The Architecture of Vaccess
Version 3.0

M.D.Geib

Vista Control Systems, Inc.

I. INTRODUCTION

This paper discusses some of the features of the Vaccess V3 architecture and the related functionality provided by the architecture. The initial release of V3 is scheduled for early next year and will include at that time about four man years of effort.

Philosophy

The basic philosophy of the Vaccess V3 project was to produce a product that is backward compatible, is easily portable to a variety of computing platforms, supports long term development, and is of the highest quality possible.

Beyond that, the underlying philosophy of Vaccess, and Vsystem as a whole, is to provide a complete set of tools which can easily be used to implement control and monitoring systems. The customer should be able to configure the system as he chooses and not be restricted by the architecture of the toolset. To that end Vsystem supports all the components on all the supported platforms, with the one exception that Vdraw is not supported on VxWorks. Specifically, Vaccess databases can be hosted on all supported platforms along with the complete application programming interface (API). Planned support for the initial release of V3 includes OpenVMS, VAXELN, VxWorks, Digital UNIX, SUN Solaris SPARC, and HARRIS Power UNIX. The initial release will include equivalent functionality and full interoperability between all supported platforms. Internally, Vista also has V3 running on SUN Solaris x86 and UNIXware.

Objectives

The following are the major objectives established for the Vaccess V3 project.

Backward compatibility

> Vaccess V3 must be backward compatible with previous versions of Vaccess.

Inter-version support

> Beginning with V3, Vaccess will support networked systems that include platforms running different versions of Vsystem. This mixed version support allows for large multi-node systems to be upgraded a single node at a time, without the requirement of taking the entire system down to upgrade all systems running Vsystem.

Portability

> One of the main reasons for embarking on the V3 project was to support additional computing platforms. The previous version, V2, supported OpenVMS and VAXELN. The new targets

identified for future support included UNIX, VxWorks, and Windows NT. With such a wide variety of platforms to support, one of the most important objectives of the project was to produce a product which is easily ported to new platforms.

Maintainability

V2 was difficult to maintain. The system had a very high level of coupling and low cohesion, both characteristics of systems that are difficult to modify and maintain. From the very beginning V3 was designed and implemented with maintainability in mind. The V3 re-engineering project made extensive use of formal analysis and design methods.

Support for future enhancements

Because of the high level of coupling, for one thing, it was becoming almost impossible to add or enhance functionality in V2. Support for many new features was included in the design of V3, but in addition, the design and implementation of V3 allows for new functionality to be easily added in the future.

II. OVERVIEW

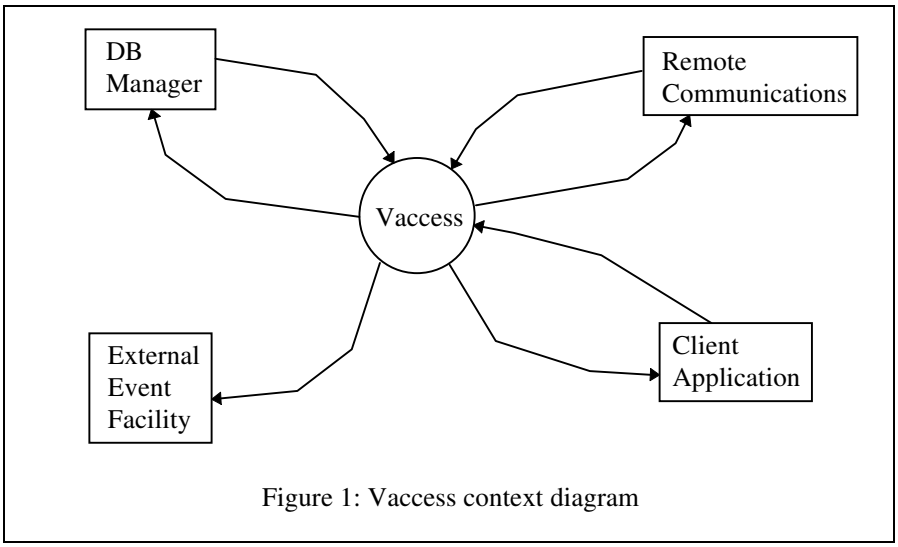

Figure 1: Vaccess context diagram

Figure 1 shows a simplified data flow diagram for the context of Vaccess. This diagram shows the relationship between Vaccess and the other facilities it interacts with. As can be seen in the diagram the remote communications facility is not part of Vaccess but is implemented as a separate facility, to make it easy to replace or modify. Likewise, the external event facility, which handles the delivery of Vaccess events is implemented as a separate facility to minimize any dependency of Vaccess on how this facility is implemented. The box labeled *DB Manager* is a new facility in Vaccess V3. The *DB manager* provides such services as mapping a database into memory, managing the database definitions and locations, and controlling access to databases. Since many of the *DB manager* services depend on the OS services provided by the platform it was decided to implement them in a separate facility that can be tied more closely to the platform.

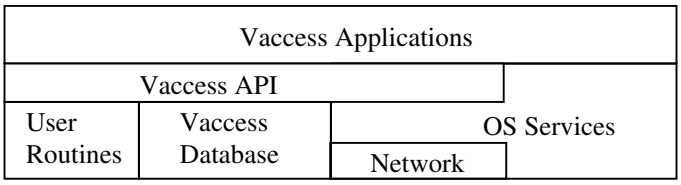

Figure 2: Vaccess application layers

Vaccess structure

Vaccess is made up of a real-time database and an API for accessing databases. Figure 2 shows the relationship between a Vaccess application, the Vaccess API, the available OS services, and the Vaccess database. The layer labeled *User Routines* are the handlers and conversion routines invoked directly by Vaccess, not the Vaccess event callbacks. The API transparently supports the access of databases hosted on remote platforms. In addition, the API provides for synchronized access to the data structures that make up the real-time database when access is local (that is, on the platform which is hosting the database). An active database consists of a number of data structures that are mapped in memory.

Supported configurations

There are no restrictions on the number and type of platforms which participate in a Vaccess based system. Any platform supported by Vaccess is equivalent in every way with all other supported platforms.

The number of databases supported on a single node is limited only by the amount of memory available. Any single process can connect to 65536 databases, local or remote in any combination. Each Vaccess V3 database supports 65536 channels. These are not hard limits, and can be increased in a future version of Vaccess.

III. DATABASE STRUCTURE AND FUNCTIONALITY

A Vaccess V3 database is composed of a database header, sorted tables, database channels, a table for storing information about each process and thread connected to a database, and two dynamic memory pools. The memory pools are used for event support, dynamic channel creation, and for storing channel fields with dynamic sizes.

Channel Types

The type of a Vaccess channel is related to the data type of the values stored in the channel. A channel is made up of many different fields that can have many different data types.

Vaccess supports channels of the following types:

- integer, single and double precision real values
- arrays of integer, and single and double precision real values
- character or single byte arrays
- time and time arrays
- binary and binary arrays

All channel types include both internal and external representation. All channel types also include support for timestamping.

Channel Functionality

All channels support the same basic functionality. Some additional functionality is provided in specific types; this functionality will be noted when present.

- A Vaccess event is generated when any field within a channel is modified.
- A Vaccess event is generated when any channel changes state (for example alarm state or clipping state).
- A standard set of fields for naming and describing the channel.
- A set of fields that specify any clipping limits for the channels value.
- A similar set of fields is provided to specify any limits for displaying the channels value.
- A delta value that specifies the required change in a channels value before it is considered significant.
- Alarm support for a user specified number of alarm levels. Included in the alarm block is a string field for specifying an alarm label or alarm message for a channel. A channel's alarm levels can also be specified to be relative to another channel's value, as either a percentage or a fixed offset. Integer channels can be configured to enter an alarm when the channel's value matches a specified match alarm value.
- A hardware block can be configured to include any number of hardware related parameters. It also includes a number of text and integer data fields that can be used to identify any hardware associated with the channel. The hardware block also contains the name of the hardware handler routine which is invoked by Vaccess during a read or write to the channel. The handlers function is to interface with hardware to read or write the database value.
- A conversion block can be configured to include any number of conversion related parameters. A conversion block is included in the channel when a user specifies a user supplied conversion routine for converting a channel's external value to the internal, and vice versa. The conversion also contains the name of the user-supplied conversion routine invoked by Vaccess to perform conversions. Both external-to-internal and internal-to-external conversion routines can be specified. As an alternative, a channel can also be configured to include a linear conversion by specifying a slope and intercept.

Support for threads

V3 of Vaccess supports the use of threads. Earlier versions prevented the use of threads.

Threads can be used to achieve high levels of concurrency in a program. For access to remotely hosted databases, threads can be used to improve response to remotely executed routines.

Polled or Event Driven

Vaccess supports the development of polled or event driven systems, or a mixture of the two methods.

Platform independent data

All files produced and used by Vsystem are platform independent. Database files can be generated on one platform and used on all supported platforms. Likewise, Vdraw screen files can be produced on one platform and used on all platforms.

Flexible and efficient

Vaccess V3 supports a very flexible database configuration. When defining a database the user has control over what structures and features are included in each channel. For channels that require a minimum of functionality, the memory resident representation of the channel is very small, consuming a minimum of memory resources. Only channels which are explicitly defined to include optional features and structures consume the resource to provide those requirements. In addition, there are no limits on the number of fields included in a channel's definition. For example, a channel can be defined to provide 1 or 1000 hardware parameters; the only practical limit is available memory.

Vaccess V3 also supports the creation of database channels at runtime, while a database is active. A user program can create a new channel without shutting down the entire system and re-activating a new database. Other processes can ask to be notified with a Vaccess event when a new channel is created.

A user can now create sorted tables in addition to the standard tables provided by Vaccess. Vaccess provides a sorted table that holds the channels in alphabetical order and one that sorts the channels based on hardware block values. The user can create additional tables that are sorted using routines supplied by the user. The user sorting routine can access any channel data to use in the sorting.

IV. NETWORK SUPPORT

Vaccess provides the network support that allows for systems to include multiple nodes. Systems based on the Vsystem tools can be a single computer or many computers. The network support is totally invisible to all the applications that make up a system. The low level network support has been implemented in modular way to make it easy to support different network transports in the future.

Transparent database location

Vaccess databases can be hosted on any platform supported by Vsystem. The actual location of a database is not fixed and easily changed. All of the Vsystem tools, as well as all user developed programs, function the same whether the database is located locally or remotely. Complete systems can be developed without concern for which platform will ultimately host the real-time database(s). During development, each developer can have a private database to eliminate problems that might arise if multiple developers were using and modifying a single database. Once the system is ready, all the programs run, unchanged, by simply changing the database definition(s) to point to the correct database.

Support for threads

As mentioned above, Vaccess V3 supports the use of threads. For programs that use threads to increase concurrency the same advantage is gained for remote access to a database. Each thread within a program establishes an independent communications channel to the platform hosting the database. Independent communication channels for each thread within a process allows for multiple remote functions to execute in parallel, improving the overall response to remote access.

Heterogeneous platform support

There are no restrictions on the type of platforms that participate in a networked system based on Vsystem. Any of the supported platforms can communicate and interoperate with any other supported platform, in any combination or mixture.

V. EVENTS

Vaccess events can be generated on almost any database access or change of state. As with many of the Vsystem tools, a user program can request to be notified when any event occurs. The request for notification specifies the name of a user supplied routine which is the event's callback, an optional user specified argument which is passed to the callback routine when invoked, and a specification for the event. Event specifications can be very specific or general in nature so that a single callback routine can handle many different events. When the callback is invoked it is passed a structure that includes the time of the event, the specific event that occurred, the user's argument, and the requested channel and database data.

User specified event data

When an event request is made, the user can specify what channel data to include in the data passed to the callback routine. For example, if a channel goes into an alarm state, the user may need to know the current value of the channel, the current alarm levels, and maybe the internal value of the channel. In previous versions of Vaccess the callback routine would have to read all this information from the channel, except the channel value, with additional API calls. In V3 any channel data can be passed to the callback routine. Eliminating extra API calls improves the performance of the callback routine, especially when the database being accessed is hosted remotely.

Time modified events

Requests for event notification can now include a number of time related options. The request can specify that the delivery of an event be delayed by a specified delta time or until an absolute time. For change of state events, like alarms, if the channel's state again changes before the delay time expires, the original event is canceled and never delivered. This feature might be used to prevent numerous alarm events from being delivered on a channel that is going through a transition which results in the value moving in and out of alarm rapidly.

An event notification request can also specify the minimum time that must pass between the delivery of an event. This action effectively throttles the delivery of an event. For example, if a channel's value is being changed every 100 milliseconds there is no need to deliver events to a Vdraw screen at that rate. Vdraw can specify some time interval at which the events can be delivered to it for display update. This feature is very effective in reducing the load on a system that would normally generate events at a high rate.

VI. REQUIRED OS FACILITIES

Vaccess V3 was designed to port to a wide variety of platforms. To help reduce the work of porting, a minimum set of OS services have been used. This also makes it easy to determine if a new platform will support Vaccess. Following is the list of required services or facilities that a platform must provide to support Vaccess.

threads

V3 of Vaccess supports and makes use of threads. Threads have been used to minimize the use of platform-specific services and facilities. For example, the event callback routines are now executed as a separate thread in a user's program. Earlier versions of Vaccess used OpenVMS AST and VAXELN event services to asynchronously deliver events.

global shared memory

The Vaccess database is a memory resident set of data structures. All local processes that access the database map the database through the API calls. In order for more than one process to map the database, the platform must provide global shared memory services.

broadcast signaling

In order to minimize the process specific information stored in a Vaccess database, it was decided to use a broadcast type signal to inform a process's event processing thread when a new event had been delivered to it. When this signal is generated, all event processing threads in all processes connected to a database are notified to check for a new event.

VII. SUMMARY

The architecture of Vaccess V3 is the result of achieving the objectives established for the project. Those objectives include inter-version support, portability, maintainability, and support for future enhancements. In addition, some of the design decisions were the result of the requirement that V3 be backward compatible with previous versions.

The portability of V3 has been proven by the variety of platforms on which it is already running. Likewise, it has already proven to be simple to maintain. The backward compatibility has been verified by porting many applications from the previous version of Vaccess to V3 with almost no changes to the source required.