COMPAQ

# Software Product Description

**PRODUCT: Compaq Visual TeMIP V4.0 for Tru64 Unix** **SPD 60.64.06**

## DESCRIPTION

Visual TeMIP is a C++ software development environment for TeMIP applications (management modules), providing third parties and users with a TeMIP environment open to new functionality integration and significantly reducing management module development time.

The Developer's Toolkit consists of software libraries, various sample modules and tools to facilitate the creation of management modules. In particular, it contains the following:

- Visual TeMIP classes and their corresponding libraries
- Visual TeMIP Dev AM, Example AM, Example FM, Dictionary Driven Dev AM and Dictionary Driven Example AM management modules

Visual TeMIP makes extensive use of the object-oriented features of C++ (inheritance and virtual functions), but a basic C++ understanding as well as a minimum knowledge of TeMIP Framework is enough to fully benefit from this toolkit.

Visual TeMIP now supports the RogueWave Tools.h++ class library. Utility classes are provided in Visual TeMIP to support RogueWave's queue and list features. Visual TeMIP classes have been modified to better integrate with the RogueWave classes, and have been made collectable as defined by RogueWave.

Using Visual TeMIP, third parties and users can develop applications for the TeMIP Network Management family of products to provide access to, and management functions for, any manageable object. Software and documentation for the development of integrated TeMIP modules are available as part of the Visual TeMIP Developer's Toolkit.

## VISUAL TEMIP CLASSES

Visual TeMIP provides three levels of **class** to develop a management module. Each level is designed so that it can be used without the layers above it. These are the Support Classes, Wrapper Classes (including the Extension Classes), and Framework Classes, as described below:

### Support Classes

The Support classes provide a set of basic, general purpose C++ classes that support common functions, such as:

- Queues and lists: the Visual TeMIP classes are based on the RogueWave queuing functions with some additional features added, such as thresholds for signaling a normal or congested state, and a monitor to ensure threadsafe operation
- Thread management: for the synchronization of threads; mutex, lock and condition variable handling
- Tracing
- Exception handling
- Tidy pointers: provide leak-free memory management
- Sets and sequences: generic set/sequence management for any base datatype is provided by functions such as Add, Remove, Contains, Intersection, and Union. Bags provide sets with counted membership.

**Note:** These classes have now been replaced by RogueWave functions, but are maintained for backward compatibility.

The Support classes are designed to interact with the Wrapper and Framework classes, and are used extensively by the Framework.

**February 2000**
**AE-QX5KG-TE**

### Wrapper Classes

The Wrapper classes and their member functions provide a wrapper around the TeMIP function calls and their associated datatypes, greatly reducing the amount of repetitive code needed to use them.

Many of the Wrapper Classes include conversion operators that provide access to the TeMIP Framework datatypes that they include. Thus they can be passed as parameters to the classic TeMIP Framework functions as well as to Wrapper and Framework class functions.

Wrapper classes are intended to ease the task of implementation, but not change or simplify the semantics of the TeMIP Framework interfaces.

The Wrapper classes provide a set of basic C++ classes (or families of classes) for the following:

- TeMIP Framework datatypes. Most Framework datatypes have their corresponding wrapper classes (Refer to the Visual TeMIP Reference Guide for a full list). Also provided are classes for facilitating the support of constructed and user-defined types. All datatype classes are collectable, as defined by RogueWave (in the Tools.h++ User's Guide) to facilitate integration with RogueWave classes.
- Generic datatypes, which can be encoded and decoded using the TeMIP dictionary.
- TeMIP descriptors
- Entity specifications
- Entity classes
- Handles
- Strings
- ILV encoding/decoding
- Time specifications for handling Abstract Time Specifications (ATS)
- Management Information Repository (MIR)
- Exception handling
- Event handling, allowing the creation of an event, adding arguments to it and the use of the Event Manager to dispatch it. Specific support is provided for OSI events
- Making Call Requests. (Note: receiving call requests is done by the Framework Classes.) Using Visual TeMIP's generic support for request/response objects, it is possible to encode/decode calls and to encode events by referring to the TeMIP dictionary.

### Wrapper Classes (Extension)

The Wrapper Extension Classes (layered on top of the Wrapper Classes) provide a set of extended functionality around TeMIP Framework. They offer more value-added functions around Presentation Module development and Event Filtering management:

- TeMIP Security (ACLOC) C++ API.

This is delivered as a separate library and allows the filtering and logging of operator requests and verification of whether a request is authorized or not. It supports multiple sessions: for modules that act as multi-user servers, distinct security profiles can be used.

- Asynchronous Call Support.

This is delivered as a separate library and include file and allows Call Requests to be made without needing to wait for the response. This is particularly adapted to Presentation Module development, where user interface events are asynchronously delivered.

- Low Level Event Filtering C++ API.

Allows the filtering of events put in the Event Manager, benefiting from the Event Filtering and Correlation features (Wrapper Classes).

### Framework Classes

The Framework classes provide high-level classes for dealing with management module classes, attributes, directives and events. The Framework classes enable management modules to be written, using C++, without a detailed understanding of the underlying TeMIP Kernel functions nor of the conventions for passing information between modules.

The TeMIP Framework takes care of all the normal interactions with the TeMIP Kernel. A set of C++ macros is used to describe the structure of the entities being modeled, and their interaction with the developer's code. Using C++ virtual functions, all essential or default behavior is provided by this framework.

Aspects that are specific to the management module are coded by deriving the Framework classes (using macros) and by implementing the code as member functions of the derived classes.

The main classes that are visible to a user of the Framework are:

- MModule class: represents the properties of the module as a whole.
- MClassGroup class: represents the properties of a class group. It also contains the properties and behavior shared by the classes belonging to the class group.
- MClass class: represents the properties of a single entity class.
- MInstance class: represents the properties of a single instance of an entity class.
- MAttributeGroup class: represents the properties of an attribute group. It also contains the properties and behavior shared by the attributes belonging to the attribute group.
- MAttribute class: represents the properties of the attributes of an entity class.

- MAttribInstance class: represents the value of the attribute instance for a given instance within an entity class.
- MVerbGroup class: represents the properties of a verb group. It also contains the properties and behavior shared by the verbs belonging to the verb group.
- MVerb class: represents the properties of a particular verb of a particular class.
- MDirectiveContext class: represents the information (dynamic context) needed to process a directive.
- MEventClass class: represents the properties of the event. It is the counterpart to MVerb, for handling received events from another management module.
- MEventContext class: is the counterpart to MDirectiveContext, for dealing with received events.
- MFmkAttribList class: represents the attribute lists of the Show/Set directive.

The Visual TeMIP Reference Guide describes each class in detail.

**Dictionary Driven Features**

Dictionary driven features allow an extension of the module definition and behavior with a minimum code writing/compilation impact.

The dictionary driven mechanisms have to be turned on explicitly by the module: by default no dictionary driven feature will be used.

The core of these features is the notion of groups: each group handles the properties and behavior of a set of objects (class group for classes, verb group for verbs and attribute group for attributes). Then, an external textual configuration file provides the mapping between the behavior contained in groups (that is supplied in the code of the module), and the actual managed model.

As a consequence, it is not mandatory, when using dictionary driven features, to completely define the structure of information on which the module operates. Once the code is written/compiled for groups, any new class/verb/attribute of interest to the module that matches a group behavior can be added to the module just by modifying slightly the textual configuration file: no code writing needed, no recompilation needed, just reenroll and restart the module.

In order to optimize the dictionary driven added value, It is critical to find out the groups that best match the module functionality. The dictionary driven features also provide mechanisms to ease the directive handling:

- Manipulation of input and output arguments in a generic way on the Call Provider side.
- Declaration of the lists of input and output arguments not required: the lists can be expanded dynamically.

**Dictionary Wrapper**

The Dictionary Wrapper adds a family of C++ classes for encapsulating dictionary access. These classes provide:

- Information methods to access data specific to a given dictionary object (MOID class)
- Navigation methods to access subordinate definitions from a parent object and representation of the possible datatypes for attributes and arguments (MDictObject classes)

**ERROR HANDLING**

Visual TeMIP takes advantage of the exception mechanism in the C++ language to eliminate code-written checks of status values that can be returned from any of the TeMIP Framework functions.

All Visual TeMIP functions make use of the exception mechanism when they detect unrecoverable errors. The framework contains exception handlers at appropriate points which convert the error into some appropriate return to the caller.

All common exceptions can be generated with arguments, using the dictionary to encode the exception arguments.

**DOCUMENTATION**

The Visual TeMIP Developer's Toolkit documentation provides information to assist in the design and development of TeMIP management modules. It includes the following documents:

- The Visual TeMIP Reference Guide
- The Visual TeMIP Development Guide

**Note:** Since C++ is an extension of C, the user can at any time go to the level of the TeMIP Framework System Reference Manual and interact directly with the TeMIP Kernel. Refer also to the TeMIP Framework Software Product Description (SPD 54.17.xx) for more information.

**DEVELOPMENT PROCESS AND EXAMPLES**

Management modules that use Visual TeMIP and an object-oriented approach contain several code modules. Each code module contains the necessary code to implement or to override a method needed by the Visual TeMIP architecture. Therefore, the implementation consists of the header file and one or more conventional C++ source files (modules):

- Header Module: contains a description of the classes to be handled by the module (for example, a translation of the MSL specifications of the management module). The header module can also contain a description of the groups that will be used to dynamically manage the classes, verbs and attributes of interest to the module. In this case, an external textual configuration file, describing the mapping between groups and the actual model managed, has to be written.
- Main Code Module: contains the declarations necessary to Visual TeMIP to automatically generate the code for the standard behavior of the module, classes, attributes, and directives managed by the module (whether they are defined - in the Header Module - explicitly or indirectly through groups).
- Override Code Module: contains the code of the overridden Visual TeMIP methods necessary to modify the default behavior of standard directives to meet the user's specific needs or to implement nonstandard directives.
- Specific Code Module: contains routines that perform common internal functions, mainly for encoding and decoding complex parameters.

## VISUAL TEMIP EXAMPLES

The Visual TeMIP Developer's Toolkit provides various sample modules:

- The Visual TeMIP Development AM: implements a simple example of a management module, using the default Visual TeMIP behavior. It has only a Header and Main code module and no Override code module.
- The Visual TeMIP Example AM: implements a more complex model with specific directive processing, event generation and manipulation of complex datatypes and event parameters.
- The Visual TeMIP Example FM: a value-added module that enhances the Visual TeMIP Example AM with counter and statistic computations as well as some event-monitoring functions.
- The Visual TeMIP Dictionary Driven Development AM: implements a simple example of a dictionary driven management module.
- The Visual TeMIP Dictionary Driven Example AM: implements a dictionary driven module handling a more complex model. Most of the available dictionary driven features are used in this example.
- The Visual TeMIP Asynchronous Example: implements a TeMIP application, which performs asynchronous Call Requests that display all active management modules using a graphical representation.
- The Visual TeMIP Dictionary Wrapper Example: "dumps" the dictionary information for a given entity class.

The Visual TeMIP Development Guide explains how to build and enroll management modules.

## HARDWARE REQUIREMENTS

*Tier 1:*

Compaq AlphaServer DS10, DS20
Compaq Professional Workstation XP1000

*Tier 2:*

Compaq AlphaServer ES40

*Tier 3:*

Compaq AlphaServer GS60, GS140

### Disk Space Requirements

*Disk space required for installation:*

| | |
|---|---|
| Root file system | 0 Kbytes |
| Other file systems, /usr | 10,000 Kbytes |

Disk space *required* for use (permanent):

| | |
|---|---|
| Root file system | 0 Kbytes |
| Other file systems, /usr | 15,000 Kbytes |

These counts refer to the disk space required on the system disk. The sizes are approximate; actual sizes may vary depending on the user's system environment, configuration, and software options.

### Memory Requirements

The minimum memory supported is 128 Mbytes.

*Recommended* Configuration (for developing management modules using Visual TeMIP):

Compaq Professional Workstation XP1000
128 MB memory
RZ26 disk or equivalent disk space
Ethernet controller

**Note:** For testing purposes, one should consider increasing memory size at least to the closest operational system size. Specific network environments may require larger configurations.

## SOFTWARE REQUIREMENTS

For Systems Using Terminals and Workstations:

- COMPAQ Tru64 UNIX V4.0F
- TeMIP Framework V4.0
- DIGITAL C++ V6.1

Please contact your Compaq representative for availability dates on Tru64 UNIX V4.0-F

## OPTIONAL SOFTWARE

- Ladebug Debugger V4.0-44

## GROWTH CONSIDERATIONS

The minimum hardware/software requirements for any future version of this product may be different from the current version requirements.

## YEAR 2000 READY

This product is Year 2000 Ready.

"Year 2000 Ready" products are defined by Compaq as products capable of accurately processing, providing, and/or receiving date data from, into and between the twentieth and the twenty-first centuries, and the years 1999 and 2000, including leap year calculations, when used in accordance with the associated Compaq product documentation and provided that all hardware, firmware and software used in combination with such Compaq products properly exchange accurate date data with the Compaq products.

For additional information visit Compaq's Year 2000 Product Readiness web site located at http://www.compaq.com/year2000

To ensure that this product is Year 2000 Ready, code assessment and system tests to verify the transition between December 31st 1999 and January 1st 2000 were utilized.

To ensure that this product interoperates properly with other hardware and software, the system tests involving Compaq's TeMIP V3.2 are applicable, as this product was verified as being Year 2000 Ready.

## DISTRIBUTION MEDIA

This product is only available as part of the UNIX Consolidated Software distribution on CD-ROM. Please refer to the ordering information for each Software Media reference.

## ORDERING INFORMATION

Software License : QM-6HSAA-AA
which replaces the license QL-58RA9-AA or QM-58RAA-AA of TeMIP 3.2

Software Media : QA-6HPAA-H8
Software Documentation : QA-6HSAA-GZ
Software Product Services : QT-6HS**-**

The QA-*****-H8 part numbers no longer include the QA-*****-GZ documentation kits.Hard copy documentation for this product should be ordered using the QA-6HSAA--GZ number, if required.

**Note:** * denotes variant fields. For additional information on available licenses, services and media, refer to the appropriate price book.

## SOFTWARE LICENSING

This software is furnished under the licensing provisions of Compaq Computer Corporation's Shrinkwrap License Terms and Conditions. For more information about the Compaq licensing terms and policies, contact your local Compaq office.

Licence units for *Visual TeMIP Developer's Toolkit* are allocated on an Unlimited System Use basis, in line with the machine tier on which they run.

## COMPAQ TRU64 UNIX LICENSE MANAGEMENT

This product uses the FLEXlm Software License Key system.

A FLEXlm key must be obtained using information provided with the license deliverable. An authorization ID is provided for each license, which allows the user to generate license keys from the Compaq License Key Fulfillment Web Site according to instructions provided with the license agreement.

## SOFTWARE PRODUCT SERVICES

A variety of service options are available from Compaq. For more information on these services or other available Network Management Services, contact your local Compaq office.

**SOFTWARE WARRANTY**

This software is provided by Compaq with a 90 day conformance warranty in accordance with the Compaq warranty terms and applicable to the license purchase.

The above information is valid at time of release. Please contact your local Compaq office for the most up-to-date information.