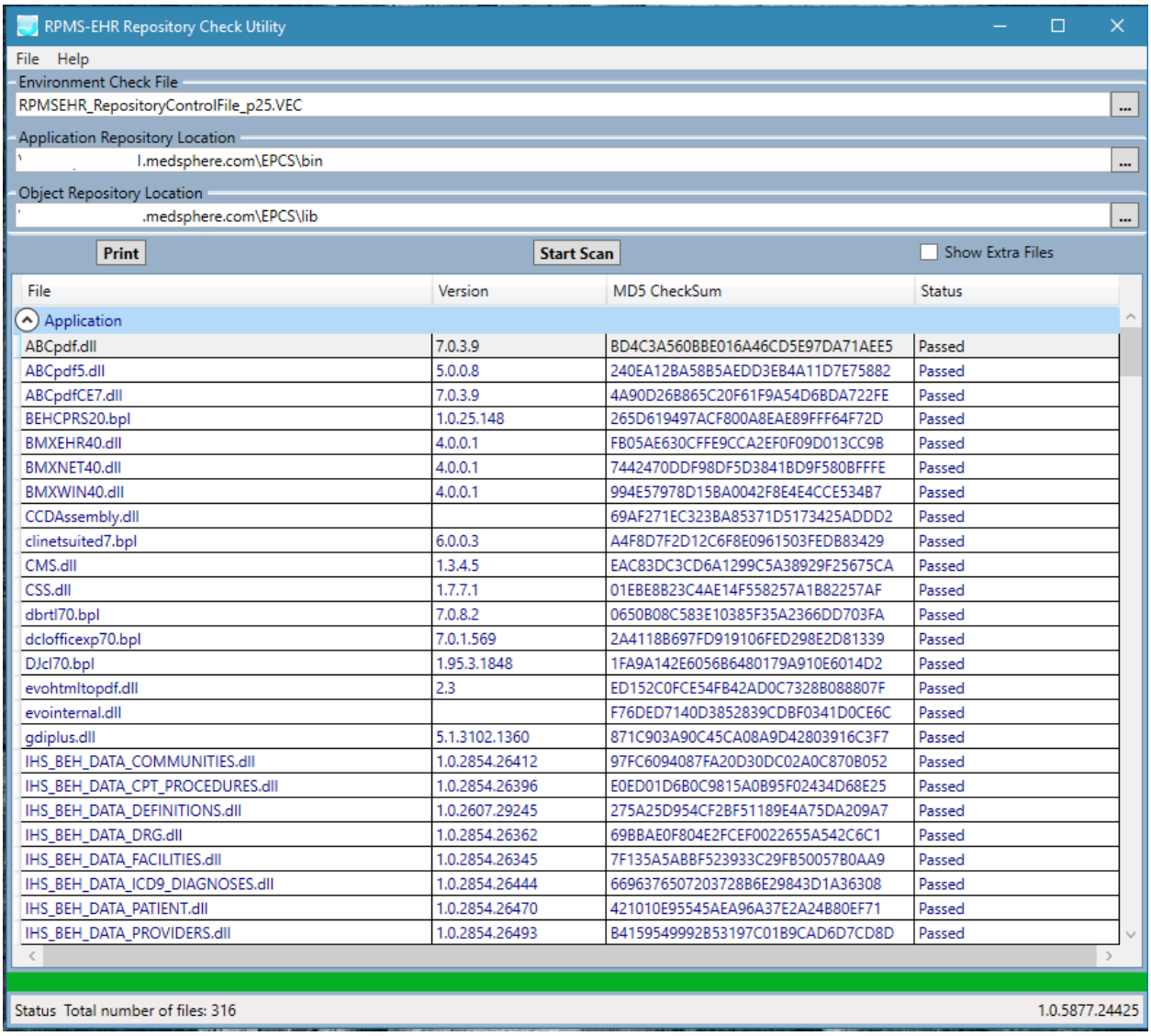


Figure 2-30: RPMS EHR Repository Check Utility – Application Section

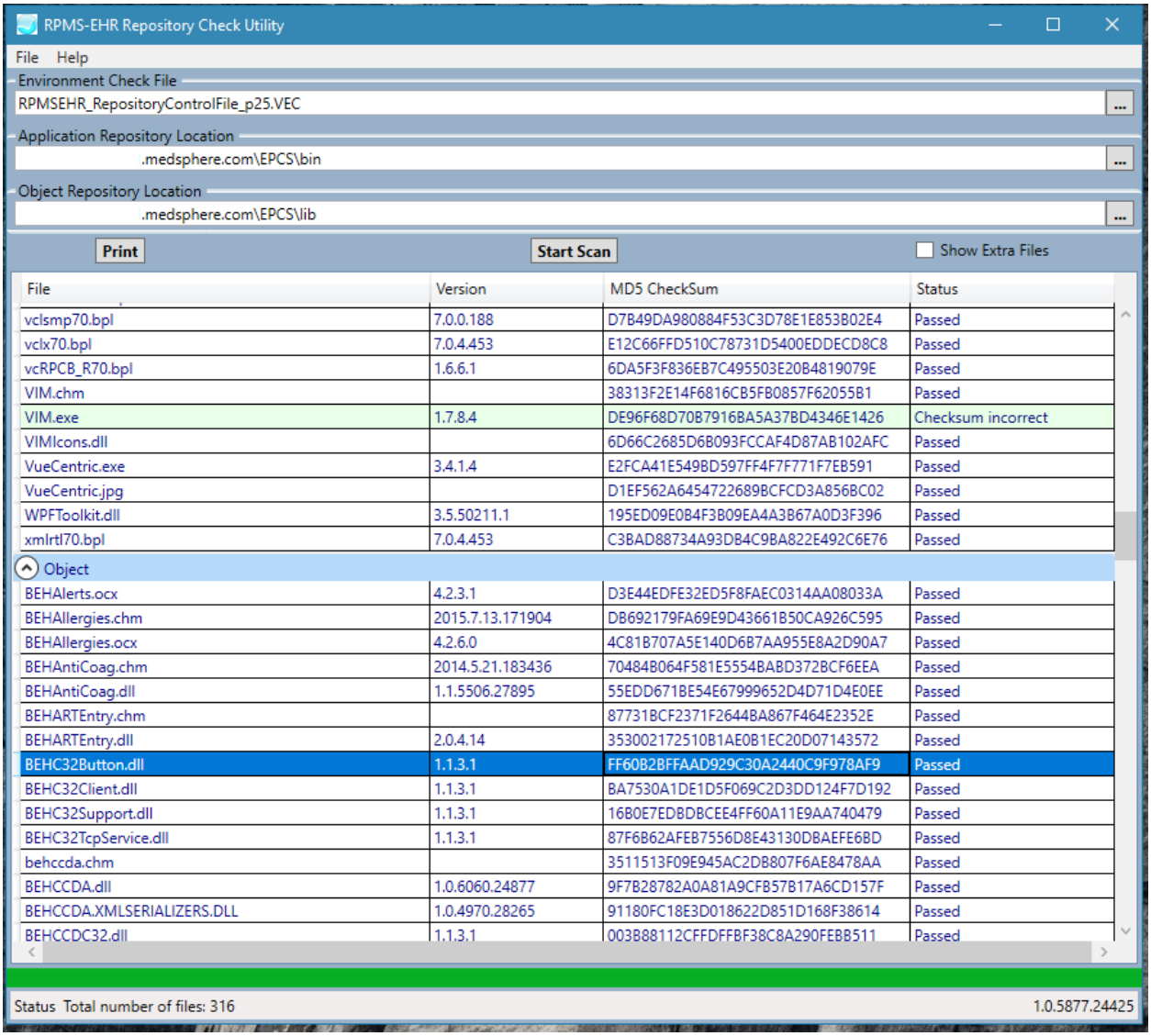


Figure 2-31: RPMS EHR Repository Check Utility – Object Section

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).

Routine	Description
CIAVMRPC	Shared remote procedure calls.
CIAVUTIL	Miscellaneous utilities.
CIAVUTIO	Host IO Support
CIAVUTPR	Parameter management.

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 may 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.
WIDTH	5	Integer		The default width, in pixels, when an object is created in design mode.
CATEGORY	6	Pointer (19930.21) (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.
MD5 CHECKSUM	7	Text		This is the MD5 checksum for the file.
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.

Field Name	#	Datatype	Indexes	Description
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.
DISABLED	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.
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.
NOREGISTER	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 want to be discovered by other objects or if the object performs its own registration.

Field Name	#	Datatype	Indexes	Description
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 may 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)

This file contains configuration templates for the VIM. A configuration template is a snapshot of a visual interface. Entries in this file are managed by the VIM itself and by the VueCentric System Management Utility.

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.

Checks if the specified user has a security key or has a parameter setting 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.

Checks if the specified encrypted verify code matches that of the current user.

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: public

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: public

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: public

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: public

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: public

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).
<return value>	String List	Values associated with a multi-valued parameter 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: public

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 List	Value associated with a word processing parameter 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	^-delimited list of entities. See description of the Entities parameter for details.

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: public

Parameter	Datatype	Description
Parameter	String	Parameter name
Value	String	Parameter value

Parameter	Datatype	Description
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 returned by EN^XPAR call.

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

2.7.13 RPC: CIAVMRPC GETVAR

Scope: public

Parameter	Datatype	Description
Variable Name	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: public

Parameter	Datatype	Description
Variable Name	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>	Integer	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: public

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: public

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	FileMan date/time value corresponding to input, or null if input was invalid.

Makes a call to ^%DT to convert the text input to a FileMan date/time value.

2.7.17 $$VERCMP^CIAVMRPC

Scope: public

Parameter	Datatype	Description
Version1	String	First version number
Version2	String	Second version number
Level	Integer	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: public

Parameter	Datatype	Description
Identifier	String	Optional unique identifier for temporary global.
<return value>	String	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 may be specified to override the default warning message that is sent to users.
Disable Logins	Boolean	If true, further application logins are prohibited. This takes effect immediately, regardless of the delay setting.
Users	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 may 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 may 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 targets, 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	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 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.
Microsoft .NET Framework	2.0	While the VueCentric Framework does not depend on the .NET Framework, it does require that any .NET embedded components comply with at least v2.0.
Kernel	8.0	Numerous dependencies throughout.

Package	Version	Dependency
PIMS	5.3	The patient and encounter context support code make a number of calls to PIMS routines.
Toolkit	7.3	Numerous dependencies throughout.
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.

Key	Description
CIAV SITE MANAGER	Grants access to the VueCentric System Management Utility.

2.12 Exported Protocols

Protocol	Type	Description
CIAV LOCK EVENT	Extended Action	This protocol is fired when the VueCentric Framework locks a user session.
CIAV LOGIN EVENT	Extended Action	This protocol is fired when the VueCentric Framework creates a user session.
CIAV LOGOUT EVENT	Extended Action	This protocol is fired when the VueCentric Framework ends a user session.
CIAV SUBSCRIBE EVENT	Extended Action	This protocol is fired when a VueCentric component subscribes to an event.
CIAV UNSUBSCRIBE EVENT		
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
CIAVM COUNTDOWN INTERVAL		Integer	User, System	This value is the number of seconds to display the timeout warning dialog.
CIAVM 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.
CIAVM DEFAULT TEMPLATE		Pointer (19930.3)	User, Class, Service, Location, Division, System	Specifies the template to be loaded if no specific template is requested.
CIAVM DISABLE CCOW		Boolean	User, System	If yes, CCOW support is disabled in the client application even if a CCOW context manager has been installed.
CIAVM OBJECT FAVORITES	Integer	String	User	Used to store personal favorites for the add object dialog in the VIM.

Parameter	Instance Type	Value Type	Precedence	Description
CIAVM PRIMARY TIMEOUT		Integer	User, System	This value sets the number of seconds that the application will remain idle before initiating an application lockout countdown.
CIAVM SECONDARY TIMEOUT		Integer	User, System	This value sets the number of seconds that the application will remain idle before initiating a logout 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 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 may be the user's private configuration or one that is defined for a specific user role or function. Specific templates may 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 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 (for example, the color of the component as set by the user). The VIM then registers the component with the 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 may 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 may 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.

Parameter	Value	Description
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 may 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.
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	<servername>	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.

Parameter	Value	Description
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 may also have application in environments that use system administrator tools to push software updates to workstations.
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.
showPBM		Shows the Powered by Medsphere icon in the status panel.
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 may view this data. The VIM intercepts the TRACE event and adds the information to a cumulative event log.

Parameter	Value	Description
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

Property	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.

Property	Datatype	Access	Description
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 may 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.
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 may be invoked to ensure 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 may be invoked to ensure 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 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 pop-up 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 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 may 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

Property	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. 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

Property	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
DebugMode	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.

Property	Datatype	Access	Description
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 may 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_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	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>	Boolean	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 datatype.

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.

Parameter	Datatype	Description
Parameters	OleVariant	Parameters to be passed to the remote procedure.
<return value>	Integer	Output of remote procedure call as a 32-bit integer 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 may 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 CallRPCText

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 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 may 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 may 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. Currently, the CSS implements a single, shared instance of the RPC broker. This means that only a single host connection may be active at a given time. This restriction may 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 may 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_SessionEvents	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 to be revoked.
Callback	ICSS_SessionEvents	A reference to the interface that was specified in the original EventSubscribe 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. 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 service identified by ProgID or nil if the service could not be located.

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 may subscribe to an event by simply implementing the corresponding interface. The *RegisterObject* procedure takes care of connecting the object's event interface (for example, event sink) to the event source.

This procedure also adds the object reference to a registered object table so that other objects may 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>	String	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>	Where <object> is the name of an object, <prop/meth> is the name of a property or method within that object, and <arg1>…<argn> is an optional argument list. An object name may 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). <arg1>…<argn> is an optional list of comma-delimited arguments if required by the method call. Arguments themselves may be replaceable parameters.
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
TraceClass	String	This is the class to which the entry belongs. For example, RPC or EVENT.

Parameter	Datatype	Description
TraceType	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 may 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 requesting 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.
EventStub	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.
ErrorText	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.

2.16.2.4 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.

Property	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 may be referenced in the ReplaceParams method of the Session object.

Property	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.

2.16.2.4.1 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.

2.16.2.4.2 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.

2.16.2.4.3 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.

2.16.2.4.4 Reset

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

2.16.2.4.5 SetContext

Parameter	Datatype	Description
Context	String	Context the object is to assume, in CCOW format.

Parameter	Datatype	Description
<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.

2.16.2.5 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 wanting 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 (for example, the ICSS_PatientEvents interface if the patient context object is the target).

Since 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:

2.16.2.5.1 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 wants 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 who want to participate in the decision to change the context for the corresponding context object may respond to this event by either prompting the user to save pending changes, warning the user that changes may be lost, and/or contesting the context change by returning a non-null value to the caller.

> **Note:** 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.

2.16.2.5.2 Committed

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

2.16.2.5.3 Canceled

The CSS invokes this parameter-less 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 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

Property	Datatype	Access	Description
NoUpdates	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.
Component[Index]	ICMS_Component	R	Array of all components registered within the Object Registry.
ComponentCount	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.
RaiseException	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 may 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

Property	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 may coexist in an application.

Property	Datatype	Access	Description
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.
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	Array of serializable properties.
PropSaveCount	Integer	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

Parameter	Datatype	Description
<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 may 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.

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.

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 (for example, 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
- 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 may 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:

Property	Datatype	Description
ALIGN	Integer	The alignment of the control relative to its parent. May 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. May 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. May 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
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 privilege.
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.

2.16.5.1.1 TObjectContainer

In addition to the properties of the container itself, properties of the contained object may also be saved. These properties may 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.

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

2.16.5.1.2 TPanelEx

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

2.16.5.1.3 TScrollBoxEx

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.

2.16.5.1.4 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.

2.16.5.1.5 TToolbarEx

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

2.16.5.1.6 TPageControlEx

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

Property	Datatype	Description
FIXEDWIDTH	Boolean	If true, all tabs maintain the same width. If false, tab widths are sized to match their caption widths.

Property	Datatype	Description
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.
REVERSETABS	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 corner

2.16.5.1.7 TTabSheetEx

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

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

2.16.5.1.8 TSplitterPaneEx

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

Property	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.

2.16.5.1.9 TPaneEx

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

Property	Datatype	Description
PANEINDEX	Integer	The relative position of the pane within the parent control.

2.16.5.1.10 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.

Property	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.

2.16.5.1.11 TTreePaneEx

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

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

2.16.5.1.12 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.

Property	Datatype	Description
CAPTIONCOLOR	Integer	The color of the pane view's caption bar.

Property	Datatype	Description
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 style

2.16.5.1.13 TGroupPaneEx

These are the individual panes within the TGroupBarEx component.

Property	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.

2.16.5.1.14 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:

Property	Datatype	Description
ICON	Integer	The index of a custom icon that is displayed next to the menu item.

Property	Datatype	Description
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.

2.16.5.1.15 XML Representation

```
<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>
</Template>
```

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).

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 would appear something similar to Figure 2-32.

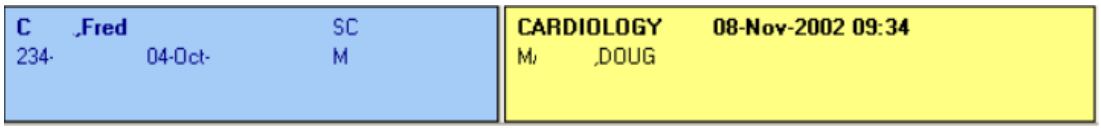


Figure 2-32: Parent-Child Relationships in the Visual Interface

2.16.6 Object Repository

The Object Repository provides a centralized location for storing the most up-to-date versions of VueCentric components. The Object Repository may 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 may 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
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 Figure 3-1).

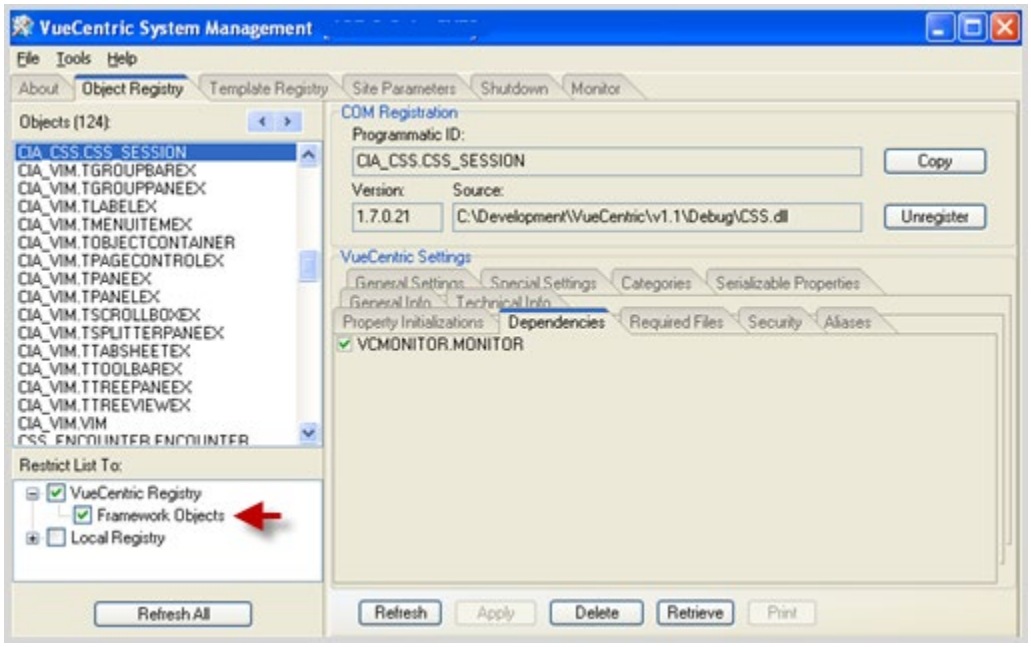


Figure 3-1: Framework Objects Option Button

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

Property	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 data>	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
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 box, enabling 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 box, enabling 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.

4.15.13 DateSelect2

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.
AllowFutureDates	Boolean	If true, future dates can be selected
<return value>	Boolean	False if dialog was cancelled.

Invokes the Date Selection dialog box, enabling the user to input a single date.

4.15.14 DateRange2

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 AllowFutureDates: Boolean
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 box, enabling the user to input a date range.

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 Architecture and Business Process Overview

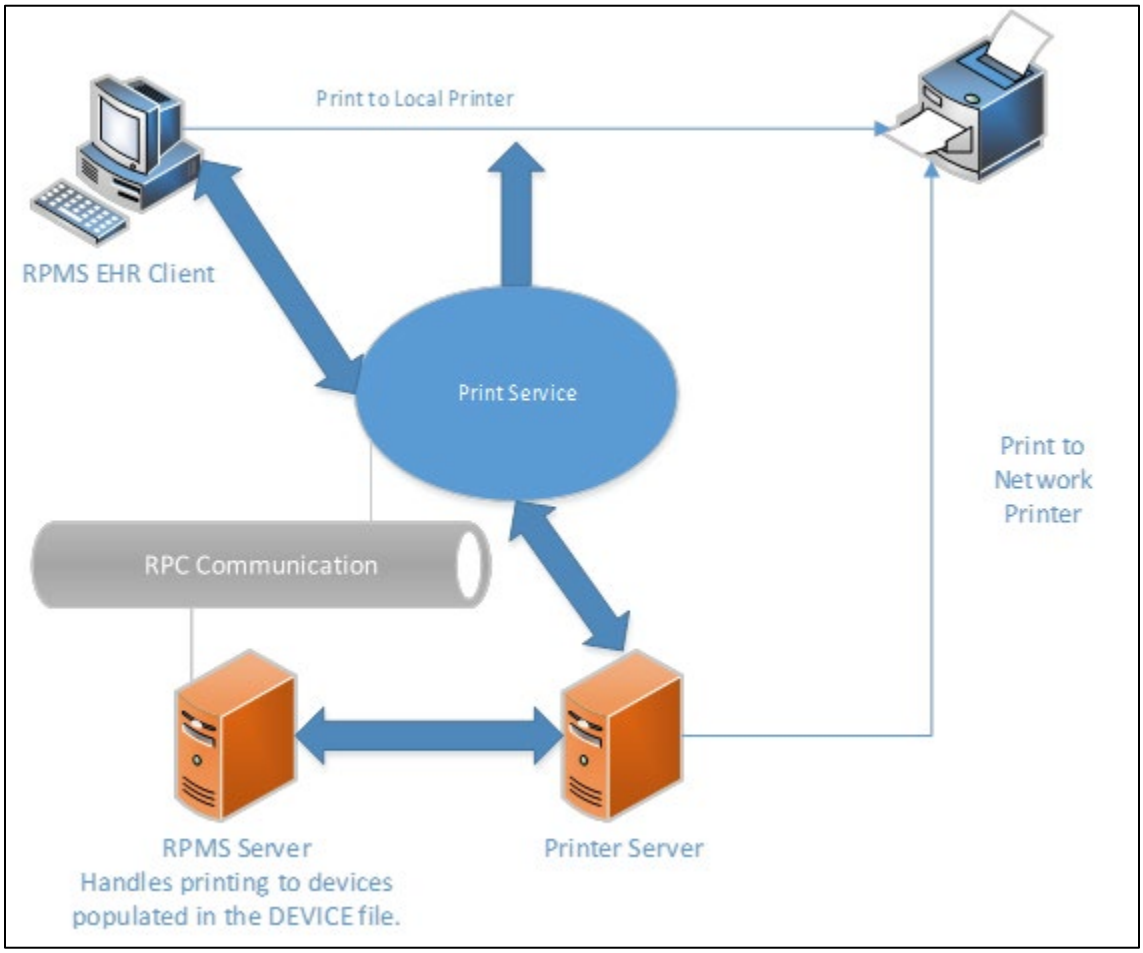


Figure 5-1: Architecture and business process overview

5.3 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	VCPRINT.VCPRINTX

Entity	Value
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.

5.4 Routine Descriptions

Routine	Description
CIAVUTIO	Device I/O support

5.5 File List

None.

5.6 Cross References

None.

5.7 Exported Options

None.

5.8 Exported Security Keys

None.

5.9 Exported Protocols

None.

5.10 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 may 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 may contain replaceable parameters.
CIAVUTIO DEFAULT PRINTER		String	User, Location	Specifies the default printer for user or location.
CIAVUTIO LOCAL PRINTER		Boolean	User, Location, System	If set to yes, the EHR will display the Windows standard printer selection dialog instead of the RPMS/VistA printer selection dialog. 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.

5.11 Exported Mail Groups

None.

5.12 Callable Routines

None.

5.12.1 OUTPUT^CIAVUTIO

Scope: public

Parameter	Datatype	Description
Execute	String	M code to be executed for report generation.

Parameter	Datatype	Description
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.

5.12.2 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 may 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.

5.12.3 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.

5.12.4 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.

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

5.12.5 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.

5.12.6 RPC: CIAVUTIO PRTISLCL

Scope: private

Parameter	Datatype	Description
Device	String	Specifier for the default printer device.
<return value>	Boolean	Returns Boolean flag

Indicates if the printer returned from PRTGETDF^CIAVUTIO call is the local default.

5.13 External Relations

None.

5.14 Internal Relations

None.

5.15 Archiving and Purging

There are no archiving or purging requirements within this software.

5.16 Components

This component supports the following properties and methods:

5.16.1 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.

5.16.2 ClosePreview

Scope: public

Parameter	Datatype	Description
Handle	Integer	Unique preview handle.

Closes the preview dialog identified by the specified handle.

5.16.3 FindPreview

Scope: public

Parameter	Datatype	Description
Handle	Integer	Unique preview handle.

Parameter	Datatype	Description
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.

5.16.4 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 may be displayed within a header or footer.
Header	String	This may 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.

5.16.5 Preview

Scope: public

Parameter	Datatype	Description
Text	String	This is the raw text of the report to be formatted.

Parameter	Datatype	Description
Title	String	This is optional title text which may be displayed within a header or footer.
Header	String	This may 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.

5.16.6 Preview2

Scope: public

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.

5.16.7 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 may be displayed within a header or footer.
Header	String	This may 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.

5.16.8 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 may be displayed within a header or footer.
Header	String	This may 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. May 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 Printer Selection Dialog to allow the user to choose the desired output device. It differs from the Print procedure in that one can contain the type of selectable printers and in that it returns an indication of whether or not the print operation was completed.

5.16.9 Reset

Scope: public

Closes all open preview dialogs.

5.16.10 SelectPrinter

Scope: public

Parameter	Datatype	Description
PrinterTypes	Integer	Determines whether selectable printers are server-based, client-based, or both. May 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.

5.16.11 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 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 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.

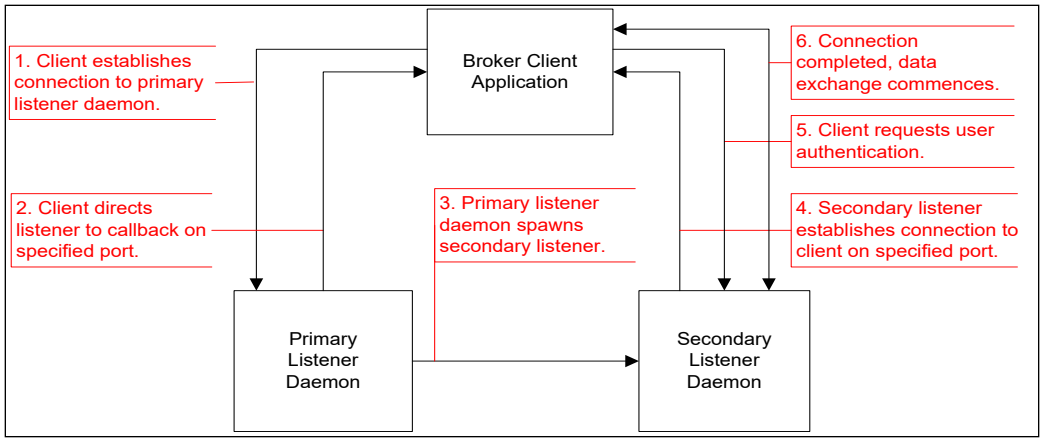


Figure 6-1: Medsphere RPC Broker

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.

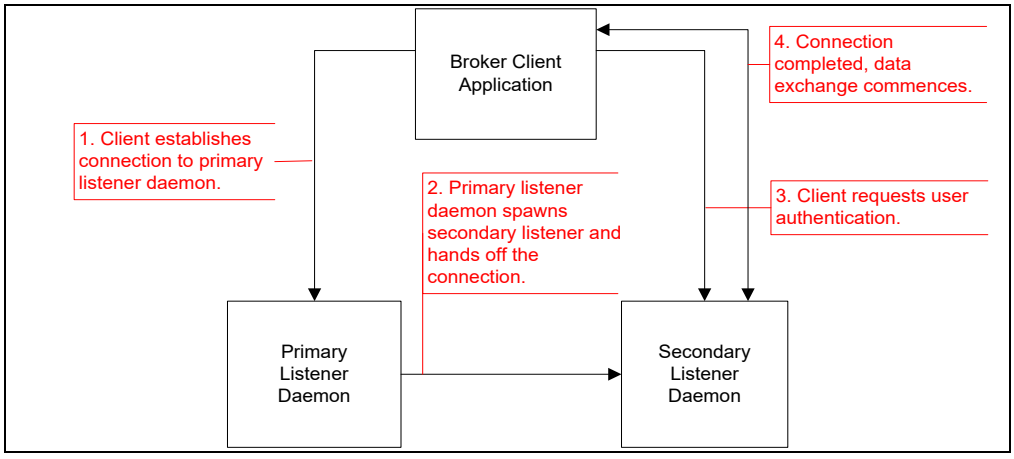


Figure 6-2: Medsphere RPC Broker Strategy

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 may 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.
CIANBLOG	Activity log support
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.
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.

Field Name	#	Datatype	Indexes	Description
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.
EVENT PROTOCOL	7	Pointer (#101)		Associated protocols must be of type Extended Action with names starting with 'CIAV'.
DISPLAY LOGIC	10	M code		Logic to display the event stub data in the log viewer. This feature is not currently implemented.
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.

Field Name	#	Datatype	Indexes	Description
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.

6.5.5 CIA ACTIVITY LOG File (#19941.24)

This file provides a means to capture session activity such as login, logout, events and RPC calls.

Field Name	#	Datatype	Indexes	Description
SESSION ID	.01	Text	B – Standard	This is the VueCentric session ID
USER	1	Pointer	BUSER	This is the user whose session generated the entry. Points to file 200.
WORKSTATION ID	2	Text	BWKID	This is the workstation name running the VueCentric session
LOGIN TIME	3	Date/Time	BLOGIN	This is the time that the session logged in to the database
LOGOUT TIME	4	Date/Time	BLOGOUT	This is the time that the session logged out of the database
DIVISION	5	Pointer	BDIV	This is the division that the session was logged into.
ACTIVITY	10	Date/Time (Multiple)		Holds the event activity
DATE/TIME	.01	Date/Time	B-Standard	Holds the activity date/time
TYPE	1	Set		Holds type of event: 1:RPC, 2:EVENT
NAME	2	Text		Holds name of event
LOG	10	WP		Holds information related to event

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

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 ACTIVITY LOGGING	Pointer to Option File	Yes/No	System, Division, User	Allows enabling of logging for a given broker context menu option.
CIANB ACTIVITY RETENTION		Numeric (1:9999 999 days)	System	Controls how long entries in the CIA ACTIVITY LOG file are retained.

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 may 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.

6.12.1.4 STARTALL^CIANBLIS

Scope: public

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

6.12.1.5 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.

6.12.1.6 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 may 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

Parameter	Datatype	Description
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. May 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.

Parameter	Datatype	Description
<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 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 of one or more state variables.

6.12.3 Event Management

None.

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 EVENTIEN^CIANBEVT

Scope: public

Parameter	Datatype	Description
TYPE	String	Event type
<return value>	Numeric	IEN of the root event type

Returns the IEN represent the root event for the passed in type.

6.12.3.4 EVENTNAM^CIANBEVT

Scope: public

Parameter	Datatype	Description
IEN	Numeric	IEN of CIA EVENT TYPE
<return value>	Numeric	Name of event type

Returns the name associated with the event type IEN.

6.12.3.5 KEYCHECK^CIANBEVT

Scope: public

Parameter	Datatype	Description
TYPE	String	Event type name
SB	Set	Defines type of security keys to check. 20:Publication ; 21:Subscription

Parameter	Datatype	Description
<return value>	Boolean	Result of user security key check

Returns true if the user does not have required keys.

6.12.3.6 HASKEY^CIANBEVT

Scope: public

Parameter	Datatype	Description
KEY	String/Numeric	Lookup on key name occurs if IEN is not passed in.
<return value>	Boolean	Result of key lookup on logged in user

Returns true if user has key.

6.12.3.7 EVENTIEN^CIANBEVT

Scope: public

Parameter	Datatype	Description
TYPE	String	Event type
<return value>	Numeric	IEN of the root event type

Returns the IEN represent the root event for the passed in type.

6.12.3.8 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.

6.12.3.9 QUEUE^CIANBEVT

Scope: public

Parameter	Datatype	Description
TYPE	String	Event type to send
STUB	String	Event stub

Parameter	Datatype	Description
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.

6.12.3.10 $$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.

6.12.3.11 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.

6.12.3.12 TASKPRG^CIANBEVT

Scope: public

Tasks the event log purge to run in the background.

6.12.3.13 UNSUBALL^CIANBEVT

Scope: public

Withdraws all subscriptions for the session.

6.12.4 Miscellaneous

None.

6.12.4.1 CLEANUP^CIANBUTL

Scope: public

Purges data for all inactive sessions. Sessions which terminate abnormally may 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.

Parameter	Datatype	Description
DLG (by reference)	String array	Receives the text of the requested dialog.
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.

6.13 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.

6.14 Internal Relations

None.

6.15 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.

6.16 Components

None.

6.16.1 Delphi Component

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

6.16.1.1 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 that follow.
Picture	TPicture	RW	Graphic image used as background for the authentication dialog.

Property	Datatype	Access	Description
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.
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 may 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 may 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.

Event	Parameters	Description
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).

6.16.1.2 Call

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

6.16.1.3 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.

6.16.1.4 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.

6.16.1.5 CallRPCAbort

Parameter	Datatype	Description
Handle	Integer	The remote procedure name.
<return value>	Boolean	Returns success of call.

Stops a previously sent asynchronous remote procedure call.

6.16.1.6 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.

6.16.1.7 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.

6.16.1.8 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.

6.16.1.9 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.

6.16.1.10 Disconnect

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

6.16.1.11 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.

6.16.1.12 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.

6.16.1.13 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 may interfere with the current thread.

6.16.1.14 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.

6.16.1.15 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.

6.16.1.16 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.

6.16.1.17 Unlock

Removes a lock established by the Lock method.

6.16.2 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 that may 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.

6.16.2.1 Assign

Parameter	Datatype	Description
Source	TPersistent	The object to be copied. This may 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 (for example, the first TStrings entry, index 0, corresponds to a subscript value of one on the server).

6.16.2.2 Clear

The internal value list is cleared.

6.16.2.3 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.

6.16.2.4 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.

6.16.2.5 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.

6.16.2.6 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.

6.16.2.7 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.

6.16.2.8 SortSubscripts

Sorts subscript list in MUMPS collation order.

6.16.2.9 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. The Value property is stored internally with a null subscript value. Therefore, the subscript value returned can be null.

6.16.2.10 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. 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 Architecture and Business Process Overview

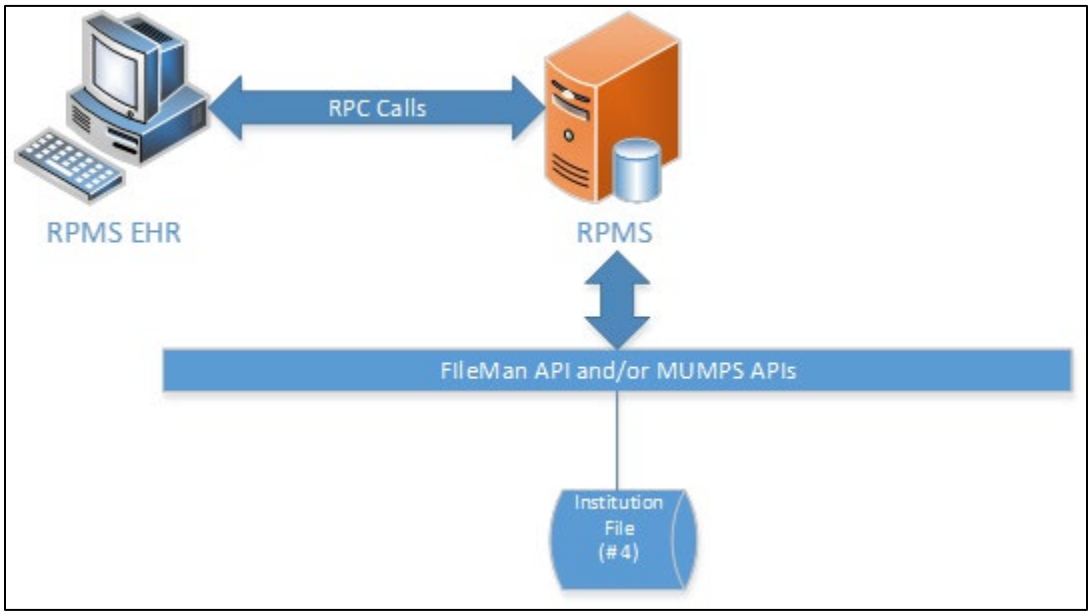


Figure 7-1: Architecture and business process overview

7.3 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	CSS_SITE.SITE
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

Entity	Value
All Keys Required	no
Hidden from Property Editor	no
Side-by-Side Versioning	yes
Service	yes
.Net Component	no
Associated Build	BEHO*1.1*007003

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

7.4 Routine Descriptions

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

Routine	Description
BEHOSICX	Site context support

7.5 File List

None.

7.6 Cross References

None.

7.7 Exported Options

None.

7.8 Exported Security Keys

None.

7.9 Exported Protocols

None.

7.10 Exported Parameters

None.

7.11 Exported Mail Groups

None.

7.12 Callable Routines

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

7.12.1 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 Agency Code

Returns context data about the current site.

7.13 External Relations

None.

7.14 Internal Relations

None.

7.15 Archiving and Purging

There are no archiving or purging requirements within this software.

7.16 Components

This component supports the following properties:

Property	Datatype	Access	Description
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.
Agency Code	String	R	Agency Code 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 site.

8.2 Architecture and Business Process Overview

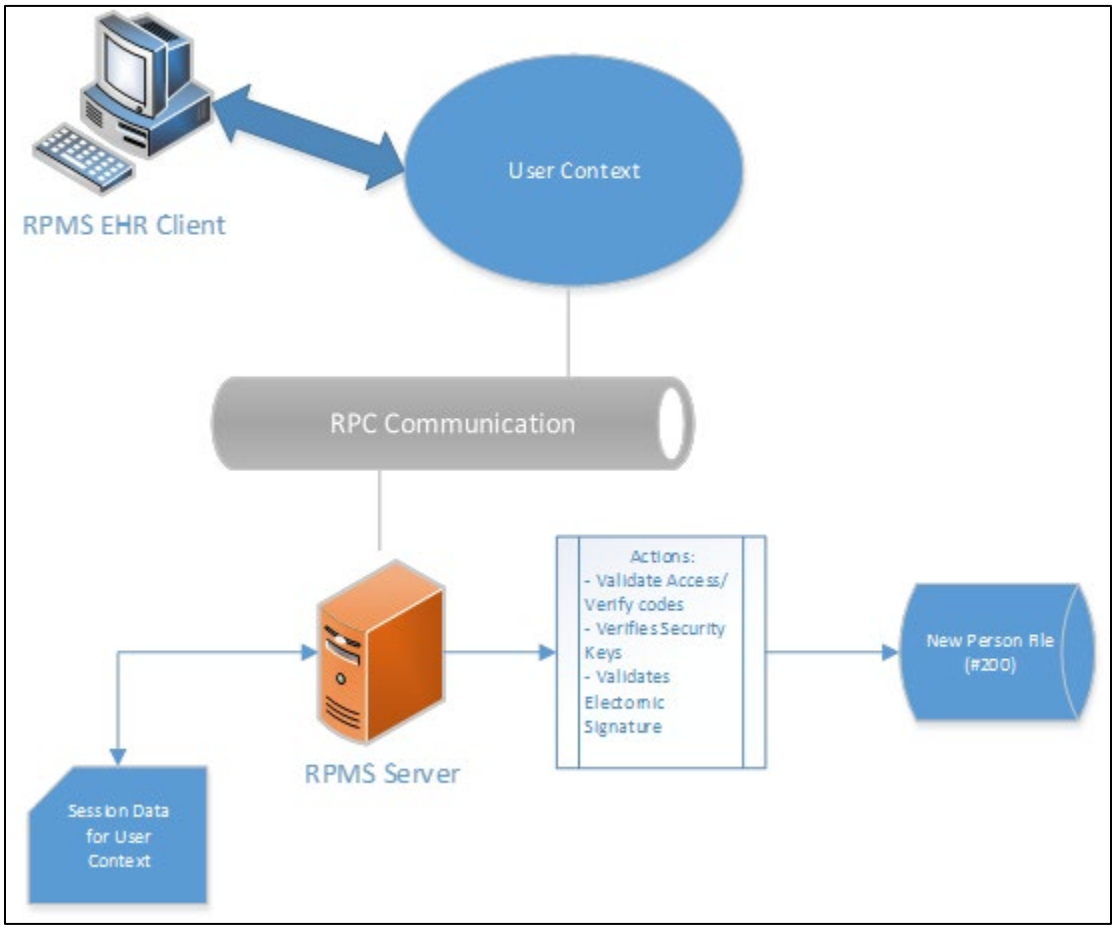


Figure 8-1: Architecture and business process overview

8.3 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	CSS_USER.USER
Class Identifier	{C8C185BB-360C-4FAD-BCB7-7AE680550423}

Entity	Value
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*006003

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

8.4 Routine Descriptions

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

Routine	Description
BEHOUSCX	User context support.

8.5 File List

None.

8.6 Cross References

None.

8.7 Exported Options

None.

8.8 Exported Security Keys

None.

8.9 Exported Protocols

None.

8.10 Exported Parameters

None.

8.11 Exported Mail Groups

None.

8.12 Callable Routines

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

8.12.1 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.

8.12.2 $$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).

8.12.3 $$ISPROV^BEHOUSCX

Scope: private

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

Returns true if the current user possesses the provider key.

8.12.4 $$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.

8.12.5 RPC: BEHOUSCX HASKEYS

Scope: public

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

8.12.6 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 criteria in the format: <ien>^<name>

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

8.12.7 $$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.

8.12.8 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.

8.12.9 RPC: BEHOUSCX VALINSIG

Scope: public

Parameter	Datatype	Description
Esig	String	Encrypted electronic signature code.
<return value>	String	1 if the specified electronic signature code compiles with the input transform for the field.

Verifies than an electronic signature code complies with the input transform.

8.12.10 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.

8.12.11 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.

8.12.12 RPC: BEHOUSCX STORESIG

Scope: public

Parameter	Datatype	Description
Esig	String	Encrypted electronic signature code.
<return value>	String	0 if the specified electronic signature code stores for the user.

Stores an electronic signature code.

8.12.13 RPC: BEHOUSCX HASESIG

Scope: public

Parameter	Datatype	Description
<return value>	Boolean	True if the logged in user has an electronic signature code.

Determines if the current user has an electronic signature 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.

Property	Datatype	Access	Description
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.

8.16.5 ESigModify

Parameter	Datatype	Description
N/A	N/A	N/A

Method call that presents the user with a dialog to enter or change the electronic signature code.

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 Architecture and Business Process Overview

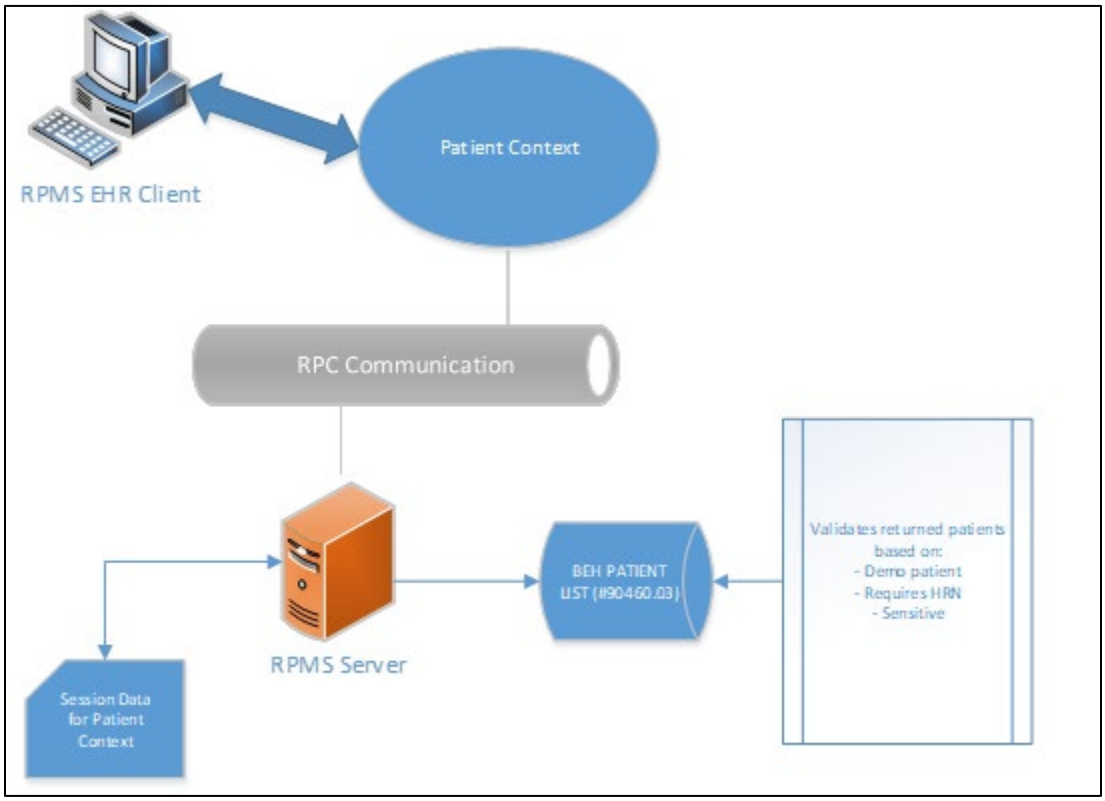


Figure 9-1: Architecture and business process overview

9.3 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	CSS_PATIENT.PATIENT
Class Identifier	{8955EBC9-3342-40B1-8295-FE20415CCA4D}
Image File	CSSPatient.dll

Entity	Value
Property Initializations	PHOTOMASK=@BEHOPTCX PHOTO MASK LOOKUPIDS=NNNN;NNNNN;NNNNNN;TNNNNN;tNNNNN^BEHOPTPL HRNLKP|`N;`NN;`NNN;`NNNN;`NNNNN;`NNNNNN;`NNNNNNN;`NNNNNNNN^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*004011

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

9.4 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 BEHOPTP1 BEHOPTP2 BEHOPTP3	Patient list support
BEHOPTC1	Patient Inquiry
BEHOPTPC	Primary provider data access

9.5 File List

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

9.5.1 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. May 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.

9.6 Cross References

Cross references are described in the preceding section.

9.7 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.
BEHOPTCX IGNORE ALIASES	action	Set parameter to ignore aliases in patient lookup
BEHOPTCX RECALL LAST	action	Set parameter for recalling last patient viewed.
BEHOPTCX REQUIRES HRN	action	Set parameter to require HRN for patient selection
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 PATIENTS	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.

9.8 Exported Security Keys

None.

9.9 Exported Protocols

Protocol	Type	Description
BEHOPTCX ADT EVENT	Action	Passes the ADT event back to the RPMS EHR client.

9.10 Exported Parameters

Parameter	Instance Type	Value Type	Precedence	Description
BEHOPTCX ADT DISPLAY ACTION		Set of Codes (I,P,F)	User, Division, System	I-Do not change Encounter Context P-Prompt user F-Force Encounter Context
BEHOPTCX DEMO MODE		Boolean	User, Division, System	If yes, only demo patients may 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.
BEHOPTCX IGNORE ALIASES		Boolean	Division, System	If yes, aliases encountered in the primary index of the patient file are ignored.
BEHOPTCX LAST PATIENT	Facility	Pointer (#2)	User	Saves the last patient selected for the current facility.
BEHOPTCX RECALL LAST		Boolean	User, Division, System	If yes, the patient context is set to the last patient selected upon startup.
BEHOPTCX REQUIRES HRN		Boolean	System	If yes, patient must have an HRN to be selected.
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.

9.11 Exported Mail Groups

None.

9.12 Callable Routines

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

9.12.1 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.

9.12.2 $$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.

9.12.3 $$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.

9.12.4 RPC: BEHOPTCX FIREVST

Scope: private

Parameter	Datatype	Description
DFN	String	Patient's internal entry number to File 2
<return value>	Boolean	Returns true if the encounter context change was successfully fired.

Fires an Encounter Context change for the patient setting it to any recent ADT changes.

9.12.5 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.

9.12.6 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.

9.12.7 $$ISACTIVE^BEHOPTCX

Scope: private

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number.

Parameter	Datatype	Description
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.

9.12.8 $$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.

9.12.9 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.

9.12.10 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 may 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.

9.12.11 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.

9.12.12 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.
<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.

9.12.13 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.

9.12.14 $$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.

9.12.15 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.

9.12.16 RPC: BEHOPTPC GETBDP

Scope: private

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number.

Parameter	Datatype	Description
<return value>	String List	Returns formatted information about a patient's designated providers.

Retrieves detailed information about a patient's designated providers.

9.12.17 RPC: BEHOPTPC GETCATS

Scope: private

Parameter	Datatype	Description
<return value>	String List	Returns formatted list of designated provider categories.

Retrieves list of designated provider categories.

9.12.18 RPC: BEHOPTPC SETBDP

Scope: private

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number.
TYPE	Pointer (#90360.3)	Internal entry number
PROVIDER	Pointer (#200)	Provider's internal entry number
<return value>	Boolean	Returns true if successful.

Associates a designated provider with a patient.

9.12.19 $$OUTPTPR^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.

9.12.20 $$OUTPTTM^BEHOPTPC

Scope: private

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number.

Parameter	Datatype	Description
<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.

9.12.21 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.

9.12.22 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: <range specifier>^<display text> Where <range specifier> is either "S" for user selectable, or the format: <start>;<end> 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.

9.12.23 RPC: BEHOPTPL DOBLKP

Scope: private

Parameter	Datatype	Description
DOB	String	Date of birth in external format, prefixed with a "B" character.

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 date of birth.

9.12.24 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.

9.12.25 RPC: BEHOPTPL HRNLKP

Scope: private

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

9.12.26 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.

9.12.27 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.

9.12.28 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.
<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.

9.12.29 RPC: BEHOPTPL LISTPTS

Scope: private

Parameter	Datatype	Description
LIST	Pointer (#19930.4)	Internal entry number of the patient list type.

Parameter	Datatype	Description
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.

9.12.30 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.

9.12.31 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.

9.12.32 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 may be managed by the user, this RPC provides support for list management activities.

9.12.33 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.

9.12.34 $$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.

9.13 External Relations

Entity	Name	Description
Package	PIMS	version 5.3

9.14 Internal Relations

None.

9.15 Archiving and Purging

There are no archiving or purging requirements within this software.

9.16 Components

This component supports the following properties and methods:

9.16.1 Properties

Patient identifier properties that are writable (*Handle* and *ICN*) may 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.

Property	Datatype	Access	Description
DOD	Date	R	Date of the patient's death.
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.

Property	Datatype	Access	Description
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.

9.16.2 Clear

Clears the patient context.

9.16.3 Detail

Parameter	Datatype	Description
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.

9.16.4 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 necessary 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 (for example, placing orders). Other operations require both (for example, 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 Architecture and Business Process Overview

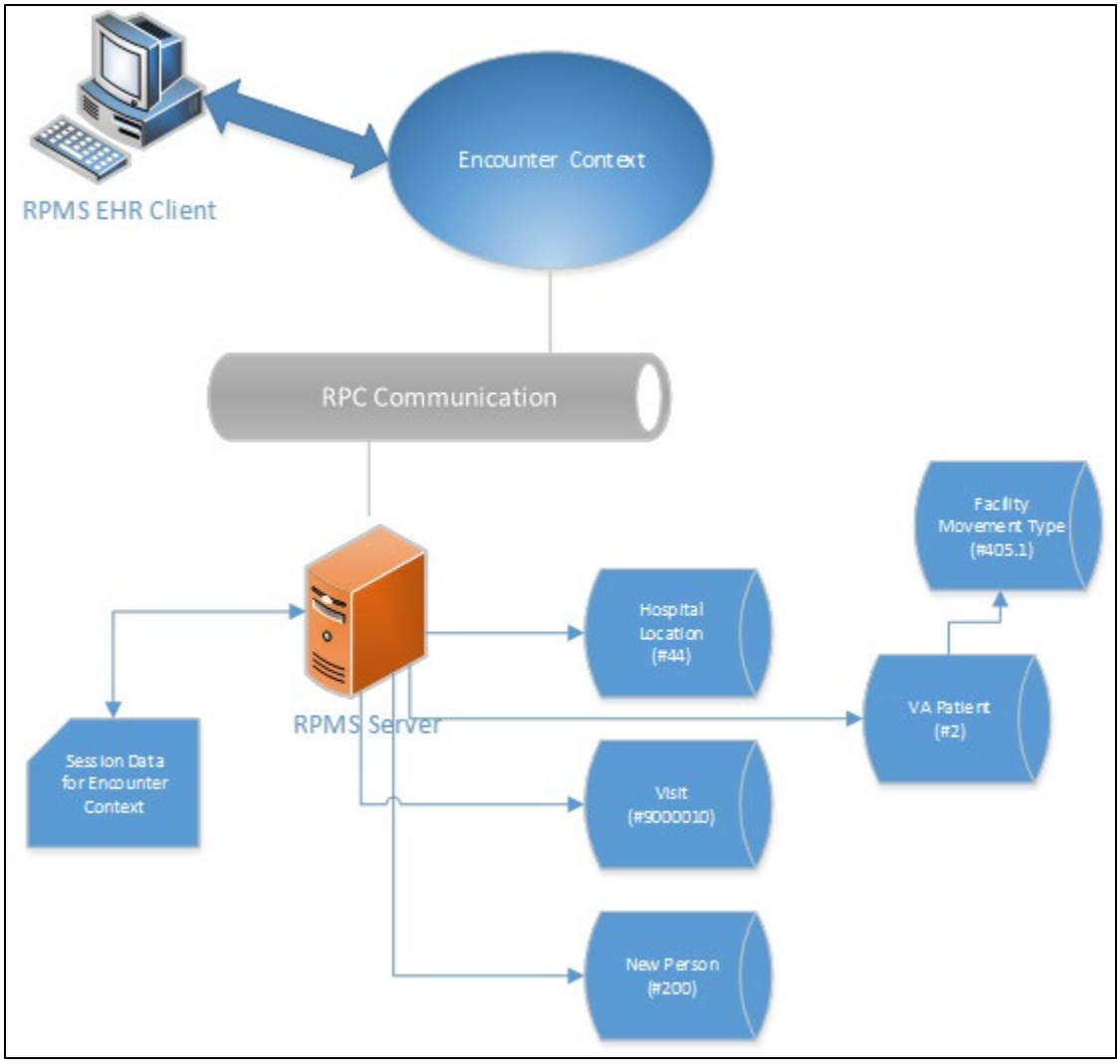


Figure 10-1: Architecture and business process overview

10.3 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	CSS_ENCOUNTER.ENCOUNTER
Class Identifier	{5C87BBCA-BC10-462B-9DB7-6F1E886C3D3F}
Image File	CSSEncounter.dll
Property Initializations	none
Serializable Properties	none
Required Files	Interop.CSS_Encounter.dll

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	yes
.Net Component	no
Associated Build	BEHO*1.1*005012

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

10.4 Routine Descriptions

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

Routine	Description
BEHOENCX BEHOENC1	Encounter context support.
BEHOENIN	Installation support.
BEHOENPC BEHOENP1	PCC data management.
BEHOENPP BEHOENPR BEHOENPS BEHOENPV BEHOENP2	Encounter summary report.

10.5 File List

None.

10.6 Cross References

None.

10.7 Exported Options

Option	Type	Description
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.

10.8 Exported Security Keys

None.

10.9 Exported Protocols

None.

10.10 Exported Parameters

Parameter	Instance Type	Value Type	Precedence	Description
BEHOENCX CREATE VISIT		Boolean	User, Class, Service, Location, Division, System	If yes, the user may create visits. This enables the new visit tab of the encounter selection dialog.
BEHOENCX DETAIL REPORT		Free Text	User, Division, Service, System	M Code to generate a visit detail report.
BEHOENCX PROV ENC FETCH		Boolean	Division, System	Specifies that the current user should be specified as the visit provider.

Parameter	Instance Type	Value Type	Precedence	Description
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 may 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.

Parameter	Instance Type	Value Type	Precedence	Description
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.
BEHOENPS SUMMARY END		Date/Time	System	Holds the end date for the current run of the report. Is not user definable.
BEHOENPS SUMMARY START		Date/Time	System	Holds the start date for the current run of the report. Is not user definable.

10.11 Exported Mail Groups

None.

10.12 Callable Routines

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

10.12.1 $$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.

10.12.2 $$ADDPRV^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.

10.12.3 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.

10.12.4 $$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.

Parameter	Datatype	Description
<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.

10.12.5 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.

10.12.6 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.

10.12.7 RPC: BEHOENCX CHKVISIT

Scope: public

Parameter	Datatype	Description
VISIT	Pointer (#9000010)	Visit internal entry number
<return value>	String List	Notification of sending of OE/RR POV/EMC Notifications

Returns indication that POV and EMC notifications have been generated when needed for the visit.

10.12.8 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.

10.12.9 RPC: BEHOENCX ENINQ

Scope: public

Parameter	Datatype	Description
VISIT	Pointer (#9000010)	Visit internal entry number
<return value>	String List	Detail Report for the visit.

Returns the detail visit report.

10.12.10 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.

Parameter	Datatype	Description
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.

10.12.11 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. May 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
<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.

10.12.12 $$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.

10.12.13 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.

10.12.14 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.

10.12.15 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.

10.12.16 $$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.

10.12.17 RPC: BEHOENCX HOSPLOC

Scope: public

Parameter	Datatype	Description
FROM (Optional)	Pointer (#44)	Hospital Location file internal entry number
DIR (Optional)	Numeric	-1,1 representing which direction to search
MAX (Optional)	Numeric	Represents the number of entries to return (Default: 44)
TYPE (Optional)	String	When present, restricts entries to those have this value in the TYPE field.
START (Optional)	Date	Returns locations with appointment dates after this value.
END (Optional)	Date	Returns locations with appointment dates before this value.
<return value>	String List	List of Hospital Locations matching the input criteria

Returns a list of hospital locations.

10.12.18 RPC: BEHOENCX ISSTOPCD

Scope: public

Parameter	Datatype	Description
LOCIEN	Pointer (#44)	Hospital Location internal entry number
STOP CODE IEN	Pointer (#40.7)	Clinic Stop internal entry number
<return value>	Boolean	Returns true if location ien is associated with clinic stop code.

Returns true if location is associated with clinic stop code.

10.12.19 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.

10.12.20 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.

10.12.21 $$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.

10.12.22 $$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.

10.12.23 RPC: BEHOENCX UPDPRV

Scope: public

Parameter	Datatype	Description
DFN	Pointer (#2)	Patient's internal entry number
VSTR	String	Visit String representing the visit
PRV	String List	String List of Provider IENs (File #200) set to the type of provider ('P': Primary, 'S':Secondary)
<return value>	String	Returns status of update

Update the primary provider on a visit.

10.12.24 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.

10.12.25 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.

10.12.26 $$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.

10.12.27 $$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.

10.13 External Relations

Entity	Name	Description
Package	PIMS	version 5.3

10.14 Internal Relations

None.

10.15 Archiving and Purging

There are no archiving or purging requirements within this software.

10.16 Components

This component supports the following properties and methods:

10.16.1 Properties

Encounter identifier properties that are writable (*Handle, VisitID,* and *VisitStr*) may 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 *Service Category* field of the *Visit* file.
VisitID	String	RW	The unique identifier for the encounter. If set, the context is set to the encounter represented by that visit identifier.

Property	Datatype	Access	Description
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.

10.16.2 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.

10.16.3 Prepare

Parameter	Datatype	Description
OptionFlags	Integer	Sets various options for the function. May 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.

10.16.4 SelectLocation

Parameter	Datatype	Description
LocationType	Enum	May be one of: ltAll (0)=All locations ltOutpatient (1)=Outpatient locations ltInpatient(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

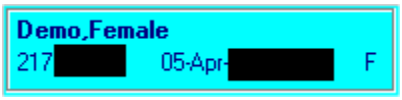


Figure 11-1: Patient Identification header

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 Architecture and Business Process Overview

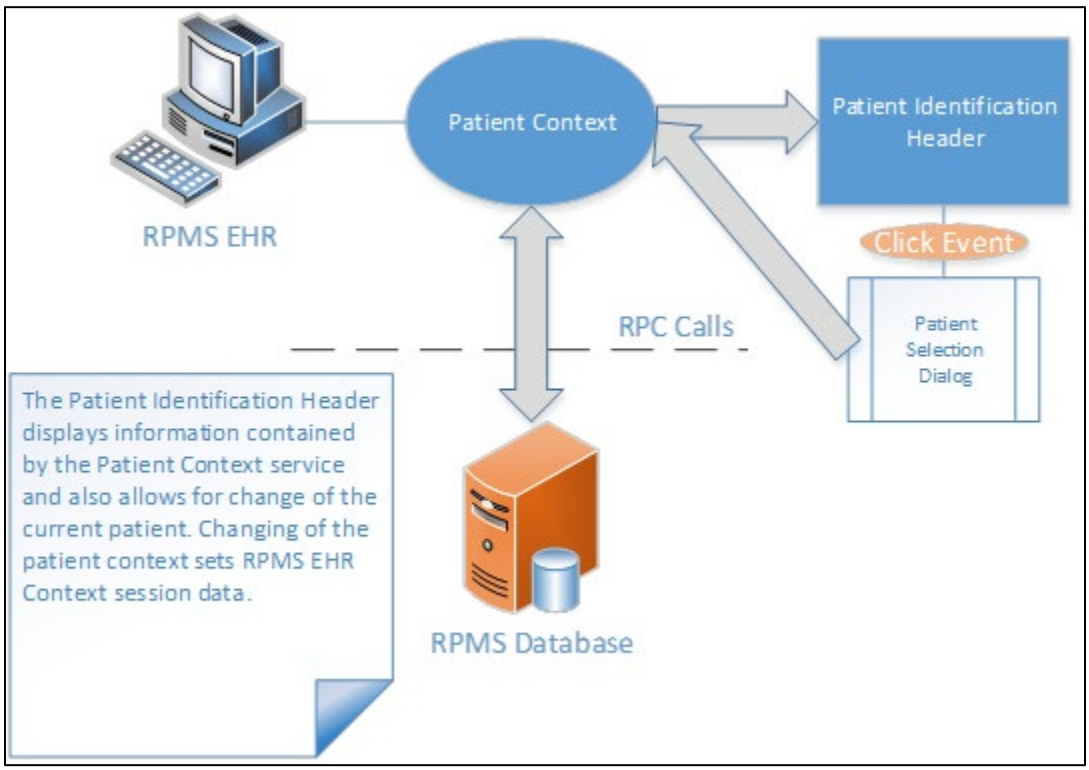


Figure 11-2: Architecture and business process overview

11.3 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHPATIENTID.PATIENTID
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.

11.4 Routine Descriptions

None.

11.5 File List

None.

11.6 Cross References

None.

11.7 Exported Options

None.

11.8 Exported Security Keys

None.

11.9 Exported Protocols

None.

11.10 Exported Parameters

None.

11.11 Exported Mail Groups

None.

11.12 Callable Routines

None.

11.13 External Relations

None.

11.14 Internal Relations

Entity	Name	Description
Component	Patient Context Object	Uses support APIs.

11.15 Archiving and Purging

There are no archiving or purging requirements within this software.

11.16 Components

This component supports the following properties:

11.16.1 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. May 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 may override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPFILE	String	RW	Sets the name of the help file associated with the component.

Property	Datatype	Access	Description
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 may attain.
MINWIDTH	Integer	RW	Sets the minimum width, in pixels that the component may 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

Figure 12-1: Encounter Information

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 Architecture and Business Process Overview

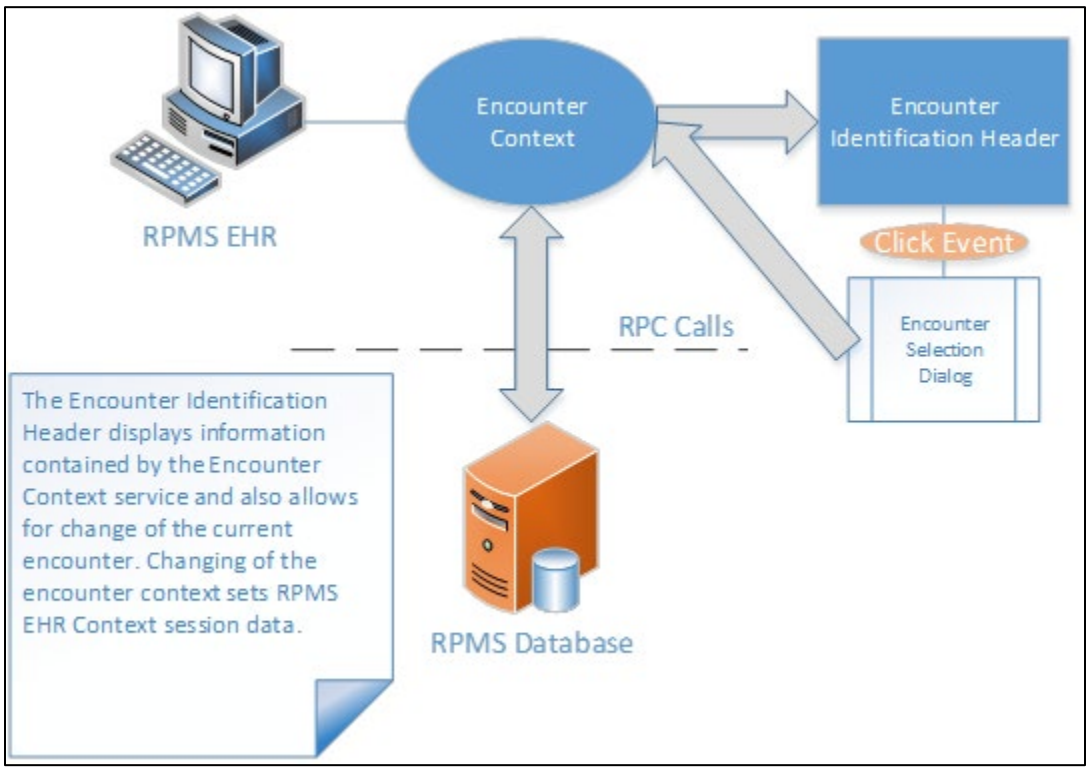


Figure 12-2: Architecture and business process overview

12.3 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHENCOUNTERINFO.ENCOUNTERINFO
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*031004

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

12.4 Routine Descriptions

None.

12.5 File List

None.

12.6 Cross References

None.

12.7 Exported Options

None.

12.8 Exported Security Keys

None.

12.9 Exported Protocols

None.

12.10 Exported Parameters

None.

12.11 Exported Mail Groups

None.

12.12 Callable Routines

None.

12.13 External Relations

None.

12.14 Internal Relations

Entity	Name	Description
Component	Encounter Context Object	Uses support APIs.

12.15 Archiving and Purging

There are no archiving or purging requirements within this software.

12.16 Components

This component supports the following properties:

12.16.1 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. May 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 may override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPFILE	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 may attain.

Property	Datatype	Access	Description
MINWIDTH	Integer	RW	Sets the minimum width, in pixels that the component may 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

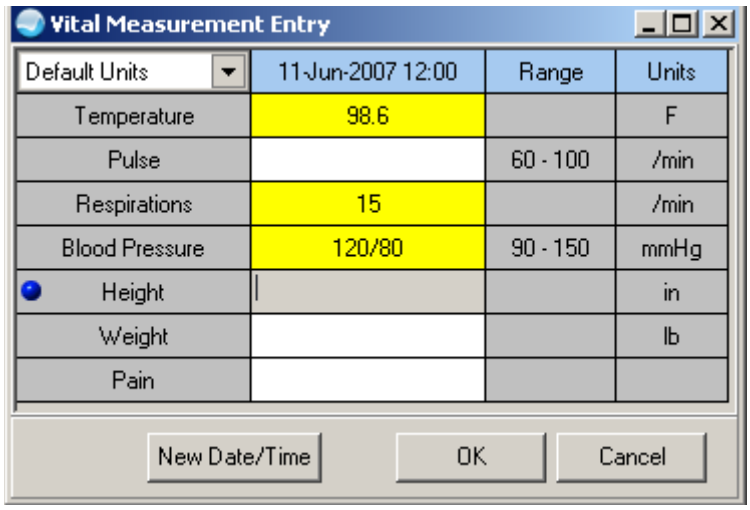


Figure 13-1: Vital Measurement Entry

Vital measurement entry is implemented as a service that may 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 may be dropped directly into the user interface in design mode. Both are contained with the same executable image.

13.2 Architecture and Business Process Overview

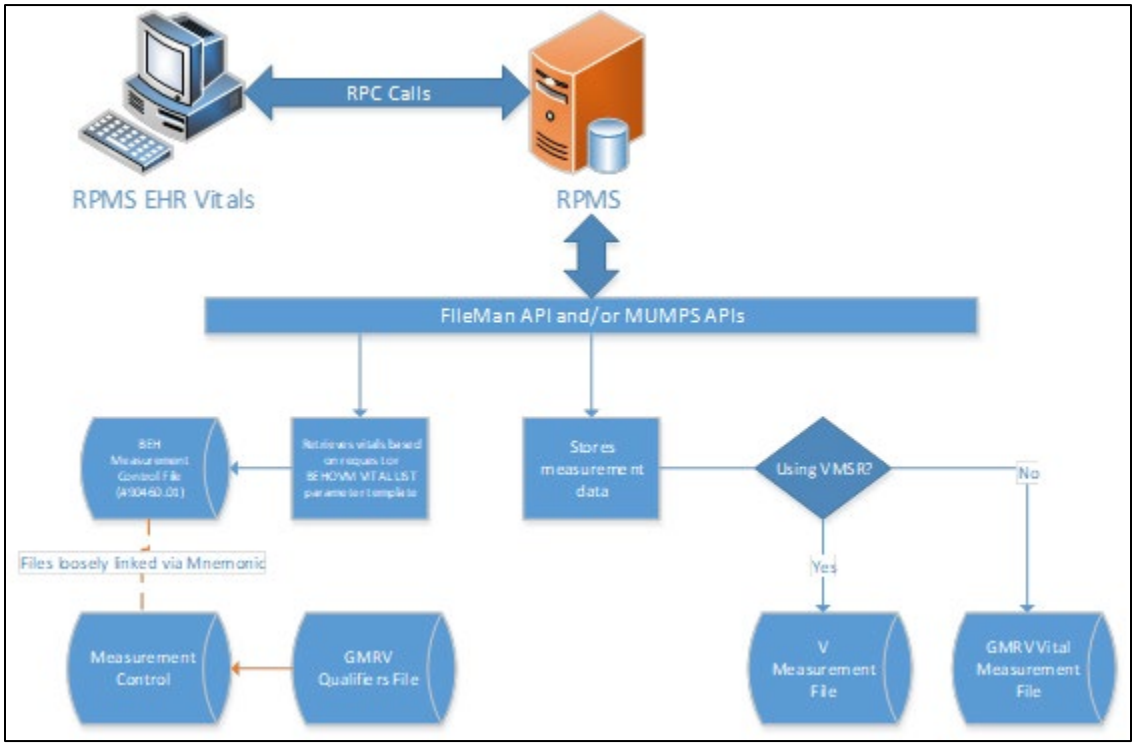


Figure 13-2: Architecture and business process overview

13.3 Implementation and Maintenance

This object has the following configurations:

Vital Measurement Entry (service)

Entity	Value
Programmatic Identifier	BEHVITALENTRY.VITALENTRY
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*001012

Vital Measurement Entry (visual component)

Entity	Value
Programmatic Identifier	BEHVITALENTRY.VITALENTRY2
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*001012

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

13.4 Routine Descriptions

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

Routine	Description
BEHOVM BEHOVM2 BEHOVM4 BEHOVM5	Vital measurement support

Routine	Description
BEHOVMC BEHOVMC2 BEHOVMER BEHOVMRP	Vital measurement report
BEHOVMIN	Installation support

13.5 File List

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

13.5.1 BEH MEASUREMENT CONTROL (#90460.01)

This file controls which measurement types may 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.

Field Name	#	Datatype	Indexes	Description
RETRIEVAL LOGIC	10	M Code		Special retrieval logic. This would include measurement types that are computed values (for example, BMI).
DESCRIPTION	99	Word Processing		Text to be displayed when help is requested.

13.6 Cross References

Cross-references are described in the preceding section.

13.7 Exported Options

Option	Type	Description
BEHOVM DATA ENTRY	action	Set data entry permissions.
BEHOVM DEFAULT UNITS	action	Override default units.
BEHOVM ERROR RPT	action	User access to Vitals Error Report
BEHOVM MAIN	menu	Vital measurement configuration menu.
BEHOVM NEW DATE DEFAULT	Action	Select new date default.
BEHOVM TEMPLATE	action	Create/edit data entry templates.

13.8 Exported Security Keys

None.

13.9 Exported Protocols

None.

13.10 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.

Parameter	Instance Type	Value Type	Precedence	Description
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 ERROR RPT		Boolean	User, Class	If yes, the user can view the entered in error reports.
BEHOVM NEW DATE DEFAULT		Set (0:Current, 2: Now)	User, Class, Service, Location, Division, System	Select new date default radio button
BEHOVM MAX RETURN	Type (T:Template, G:Grid)	Numeric	User, Division, System	Maximum number of a type to return
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

13.11 Exported Mail Groups

None.

13.12 Callable Routines

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

13.12.1 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.

Parameter	Datatype	Description
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	Text containing detailed information about each requested measurement type.

Returns measurement data for detail view.

13.12.2 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.

13.12.3 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.

13.12.4 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.

13.12.5 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.

Parameter	Datatype	Description
<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.

13.12.6 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.

13.12.7 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.

13.12.8 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.

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	Returns data for vital entry template as specified by the BEHOVM TEMPLATE parameter.

Retrieves vital measurement data for populating the data entry grid.

13.12.9 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.

13.12.10 RPC: BEHOVM2 EIE

Scope: private

Parameter	Datatype	Description
BEHDATA	String	IEN(File #9000010.01)^DUZ^REASON
<return value>	String	0 if stored Error code if problem storing value

Marks a vital measurement as entered in error.

13.12.11 RPC: BEHOVM2 GETCATP

Scope: private

Parameter	Datatype	Description
VIEN	IEN (#9000010.01)	Internal entry to the Vital Measurement file.

Parameter	Datatype	Description
<return value>	String List	Returns a list of categories for the associated measurement type and the default category.

Returns list of categories and the default category.

13.12.12 RPC: BEHOVM2 GETCATS

Scope: private

Parameter	Datatype	Description
IEN	Pointer (#90460.01)	BEH MEASUREMENT CONTROL file entry.
<return value>	String	Returns a list of categories and qualifiers.

Returns list of categories and qualifiers.

13.12.13 RPC: BEHOVM2 QUAL

Scope: private

Parameter	Datatype	Description
VIEN	Pointer (#9000010.01)	Vital Measurement file entry.
QUALS	String	IEN (#120.52)~<repeats>
<return value>	String	Returns OK or an error message

Stores qualifiers to the Vital Measurement file entry.

13.12.14 RPC: BEHOVM2 VUNITS

Scope: private

Parameter	Datatype	Description
VCTL	Pointer (#90460.01)	BEH MEASUREMENT CONTROL file entry.
<return value>	String	Returns string containing: English Units^lo^hi^metric^lo^hi

Returns units and high/low values for the measurement type.

13.13 External Relations

Entity	Name	Description
File	MEASUREMENT TYPE (#9999999.07)	Read access.
File	V MEASUREMENT (#9000010.01)	Read and write access.

13.14 Internal Relations

None.

13.15 Archiving and Purging

There are no archiving or purging requirements within this software.

13.16 Components

None.

13.16.1 Service

The Vital Measurement Entry service supports the following properties and methods:

13.16.1.1 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.

13.16.1.2 Execute

Invokes the vital measurement data entry dialog.

13.16.2 Visual Component

The Vital Measurement Entry visual component supports the following properties and methods:

13.16.2.1 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. May 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.
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.

Property	Datatype	Access	Description
CAPTIONSTYLE	Enum	RW	Sets the caption style. May 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 may override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPFILE	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 may attain.
MINWIDTH	Integer	RW	Sets the minimum width, in pixels, that the component may 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.
WIDTH	Integer	RW	Sets the width (in pixels) of the top boundary of the component.

14.0 Vital Measurement Display

14.1 Introduction

Vital Measurements

Vital	Value	Date
TMP	98.6 F (37 C)	15-Jun-2007 15:08
HT	63 in (160.02 cm)	15-Jun-2007 15:08
WT	210 lb (95.45 kg)	15-Jun-2007 15:08
BMI	37.2	15-Jun-2007 15:08
PA	4	27-Apr-2007 14:39
BP	120/66 mmHg	07-Jan-2006 18:47
HC	15.75 in (40 cm)	07-Jan-2006 18:47
HE	A	07-Jan-2006 18:47
PU	61 /min	21-Jan-2004 09:15
RS	16 /min	21-Jan-2004 09:15

Figure 14-1: Vital Measurement Display component

The Vital Measurement Display component provides a quick overview of the most recent vital measurements for display on the cover sheet.

14.2 Architecture and Business Process Overview

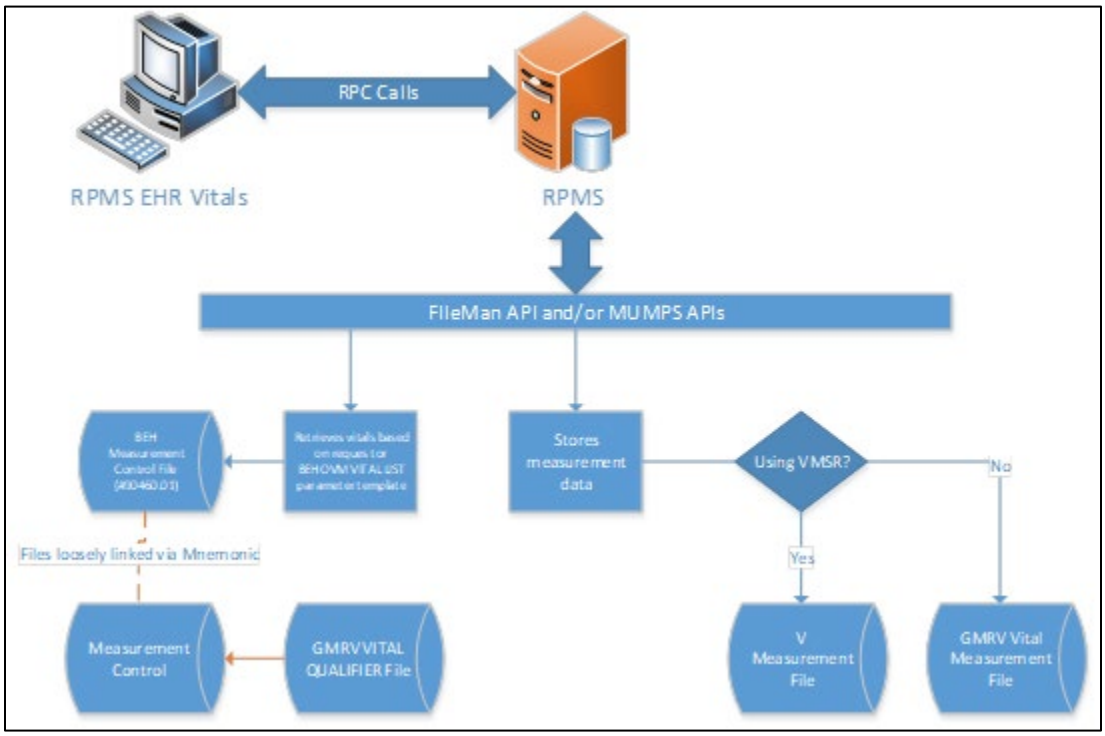


Figure 14-2: Architecture and business process overview

14.3 Implementation and Maintenance

This component has the following configuration:

Entity	Value
Programmatic Identifier	BEHVITALS.VITALDISPLAY
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*001012

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

14.4 Routine Descriptions

None.

14.5 File List

None.

14.6 Cross References

None.

14.7 Exported Options

Option	Type	Description
BEHOVM VITAL LIST	action	Specify measurements listed on cover sheet.

14.8 Exported Security Keys

None.

14.9 Exported Protocols

None.

14.10 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.

14.11 Exported Mail Groups

None.

14.12 Callable Routines

None.

14.13 External Relations

Entity	Name	Description
File	MEASUREMENT TYPE (#9999999.07)	Read access.
File	V MEASUREMENT (#9000010.01)	Read access.

14.14 Internal Relations

Entity	Name	Description
Component	Vital Measurement Entry	Uses supported APIs.

14.15 Archiving and Purging

There are no archiving or purging requirements within this software.

14.16 Components

This component supports the following properties and methods:

14.16.1 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
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. May 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. May 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 may override this setting.
HEIGHT	Integer	RW	Sets the height (in pixels) of the component.
HELPFILE	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. May 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.

Glossary

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.

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.

Daemon

A background process that performs a specified service.

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 may 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 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.

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.

Acronym List

Acronym	Meaning
CCOW	Clinical Context Object Workgroup
CMS	Component Management Service
CSL	Communication Service Layer
CSS	Component Support Services
IHS	Indian Health Service
RPC	Remote Procedure Call
RPMS	Resource and Patient Management System
VIM	Visual Interface Manager

Contact Information

If you have any questions or comments regarding this distribution, please contact the IHS IT Service Desk.

Phone: (888) 830-7280 (toll free)

Web: https://www.ihs.gov/itsupport/

Email: itsupport@ihs.gov