GEORGEJAMES
SOFTWARE

VC/m

VC/m User Guide
Version 3.2

Notices

Copyright 1990-2009 George James Software Limited.

No part of this publication may be reproduced, transmitted, stored in a retrieval system, nor translated into any human or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise, without the prior written permission of the copyright owner, George James Software Limited, 42-44 High Street, Shepperton, Middlesex, TW17 9AU, United Kingdom.

Information contained in this publication is subject to change without notice and does not represent a commitment on the part of George James Software Limited.

Any copyrighted software accompanying this publication is licensed to you only for use in strict accordance with the Software License Agreement accompanying the software. Please read the license agreement carefully before commencing use of the software.

RE/data, RE/m, RE/parser, VC/m, Serenji and Umlanji are trademarks of George James Software Limited. All other brand and product names are or may be trademarks of, and are used to identify products and services of, their respective owners.

Product Support

Support is available to all users of George James Software Products who have a current Software Maintenance Agreement. Support and assistance can be obtained from the following sources:

Telephone +44-1932-252568
E-mail support@georgejames.com
Web page www.georgejames.com

Rev	Document Reference	Date	Prepared by	Details
0	\\Wiz\D\Lib\Manuals\VCm\ VCmUserGuide.doc	30 August 2005	Gail Treves-Brown	
1	VCmUserGuide.doc	29 November 2005	John Murray	VCmManual.doc/1.2
2	VCmUserGuide.doc	20 December 2005	John Murray	VCmManual.doc/1.3
3	VCmUserGuide.doc	25 August 2006	John Murray	VCmManual.doc/1.4
4	VCmUserGuide.doc	16 November 2006	John Murray	VCmManual.doc/1.5
5	VCmUserGuide.doc	16 July 2009	John Murray	VCmManual.doc/1.6
6	VCmUserGuide.doc	14 September 2009	John Murray	VCmManual.doc/1.7

VC/m User Guide version 3.2

Contents

1 CONFIGURATION MANAGEMENT 1

2 INTRODUCTION TO VC/M 3

2.1 VC/M OVERVIEW 3
2.2 VC/M CONCEPTS 5
Components and Component Types 5
Object Versions, Objects, Variants and Versions 6
Change Requests, Change Request Types and Change Request Status 7
Physical Locations, Storage Formats, Locations and Location Classes 8
The Library 9
Systems 10
Existence and Object Status: Master, Active and Error 11
Transfer Routes and Function Codes 12
Transfer Route Dependencies 13
Access Codes 14
Modules 15
2.3 VC/M TRANSFERS 16
Transfers 16
Valid Transfers 18
Standard Transfers 19
Volume Transfers 20
Automatic Transfers 21
Tied Locations 22

3 USING VC/M FOR THE FIRST TIME 23

3.1 STARTING VC/M'S BROWSER-BASED INTERFACE 23
3.2 STARTING VC/M'S CHARACTER-BASED INTERFACE 23
3.3 GETTING AROUND IN VC/M'S CHARACTER-BASED INTERFACE 24

4 INSTALLATION GUIDE 25

4.1 SYSTEM REQUIREMENTS 25
4.2 INSTALLATION PROCEDURE 26
4.3 SERVER INSTALLATION 27
Installing the VC/m Program Files 27
Preparing the M Environment 28
Installing the VC/m Routines 31
Running the Installation Script 33
Additional Steps for Caché 5.1 or Later, or Ensemble 34
Checking the Installation 35
Entering the License Key 36
4.4 CONFIGURING THE SERVER FOR ACCESS VIA THE BROWSER INTERFACE 37
System Requirements 37
IIS Web Server with CSP Interface 38
IIS Web Server with WebLink Interface 39
IIS Web Server with serverLink Interface 40
Apache Web Server with CSP Interface 42
Apache for UNIX Web Server with WebLink Interface 43

Apache for UNIX Web Server with serverLink Interface ... 45
Apache for Windows Web Server ... 47
Checking the Browser-Interface Installation ... 48
WebLink Guide ... 49
ServerLink Guide ... 50
4.5 BACKING UP YOUR INSTALLATION ... 51
4.6 ADDITIONAL VC/M INSTALLATIONS ... 52
VC/live ... 52
Task Server ... 53
4.7 BASIC M GUIDE ... 54
Caché ... 54
DSM ... 55
GT.M ... 56
MSM ... 57
M21 ... 58

5 SYSTEM MANAGER'S GUIDE ... 59

5.1 CONFIGURATION PROCEDURE ... 59
XML Definitions ... 60
5.2 RECOMMENDATIONS FOR BEST PRACTICE ... 61
User IDs and Access Codes ... 61
Locations and Location Classes ... 62
5.3 COMPONENT TYPES ... 63
Standard Component Drivers ... 63
Component Type Table ... 66
Component Date /Time Stamps ... 67
Comment Lines ... 68
Managing Routine Components ... 69
Managing Global Components ... 70
Managing Text and Binary Files ... 71
Managing Caché MAC and INC Components ... 72
Managing M/SQL Components ... 73
Managing Caché Object Class Definitions ... 74
Managing Caché Server Pages and Rules ... 75
Creating Customized Component Drivers ... 76
5.4 PHYSICAL AND LOGICAL LOCATIONS ... 77
Storage Formats ... 77
Physical Addresses ... 78
Dependency Checks ... 81
Relaxing Control: Uncontrolled Locations ... 82
Library Locations ... 83
Sequential File Locations ... 84
Locations without Physical Addresses ... 85
Generic Locations ... 86
Copying Another Logical Location ... 87
The NEW Location ... 88
5.5 TRANSFER ROUTES ... 89
Function Codes ... 89
Dependent Locations ... 90
Installation-specific Dependencies ... 91
New Active Locations ... 92
New Master Location ... 93
Create New Version ... 94
Copy Only if Changed ... 95
Audit Trail Messages ... 96
Moving from a Library Location (De-activating) ... 97

Automatic Transfers: AUTOT Function Code ... *98*
5.6 MODEL TRANSFER ROUTES AND CONFIGURATIONS ... 99
5.7 THE AUDIT TRAIL ... 100
5.8 OBJECTS ... 101
Variant Codes ... *101*
5.9 CHANGE REQUESTS ... 102
Change Request Callouts ... *102*
Change Request Selection Callouts ... *103*
5.10 LOG-IN AND ACCESS CONTROLS ... 104
Access Codes ... *104*
Access Controls ... *105*
User IDs and Terminal-Mode Interface Log-in ... *106*
Log-in with IIS ... *107*
5.11 MENU CUSTOMIZATION ... 108
Customized Menus ... *108*
Customized Menu Options ... *110*
5.12 PERIPHERAL DEVICES ... 111
5.13 INTEGRATION WITH OTHER DEVELOPMENT AND SUPPORT TOOLS ... 112
Controlling Component Editing ... *112*
Interfacing from an Editing Tool ... *113*
Interface from Serenji ... *114*
Interface from Caché Studio (5.x or later) or Ensemble Studio ... *115*
Interface from Caché 4.x Studio ... *117*
Interface from M/SQL ... *118*
Interfaces to the Browser Interface ... *119*
Interface from Microsoft Word ... *120*
Interface from Delphi IDE ... *122*
Creating an Interface from an Editor ... *124*
Integration with Visual SourceSafe ... *127*
5.14 COMPONENT AND OBJECT TAKE-ON ... 128
Taking On Non-R Type Components ... *129*
Specifying the Object Name when Loading ... *130*
5.15 REMOVING AND DELETING ... 131
5.16 MANAGING CONCURRENT DEVELOPMENT ... 132
5.17 HOUSEKEEPING ... 134
5.18 MANAGING CODE ON OTHER MACHINES ... 135
5.19 THE TASK SERVER ... 136
5.20 VC/LIVE ... 137
Using VC/live ... *138*
Locations and Routes ... *139*
Configuring Non-Interactive Installation ... *140*
5.21 OPERATING SYSTEM-SPECIFIC INFORMATION ... 141
Unix /Linux ... *141*
5.22 M IMPLEMENTATION-SPECIFIC INFORMATION ... 142
GT.M Information ... *142*
MSM Information ... *143*

APPENDIX I: INSTALLATION AND CONFIGURATION PLANNING ... 145

APPENDIX II: CHARTS FOR CONFIGURATION PLANNING ... 148

Access ... *148*
Users ... *149*
Systems ... *150*
Physical Locations ... *151*
Logical Locations ... *152*

Location Classes *153*
Transfer Routes *154*
Change Request Types *155*
Variants *156*
Objects *157*
Modules *158*

APPENDIX III: INSTALLATION AND CONFIGURATION CHECKLISTS 159

VC/m Server Installation Checklist *159*
Essential VC/m Configuration Settings *161*
Additional VC/m Configuration Settings *163*
VC/live Installation Checklist *166*
VC/live Configuration Settings *168*
Upgrade Checklist *170*

GLOSSARY 171

INDEX 173

1 Configuration Management

Configuration Management is the art of organizing, documenting and controlling systems which are continually evolving. It originated in the aerospace industry in the 1950s to enable accurate re-creation of the prototypes which were tested at various stages of the project. It has since been applied to many design and manufacturing processes.

Software Configuration Management has 4 main facets:

- Version Control - tracking the different software versions which exist as on-going application development work takes place.
- Environment Management - the control, at each stage of the project, of which version belongs in each environment, e.g. in development, in testing or released to the live site.
- Release Management - the control of software building for a release.
- Process Control - the implementation of in-house procedures for software development.

Good Configuration Management has a key role in any mature software development process. It brings a number of important advantages:

- greater productivity
- an increase in software quality
- detailed management reporting
- faster response to customer needs and market changes
- a reduction in costs

Automating the process reduces the amount of time and effort which is expended on Configuration Management, while increasing the level of control.

2 Introduction to VC/m

2.1 VC/m Overview

VC/m is an automated Configuration Management system. It can be configured to manage a wide range of situations, from the development work of a small in-house team to the development and distribution for a large software house.

For version control, VC/m uses a database which is known as the library. The library holds a reference copy of each version of each object. This includes current live versions and a number of previous versions. Earlier versions can periodically be archived.

VC/m's Configuration Management database holds the following information:

- the locations which are declared to VC/m
- the locations between which transfers are allowed
- the systems which are installed in each location
- the objects which belong to each system
- the versions of each object which are stored in each location
- the versions of the components which belong to each object version
- the access levels of each user

All this information is used to control the process of transferring object versions between different locations.

Several kinds of transfer are defined in VC/m. Developers can transfer an object version from the library to their development area. This is known as "checking out". From there, the object version may be transferred to a test area. When the work has been completed, the object version is "checked in" - moved back to the library. There are also standard transfers, and automatic transfers which are triggered when a defined set of conditions is satisfied.

All transfers in VC/m are controlled by a standard transfer mechanism which moves object versions from place to place and updates their status. This standard mechanism has several front-ends, both general purpose ones and ones which serve specific purposes, such as Check-out and Check-in.

Several facilities are available which help developers in their work. An editor can be interfaced with VC/m so that changes made are automatically recorded in the VC/m database. A search facility enables the user to search through several different versions of an object for a particular string. Different object versions at different locations can also be compared to identify the precise changes which have been made.

When a project or enhancement requires changes to several different objects, a change request can be used to identify all the object versions concerned. This enables them to be transferred together in a single step and without the risk of missing out one of the parts.

When a major new project is begun, the baseline facility can be used to create a whole new variant using the latest versions of the required objects.

VC/m automatically controls concurrent development. It ensures that no changes are lost or inadvertently overwritten. When concurrent development work has been merged, there is a facility for updating the VC/m database. As an additional control, a report can be printed showing all instances of concurrent development.

When development work has been completed, a user with the correct access level can set the release date for the software. The bulk transfer facility is then used to make the actual release. This selects any object versions from the designated location which have a current release date, and transfers them to the appropriate live system.

When the live system is physically on a different machine from the development system, the release process creates a sequential file which contains the objects to be released. VC/live can be used to make the installation on the remote system. This ensures that the correct files are installed in each location. It can also create a hot backup and perform the rollback if any problems should arise with the new software.

VC/m maintains a detailed audit trail of all changes and transfers. A complete record is available for each version of each object and for each date. The audit trail also keeps a record when objects are merged together and when new objects and variants are created by copying an earlier object.

Various management reports are available from VC/m. The Summary report provides a breakdown of all transfers by type of transfer and by type of change for any given period. The Matrix report shows the status of each object version at each location. The Overdue Objects report shows all objects which have been checked out and have not been checked back in by the expected date. This provides control over work that had been started, but has been suspended or abandoned without being returned to the library. The Release control sheet lists the objects which will be included in a new release. This allows the contents of the release to be checked, and release documentation to be prepared.

2.2 VC/m Concepts

Components and Component Types

In VC/m, a **component** is a physical unit of an application, for example a file or other entity which will be managed by VC/m. On a website, for example, index.html and company_logo.gif would both be components.

The component name is normally the same as the name of the physical entity. It can be up to 200 characters long. For a binary or text component, it often contains a partial path.

When a change is made to a component, a new version is created. Each version of a component has a **date /time stamp**. This is the date and time which is recorded in VC/m's database for when this component version was last edited.

For each component, the **component type** is specified. This is a one to three character code that identifies the characteristics of the component for storage and transfer.

The standard component types include text and binary files. Other types are available where required, including M routines and M globals. Additional types can be purchased as a special option, or customized drivers can be written on site.

Object Versions, Objects, Variants and Versions

The main unit which is used in VC/m is an **object version**. This is a logical grouping of one or more physical components at a specific stage of development.

The full, unique object version name is made up of three parts and is given in the form:
object base/variant.version, e.g. ABC/3.0.

The **object base** is normally used to group together a unitary part of the application software. In practical terms, for a website, an object base would typically be a web page and the associated graphics files.

A **variant** is used to identify a high-level design difference. Any number of variants can exist in parallel over an extended period of time. It can be used, for example, for different language versions (English /Spanish /French), a customized version, or a major difference between two versions of an object.

In VC/m, the combination of object base and variant is known as an **object**. It has a unique, two-part name, e.g. ABC/2.

Each instance of the object also has a **version** number. The version is incremented by one every time a change is made to the object, for example because of a correction, a bug fix, the addition of new information or the development of new functionality. ABC/3.0, for example, is version 4 of object ABC/2.

Together the object base and the variant code can have one to forty characters. Letters, numbers and most punctuation characters can be used. The version number is always an integer.

A component can only be registered to VC/m when it belongs to an object version. Each object version can have several components of different types.

A single component may belong to more than one different object. Also, the same version of a component may belong to more than one version of the same object.

Change Requests, Change Request Types and Change Request Status

A **change request** is the definition of a change which is made. It specifies the object versions which are to be changed and the reason for the change. It therefore provides a record of which object versions have been changed in connection with any one task.

A change request is used to group objects together so that they can easily be moved as one unit. When a prompt in a VC/m transfer function asks for an object, a change request can be entered instead, prefixed with @.

A change request code has between one and ten alphanumeric characters.

Each change request is of a specified type, e.g. edit, bug fix, development project, customization work. The types which are available are specific to the VC/m installation.

For each change request type, change request statuses can also be defined. These define the sequence of stages which the change request must pass through. Each stage can optionally be tied to a location.

Physical Locations, Storage Formats, Locations and Location Classes

A **physical location** is a place where the components of an object version are physically stored, e.g. an operating system directory. The **storage format** for a physical location is a code which determines the way in which components are stored in this location.

A **location** (also called a **logical location**) is the conceptual place where the object version is located, e.g. live, development, testing, tested. Each location is mapped onto one or more physical locations. This means that an object version can have components which are stored in different storage formats, for example, an image which is a binary file, a web page which is a text file, and a Java applet which is stored in Visual SourceSafe.

Each physical location may have more than one logical location mapped onto it. This can be used to divide the contents of a physical location into groups. For example, a test area may have a logical location for object versions undergoing testing and one for object versions which have been tested. Object versions can then be changed from TESTING to TESTED, without physically moving them.

All transfers within VC/m are specified in terms of logical locations rather than physical ones. This makes VC/m very flexible. For example, if the physical location of the test area is changed, it is not necessary to re-define each transfer from or to it.

A **location class** is a group of locations. It provides a convenient way of grouping several logical locations together for transfer operations. For example, when there is more than one live location which the software is released to, they can all be placed in a location class called LIVE. When a prompt in VC/m asks for a 'to location', a location class can be entered instead. It is distinguished from a location name by prefixing with @.

Single-Version and Multi-Version Locations

Each physical location is defined as capable of storing only a single version or multiple versions of each object at any one time. When a new version is transferred to a single-version location, its ancestor is displaced. DOS /Windows and UNIX operating system directories are only able to store a single version of an object at any one time. Libraries and sequential files, however, may be defined either as single-version or as multi-version, depending on the role they are to be used for.

Logical locations can also be defined as single-version or multi-version. A logical location which maps onto a multi-version physical location may be defined as single-version or multi-version. However, by definition, a logical location which maps onto a single-version physical location must be single-version.

> *Note: Certain locations, e.g. DOS /Windows and UNIX operating system directories, also have a physical limitation in that they can only hold a single version of any component name at any one time. This means that when a component of the same name belongs to more than one object, only one of the component versions can exist in such a location.*

Controlled and Uncontrolled Locations

A physical location has a type of controlled or uncontrolled. A **controlled** physical location is one which VC/m expects to have full knowledge of.

An **uncontrolled** physical location is one whose contents VC/m does not assume it has exclusive control of. Uncontrolled locations can be referenced within VC/m in exactly the same way as controlled locations. The difference is that VC/m does not perform any checks on the contents of an uncontrolled location before making a transfer to it.

Controlled is the default and this should normally be used.

The Library

The library is a set of one or more multi-version physical locations which together are used to store the master copies of all object versions. This database is an essential part of a correctly set-up VC/m system.

The library functions as both a reference library and a repository library. It contains the master copies of all object versions. For each object, it stores both the most recent version and as many earlier ones as the users wish.

The master copy of an object version is taken out of the library (**checked out**) for development work. Once the work has been completed, the master copy is returned to the library (**checked in**) before being released.

In a more complex system, there may be more than one location which functions as the library.

Systems

A **system** is a group of objects which are installed together to form a working set of software. This is used to make the task of specifying which objects belong at which locations both simple and flexible.

The system code has one to eight alphanumeric characters.

When systems are used in VC/m, each object can be defined as belonging to one or more systems. Each system is defined as being installed at one or more locations.

Each variant of the object base may belong to a different set of systems. Note that the list of systems is held at the variant not the version level.

Each system has a list of locations at which it is installed. In order for an object to belong at a location, it must belong to at least one system which is installed at that location. An object which is not associated with a system will only belong at locations which also have no systems.

For example:

System	Objects
CORE	AA001/1, AA002/1 ...
CUSTOMER	CUSTAAA/1, CUSTAAB/1, ... LIB001/1, LIB002/1, ...
SUPPLIER	SUPP001/1, SUPP002/1, ... LIB001/1, LIB002/1, ...

Location	Systems
CORELIVE	CORE
CUSTLIVE	CORE, CUSTOMER
SUPPLIVE	CORE, SUPPLIER
DEV	CORE, CUSTOMER, SUPPLIER

In the example, the CORE system belongs at every location. The SUPPLIER and CUSTOMER systems each have their own live areas, but share the development area. Note that some objects, such as SUPP*, only belong to one system, whereas other objects, such as LIB*, belong to two systems, and can therefore be transferred to any location where either system belongs.

By linking each object to one or more systems and by associating each system with the location or location where it will be installed, VC/m is able to determine which objects belong in which locations.

> *Note: When a transfer function is used to transfer an object to a location or location class, VC/m will only transfer the object to the location if it belongs to a system which is installed at that location.*

Existence and Object Status: Master, Active and Error

A component **exists** at a location if it is physically present at the location.

A component is **registered** to VC/m at a location if it belongs to an object version which is active at that location. It is possible for it to be registered but not exist at a location.

An object version **exists** at a location if all its components exist there, and the object version is recorded in VC/m's database.

An object version which is recorded at a location in VC/m's database has a **status**, either active (A) or error (E). At one of the active locations, the object version will have an additional status of master (M), except in VC/live.

Note: An object must be active in at least one location in order for it to have components.

Master means that the copy at this location is the primary copy of the object version. By definition, each object version can have only one master location. VC/m will not overwrite or delete the master copy. This is always the one which should be used as the basis for development or release.

Active means that the copy at this location is a version which can be used by VC/m as the source for a transfer and to satisfy location dependencies. Each object version can be active at one or more locations. It may or may not exist at a location where it is active.

Error means that the copy at that location is not a complete or integral copy. Normally this is because an error occurred during a transfer of an object comprising of multiple components to the location. One or more components will have failed to transfer due to errors such as no write access, out of disk space or network failure. An object version with a status of error at a location cannot be used as the source for a transfer.

Transfer Routes and Function Codes

A **transfer route** is a path that an object version is moved along. They are used to define which transfers are allowed and by who. They include where the object version is moved from, where it is moved to, access codes, whether the transfer is a move or a copy and whether the version number is incremented.

Transfer functions in VC/m can only be used when a transfer route for the function has been set up. For example, to allow developers to check out objects from a library location called LIB to a development area called DEV, a transfer route from LIB to DEV must be set up for the check-out function.

The **function code** identifies the type of transfer for which the transfer route is used.

Transfer Route Dependencies

A **dependency** for a transfer route is a condition on the transfer. There are several types of dependency which can be set, depending on the type of transfer.

Dependent Location

A **dependent location** is a condition that an object version can only be transferred if the version is already active at the dependent location. This provides a mechanism whereby an object has to meet certain conditions before it can proceed to the next step in the development or release process. For example, a dependency can be set up such that an object version cannot be checked back into the library unless it is active at location TESTED. The quality manager may be the only user authorized to transfer objects from DEV to TESTED, thus ensuring that untested software cannot be placed in the library. The dependency is only applicable if the system controls allow the object version to be transferred to the location. Location classes can also be used to set up dependent locations.

Installation-Specific Dependency

An installation-specific dependency allows any condition which the user chooses to be applied to a transfer.

Status Date

A **status date** is a date which is associated with an object version. Depending on how the transfer routes are defined, the status date can be entered when a standard transfer is made. It can then be used as a condition on a volume transfer.

If a status date has been specified, only those objects that have a status date earlier than or equal to it will be selected. Any objects without a status date will also be selected. If no status date has been entered, any object with a status date that is in the past will be selected.

Access Codes

An **access code** is used to control which users are allowed to use a transfer route. Access codes can also optionally be used to control access to a menu or menu option and to determine which locations are displayed on the browser interface.

Each user ID has an access code or codes for that user. Each transfer route has an associated access code which a user must have in order to use it. If access security is used for menus or locations, then these items optionally also have an access code.

Modules

A **module** is a template for the creation of a new object and its components. It can automatically allocate the next object name in a sequence. It can also generate skeleton routines for the objects.

2.3 VC/m Transfers

Transfers

When an object version in VC/m is moved or copied from one location to another, a **transfer** takes place. Before an object version can be transferred, a transfer route must be defined. Certain conditions must also be satisfied which make the transfer valid and therefore allowed by VC/m.

The transfer route defines whether the object version is moved or copied. If the transfer is a **move**, the object version is removed from the 'from location' after it has been transferred. It no longer exists there. However, if the 'from location' still contains the master copy of the version, this is not deleted. If the transfer is a **copy**, it is retained in the 'from location' as well as the 'to location', so it exists in both.

When an object version is transferred to a single-version location, it will **displace** any ancestor version. This means that the previous version is removed and replaced by the new version.

Transfer Mechanisms

There are two, slightly different, mechanisms which are used for making transfers, depending on which option is selected from the menu.

Standard transfers can be made using the Transfer, Out and In options on the menu. There may also be additional customized menu options for standard transfers. Dependent locations and installation-specific dependencies can be used as conditions on standard transfers.

Volume transfers can be made using the Bulk transfer or the Release options. In these options, dependent locations, installation-specific dependencies and status date can be used as conditions on the transfer.

The Archive and Purge option also makes a specific type of volume transfer.

Processing

When a transfer takes place, the components of the object version are physically transferred to the new location. If the object version is moved, the components are deleted from the 'from location' after they have been transferred. If it is copied, they are retained in the 'from location' as well as the 'to location'.

If the transfer causes one or more object versions at the 'to location' to be displaced, the database is updated for each of the displaced objects to show that it no longer exists at that location.

If VC/m is transferring to a controlled, single-version location and the location contains a component of the same name which has not been registered (i.e. does not belong to any object version), the transfer is not allowed. If VC/m is transferring to an uncontrolled location, it will automatically overwrite any existing version of a component, without making any checks.

> *Note: Certain locations, e.g. DOS /Windows and UNIX operating system directories, have a physical limitation in that they can only hold a single version of any component name at any one time. This means that when a component of the same name belongs to more than one object, transferring one of the objects to the location will overwrite the component of the other object. VC/m will logically deactivate the object whose component has been overwritten.*

If a previous object version is at the 'to location' when VC/m is transferring to a controlled, single-version location, and the new object version does not include one of the components which belonged to the previous version, it will delete the redundant component from the location. If the new version of the object has no components, transferring it deletes from the 'to location' all the components which previously belonged to the object version.

If an object version already exists at the 'to location' but the object no longer belongs to any system that is installed at that location, transferring the object version to the location causes it to be deleted from that location. This also physically deletes the components of the object version from the location.

In all of the above cases, if the transfer is to an uncontrolled location, the components will not be deleted.

If a transfer is a copy between two locations which map onto the same physical locations, no physical transfer of components takes place. However, a move will cause them to be physically deleted from a controlled location.

When an object version is moved to a multi-version location, it will exist alongside any previous versions of the object which were already at the location. Each version of each component will have one copy stored at the location. The VC/m database is updated with information about which version of which component belongs to each object version.

When a transfer takes place, the status of the affected object versions is updated.

If a list of new active locations is defined for the transfer route, the object version is made active at each of these locations. If the object does not belong to a system which is installed at that location, it is not made active. This happens irrespective of whether the object version physically exists at the location. For example, a transfer from DEV to LIBRARY may activate both the LIBRARY and the UNRELEASED locations. When the same object version is transferred from LIBRARY to LIVE the UNRELEASED location may be de-activated. This does not require the object version to ever physically exist at the location UNRELEASED.

If a new master location is defined for the transfer route, the location of the master copy of the object version is changed to the specified location. If this field is blank, the location of the master copy is unchanged.

An audit trail entry is written for each object version which is transferred.

Valid Transfers

There are a number of conditions, all of which must be satisfied, in order for a transfer to be valid.

1 A transfer route for the transfer must be defined.

2 The user must have the correct access level.

3 The object version must exist at the ‘from location’, i.e. all the components of the object version must physically exist there.

4 The object version must have a status of active at the ‘from location’.

5 If there are systems installed at the ‘to location’, the object must belong to at least one system which is installed there. If no systems are installed at the ‘to location’, the object must not belong to any systems. Alternatively, this condition will also be satisfied if the object version to be transferred already exists at the destination location but the object no longer belongs to any system which is installed at that location. In this case, the object version will be deleted from the destination location.

6 If the ‘to location’ is a single-version location and a version of the object exists there, it must be the same version or an ancestor of the version being transferred.

7 If the ‘to location’ is a single-version location, it must not contain the master copy of any version of the object.

8 If the ‘to location’ is a controlled, single-version location, it must not contain any unregistered component with the same name as a component in the object version to be transferred.

9 If the transfer route has dependent locations which are relative, each dependent location must contain an active version of the object which is the same version or a later version as the one at the ‘from location’. If the transfer route has dependent locations which are absolute, each dependent location must contain an active version of the object which is the same version as the one at the ‘from location’.
If the object does not belong at a dependent location because of the system controls, VC/m will ignore this dependency.

10 If the transfer route has dependent locations, and the latest object version at a dependent location is the same as the object version to be transferred from the ‘from location’, the dependent location must have a version of each component which is the same as or later than the version at the ‘from location’.

11 If the transfer route has installation-specific dependency conditions, these must be satisfied.

Standard Transfers

Standard transfers can be made using the Transfer, Out and In options on the menu. There may also be additional customized menu options for standard transfers. Dependent locations and installation-specific dependencies can be used as conditions on standard transfers. The transfer can be invoked for an object version or for a change request.

As the prompts for object, from location and to location are entered, VC/m makes checks that the transfer route is valid. It also checks that the from and to locations are accessible. If the route is not valid or the locations are inaccessible, an error is given.

The exact type of processing performed by the transfer depends on the transfer type (move or copy), whether the 'to location' maps to controlled or uncontrolled physical locations, and the new active and master statuses.

Once the transfer is invoked, VC/m attempts to transfer all the object versions listed to the 'to location'. If it is unable to transfer an object version for any reason, a message will be displayed at the time when the transfer is performed. The user is asked whether they wish to transfer the remaining objects. No terminates the transfers at this point; Yes will cause VC/m to continue until the next transfer error; All causes VC/m to continue without prompting if further errors are encountered.

The 'from location' prompt can be filled with either a location code or blank. It can be left blank if all object versions belonging to a change request are being transferred. The 'from location' will then be taken as the master location of each object version. This can be used to allow a selection of object versions that are at several different locations to be moved to one common location.

The 'to location' prompt can be filled with either a location or a location class. Entering the location class code (prefixed by @) enables transfers to be made to multiple locations. For example:

```
To location          [@LIVE                    ]
```

The following combinations are possible for the input prompts:

- Object version from a location to a location
- Object version from a location to a location class
- Change request from a location to a location
- Change request from a location to a location class
- Change request from blank location (interpreted as the master location) to a location
- Change request from blank location (interpreted as the master location) to a location class

Volume Transfers

Volume transfers can be made using the Bulk transfer or the Release options on the menu. In these options, dependent locations, installation-specific dependencies and status date can be used as conditions on the transfer. The transfer is invoked for a list of object versions, which may or may not be linked by a change request.

As the prompts for from location and to location or class are entered, VC/m makes checks that the transfer route is valid. It also checks that the from and to locations are accessible. If the route is not valid or the locations are inaccessible, an error is given.

When the 'from location' and optional status date have been input, VC/m selects the objects which match the criteria. Every object in the 'from location' which also matches the status date criteria will be selected. If a status date has been specified, only those objects that have a status date earlier than or equal to it will be selected. Any objects at the selected location without a status date will also be selected.
The exact type of processing performed by the transfer depends on the transfer type (move or copy), whether the 'to location' maps to controlled or uncontrolled physical locations, and the new active and master statuses.

Once the transfer is invoked, VC/m attempts to transfer all the object versions listed to the 'to location'. Each location is processed in turn. If the location is local, all transfers are performed directly. If the location is remote, a sequential transfer file is created and all transfers for this location are written to the file.

If VC/m is unable to transfer an object version for any reason, it will display a message and continue with the transfer of the next object. Certain types of error are recorded in the audit trail, but others are not. For example, if the object version does not belong to any system which is installed at the 'to location', a message is displayed but no error is recorded. If an error occurs, more information about the cause can be found by trying the same transfer with a standard transfer.

Automatic Transfers

An automatic transfer is a transfer which is triggered when its dependency conditions are satisfied. It is made automatically by the VC/m system, rather than manually by a user using one of the transfer functions. It is defined by the use of the reserved function code AUTOT. Each time VC/m transfers an object version, it checks through its list of automatic transfers and makes any ones whose conditions are satisfied.

Tied Locations

When a change request type is defined, it is possible to define the statuses through which any change request of that type must pass. These can optionally have a tied location.

Where there is a tied location, when the change request status is changed in change request maintenance, a transfer will automatically be invoked. This transfers all the objects in the change request to the tied location. As with any other transfer, a transfer route must have been set up and the conditions for a valid transfer must be satisfied for the transfer to actually take place.

3 Using VC/m for the First Time

3.1 Starting VC/m's Browser-Based Interface

The browser-based interface is the preferred way of interacting with VC/m. See page 48 for instructions. This interface requires Microsoft Internet Explorer.

See Chapter 1 of the *VC/m Reference Manual* for information about using the browser-based interface.

3.2 Starting VC/m's Character-Based Interface

To run VC/m's character-based interface, log into the M system and enter the following command:

```
do ^%vc
```

You are presented with the following main menu screen, after prompting for your user ID if this is necessary:

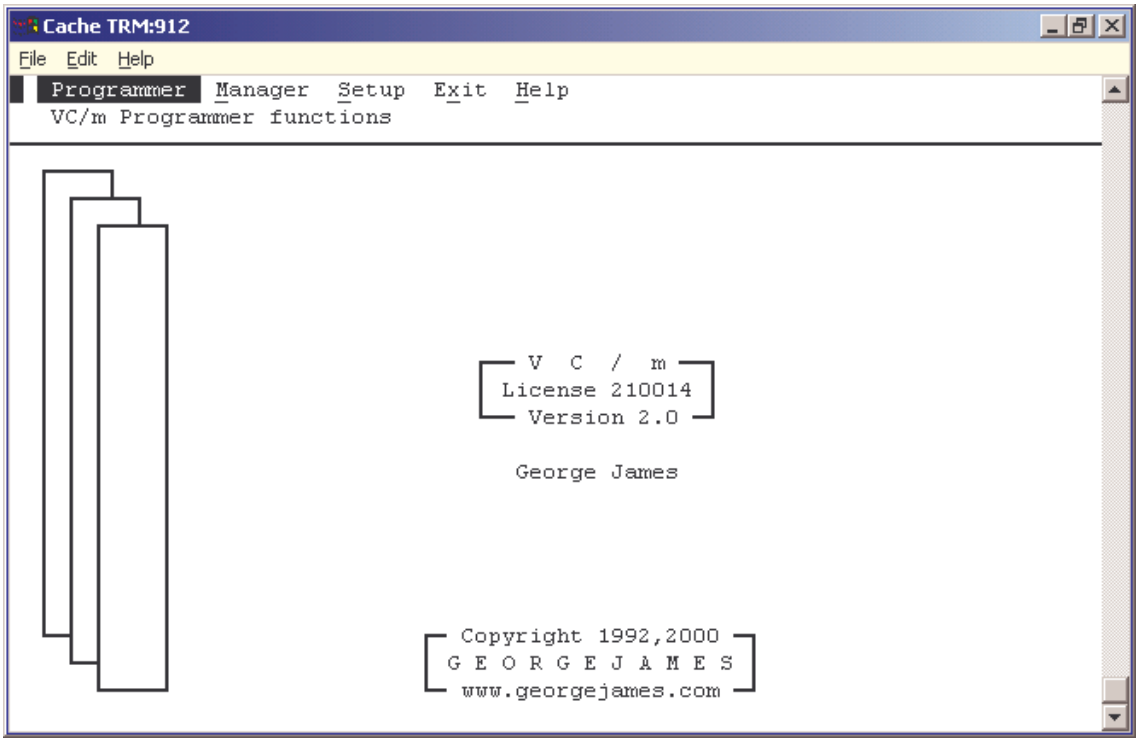


Figure 3.1 VC/m main menu screen

3.3 Getting Around in VC/m's Character-Based Interface

A menu bar is displayed at the top of the screen. Use the left and right cursor keys to move along this, and <return> to select the highlighted option.

On the line below the menu options, messages are displayed. These describe the highlighted option, or report errors and warnings.

Use the up and down cursor keys to move about the screen.

To make an entry in a field, type the entry and then press <return>. To delete an entry, enter a space and then <return>. Type ? and <return> to obtain help or a selection list of valid entries.

More detailed information can be found in the VC/m Reference Manual.

4 Installation Guide

4.1 System Requirements

VC/m requires an M implementation to be installed on the server. The following are supported:

Windows 2000 or Later

- Caché or Ensemble
- MSM

Unix /Linux

- Caché or Ensemble
- GT.M
- M21
- MSM

OpenVMS

- Caché or Ensemble
- GT.M
- DSM

Additional software is required for the browser interface. Please check the relevant section before proceeding.

4.2 Installation Procedure

Planning

Before you configure and use VC/m, it is essential that you plan how you intend to use it and design the configuration management procedures which you want to use.

Appendix I: Installation and Configuration Planning gives guidance on the questions which need to be addressed before proceeding.

Installation

Appendix III: Installation and Configuration Checklists has a checklist which should be used to ensure that all of the necessary steps are completed when installing the VC/m software for the first time.

More detail on some of these steps is given in the following sections.

For those who are not familiar with M systems, a basic M guide is provided.

After installation, you should consult the relevant M implementation-specific section (p. 142-57).

Upgrade

If you wish to upgrade a 1.x system, please consult George James Software for advice. When upgrading a VC/m installation of version 2.0 onwards, the procedure is similar to the software installation steps for a new installation.

Appendix III: Installation and Configuration Checklists has a checklist which should be used to ensure that all of the necessary steps are completed.

4.3 Server Installation

Installing the VC/m Program Files

If you are installing VC/m in a configuration where the M server and the web server are on separate machines, you must install the *complete* set of VC/m files on both machines.

The following directories are recommended or suggested for installation:

Operating System	Directory
Windows	C:\Program Files\George James Software\VCm *(recommended)*
UNIX /Linux	/usr/vcm *(suggested)*
OpenVMS	*disk:*[VCM] *(suggested)*

If your software is supplied on CD:

1 Copy the files to a directory on the server where you plan to install VC/m.

If your software is supplied as a zip file:

1 Copy the zip file to a directory on the server where you plan to install VC/m.

2 Unzip it using the appropriate method for your operating system:

Operating System	Unzip method
Windows	Use WinZip, pkunzip or similar software
UNIX /Linux	$ unzip -a vcm.zip
OpenVMS	$ UNZIP VCM.ZIP /BINARY=AUTO /TEXT=AUTO (The Info-Zip version of UNZIP.EXE is available from ftp://ftp.info-zip.org/pub/infozip/VMS/)

When unzipping, it is important to be sure that text files are unzipped as text files.

3 On OpenVMS the unzip typically creates the text files with Variable record format. On Caché it is necessary for these to be converted to Stream format. To achieve this VC/m includes a file called VAR_TO_STREAM.COM which is a DCL procedure that scans a directory and its subdirectories, converting files having Variable record format. The conversion uses the FDL specification in VCM_STREAM.FDL. To invoke it, switch to the directory containing these two files, and then invoke the procedure as shown below. For example, if the COM and FDL files unzipped to DISK$MAIN301:[VCM] the commands are:

```
$ SET DEF DISK$MAIN301:[VCM]

$ @VAR_TO_STREAM DISK$MAIN301:[VCM...]*.*
```

Each converted file will be reported. The unconverted files will remain with predecessor version numbers.

Preparing the M Environment

Configure your M environment in preparation for the installation of the VC/m routines and globals.

Caché or Ensemble

By default the VC/m routines would be installed into the CACHELIB database, with the result that they will be deleted each time Caché is upgraded. Additionally, from Caché 5.1 onwards the CACHELIB database is by default mounted read-only, preventing the VC/m routines from loading. We therefore recommend the following configuration:

1 Create a database, e.g. VCM, for the VC/m application.

2 Create a namespace which has this database as its primary database for both routines and globals.

3 In this new namespace, map the globals (%vc*) and the routines (%vc*) to the same database. (This will override the default that % items are stored in CACHELIB.) For example (5.0 Configuration Manager shown below; on 5.1 or later use the System Management Portal):

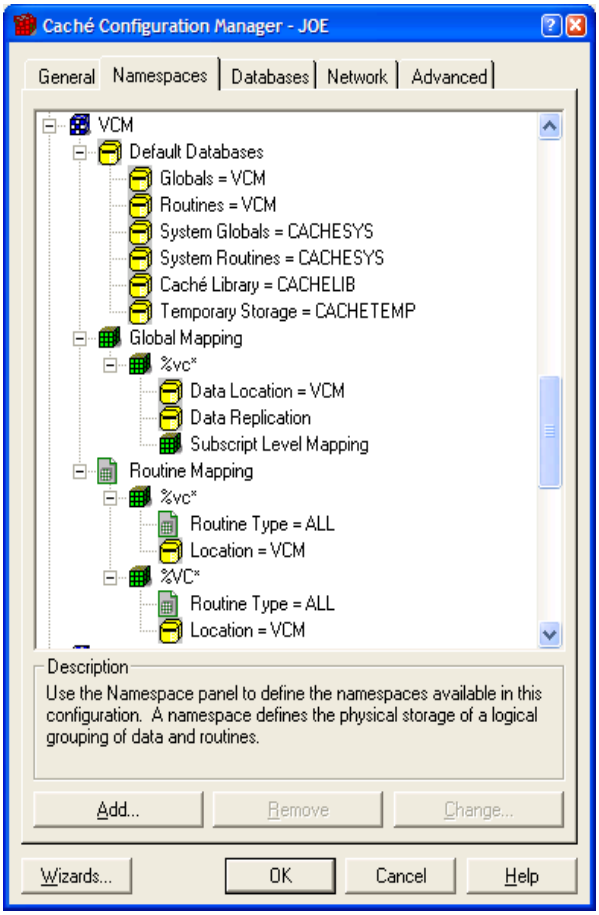


Figure 4.1 VCM namespace for Caché

4 For each namespace which is controlled by VC/m (probably all of them), add the same mappings as are given above. The mapping of %VC* routines shown in the screenshot above is unnecessary for new VC/m installations because the uppercase alias commands have been removed.

Note: The most efficient way of adding all the mappings is by editing the CACHE.CPF file.

By default, on Windows systems, Caché background processes and telnet sessions only have access to the local file system. If you want VC/m to manage files on other nodes in the network, the Caché service must be started with an account which has appropriate network access rights. Edit the Log On As parameters of the Caché Windows service to use suitable credentials. Alternatively (Caché 5.0 or earlier only) do the following:

a. Launch the Caché Configuration Manager (e.g. from the blue Caché cube).

b. Click on the Advanced tab.

c. Open the section labeled Input/Output.

d. Enter values for the Network Server Username and Network Server Password. These must correspond to a local or domain account which has suitable rights.

Note: The password is stored in the CACHE.CPF file in a very weakly-encrypted form. This could expose a serious security risk if you allow public access to your Caché server without the protection of a properly configured firewall.

GT.M

1 Create a GT.M database file (mumps.dat) for VC/m in the installation directory. It should have a minimum record size of 512 and a key size of 255. It is also recommended that the mumps.dat file name be specified explicitly with a full path specification rather than a relative path.

Note: Do not use "./mumps.dat" or "mumps.dat" as this causes problems when extended global references are used.

The following is an example of how to use gde and mupip to create a suitable database:

```
$gde
GDE> change -region default -record_size=512 -key_size=255
GDE> change -segment default -file_name="/usr/vcm/mumps.dat"
GDE> exit
$mupip create
```

2 If you want all users to have access to the VC/m character interface from any directory, add the VC/m directory "/usr/vcm" to the gtmroutines environment variable for all users.

3 Add a mapping to each mumps.gld, directing access for ^%vc* globals to the VC/m database file.

```
$gde
GDE> add -segment VCM -file_name="/usr/vcm/mumps.dat"
GDE> add -region VCM -dynamic=VCM -record_size=512 -key_size=255
GDE> add -name %vc* -region=VCM
GDE> exit
```

4 If you plan to use VC/m to manage globals (or sub-trees of globals), you need to ensure that each mumps.gld which VC/m addresses is referencing the mumps.dat database files explicitly rather than using a relative path. This is because VC/m uses extended global references to manipulate globals, and in GT.M these do not operate correctly if the mumps.dat database files are not referenced using an explicit path. For example:

```
$gde
GDE> change -segment default -file_name="/usr/joe/mumps.dat"
GDE> exit
```

DSM

1 If you want to manage routines which have lines longer than 256 characters, you must ensure that the volume sets used to store the VC/m configuration management database and each VC/m library are upgraded to include support for extended string lengths.

2 VC/m uses and updates globals stored in the manager UCI. You should ensure that the protection codes for these globals (^%vc*) are such that they can be read, written and deleted from all UCIs where VC/m will be used.

3 In each UCI where a VC/m library is installed, change the global protection for the global ^vcli to System: RWP and Group: RWP.

M21

1 VC/m uses and updates globals stored in the manager UCI. You should ensure that the protection codes for these globals (^%vc*) are such that they can be read, written and deleted from all UCIs where VC/m will be used.

2 In each UCI where a VC/m library is installed, change the global protection for the global ^vcli to System: RWP and Group: RWP.

MSM

1 VC/m uses and updates globals stored in the manager UCI. You should ensure that the protection codes for these globals (^%vc*) are such that they can be read, written and deleted from all UCIs where VC/m will be used.

2 In each UCI where a VC/m library is installed, change the global protection for the global ^vcli to System: RWP and Group: RWP.

3 Ensure that you are using a partition of at least 30 Kbytes when installing the VC/m routines. Otherwise you are likely to get <PGMOV> errors due to insufficient partition space.

Installing the VC/m Routines

The routines of the VC/m software are supplied in a pair of files called VCM.RO and VCM.RSA. The .RO file is suitable for use on Caché, and the .RSA is for the other platforms.

Caché or Ensemble

If you have configured your system as described under 'Preparing the M Environment', the routines should be restored into the namespace which they are mapped to, e.g. VCM. If you have not configured your system in this way, they should be restored into the %CACHELIB namespace.

Restore the routines using the %RI utility. For example:

```
do ^%RI
Input routines from Sequential
Device: /usr/vcm/VCM.RO   Parameters: ("R")=>
File written by George James Software Limited using VC/m (%RO format)
with extension INT and with description:
VC/m Version 3.2

( All Select Enter List Quit )
Routine Input Option: all Routines
If a selected routine has the same name as one already on file,
shall it replace the one on file? No => No
Recompile? Yes => Yes
Display Syntax Errors? Yes => No

^ indicates routines which will replace those now on file.
@ indicates routines which have been [re]compiled.
- indicates routines which have not been filed.

%vc@     %vcins@   …

nnn routines saved.
```

Alternatively, use the routine import facilities in the System Management Portal.

GT.M

The routines should be restored from VCM.RSA into the VC/m installation directory (e.g. /usr/vcm).

Restore the routines using the %RI utility. For example:

```
$gtm
GTM>d ^%RI
Routine Input Utility - Converts RO file to *.m files.
Formfeed delimited <No>?
Input device: <terminal>: VCM.RSA
George James Software Limited^INT^VC/m
VC/m (%RO format)

Output directory : /usr/vcm/
%vc       %vcins    …

Restored xxx lines in xxx routines.
GTM>
```

Once loaded, the routines *must* be compiled. (Do not rely on GT.M's Auto-Link functionality as this does not handle source files with names greater than 8 characters correctly.) Use the following command to compile all routines in the VC/m installation directory:

```
mumps *.m
```

Syntax errors will be reported, but these can safely be ignored.

DSM /M21 /MSM

The routines should be restored from VCM.RSA into a manager UCI, i.e. [MGR,xxx].

To restore routines, enter the following at the M prompt:

```
do ^%RR
```

Syntax errors will be reported, but these can safely be ignored.

Running the Installation Script

The routine ^%vcins must be run to install the appropriate driver for your operating system and M implementation. This routine also creates root level entries for all manager globals used by VC/m; installs menus and help files and creates icons for the browser interface. In order to create the icons, write access to the VC/m installation directory is required.

Note: If VC/m is being installed on GT.M, set the current directory to be the VC/m directory before running ^%vcins.

Enter the following at the M prompt and then choose the appropriate options for your system. The prompts and messages will vary from the example below depending on your platform and version. You may wish to capture the text in case it is needed for subsequent review.

```
do ^%vcins

V C / m  Version 3.2
The Force of Change
Copyright 1992,2009 George James

For support call +44-1932-252568
or e-mail support@georgejames.com

What type of M system are you running?
     1  InterSystems Caché, Ensemble
     2  InterSystems MSM
     3  InterSystems DSM for OpenVMS
     4  Fidelity     GT.M
     5  M21 Limited  M21
M system (1-5)? <1> 1
What type of Operating System are you running?
      1  Windows 2000/2003/XP/Vista
      2  OpenVMS
      3  UNIX, Linux
Operating System? <1> 1
VC/m - installing Caché drivers ............
VC/m - M Location driver - Installed
VC/m - Packed Sequential File Location driver - Installed
VC/m - Library driver - Installed
VC/m - Binary File Location driver - Installed
VC/m - Text File Location driver - Installed
VC/m - initializing global roots ...

VC/m - Default user ID created = Administrator
Enter the directory specification of the location where VC/m has been
installed <C:\Program Files\George James\Software\VCm>
When a text or binary file is written by VC/m should it be set to read-
only (unless it is checked-out)? <Y>
VC/m - Enable CSP web-access to VC/m?
VC/m - Enable WebLink web-access to VC/m?
VC/m - Enable serverLink web-access to VC/m?
VC/m - Installing menu and help data ...
```

Additional Steps for Caché 5.1 or Later, or Ensemble

During %vcins processing a set of classes, templates and add-ins are loaded. Compilation of the templates and add-ins will report errors on Caché 5.1 or later. This is because a package mapping is required but cannot be defined before the relevant classes have been loaded.

In the Caché System Management Portal, go to Configure Namespaces (use the dropdown list in the upper right of the page). Click ‘Package Mappings’ for the VC/m namespace. Choose ‘New Package Mapping’. From the first dropdown pick the VC/m database. In the second one select the package named VCmStudio. Pick ‘All Local Namespaces’ from the third dropdown. OK the dialog. Save the changes.

At a terminal session in the VC/m namespace:

```
do Cache5^%vcins()
```

This time the compilations should proceed without errors.

Checking the Installation

Check that the installation procedure has been performed correctly by invoking the character-based interface of VC/m. This is done by typing the following command at the M prompt:

```
do ^%vc
```

A user ID is set up automatically for the person who first installed VC/m (see the %vcins output). Use this to log in.

If VC/m has been correctly installed, you will be presented with a screen similar to the following one, showing the VC/m main menu, license number, licensee and copyright notice.

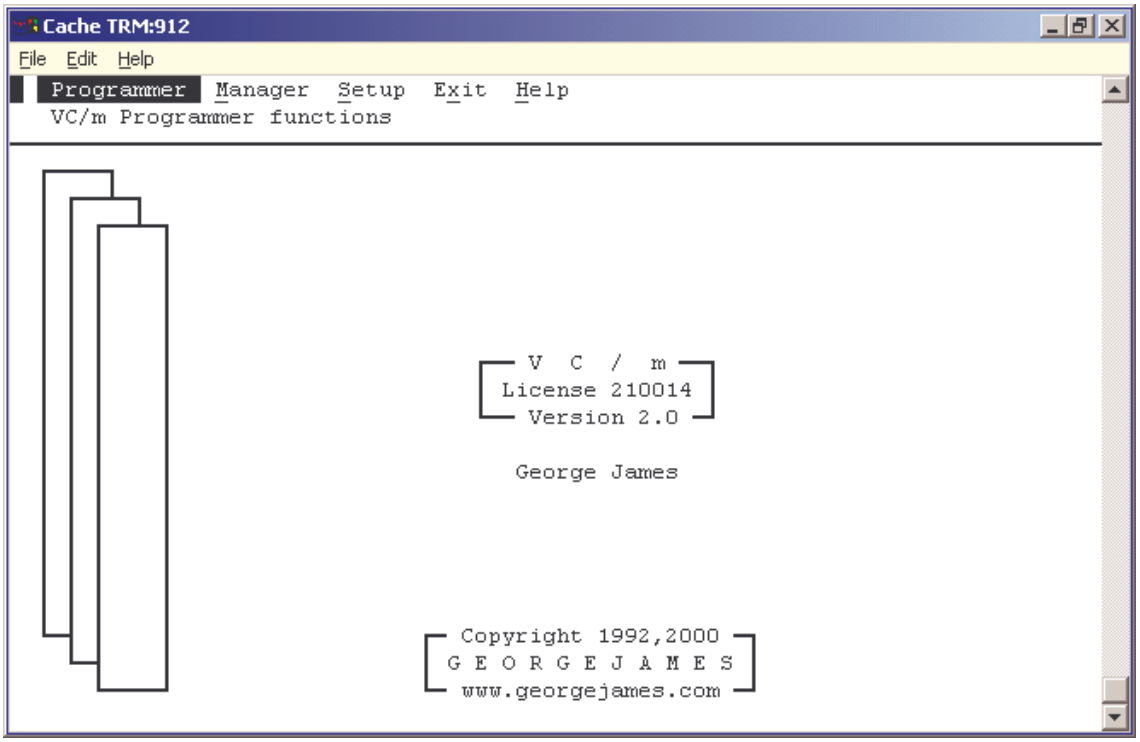


Figure 4.2 VC/m main menu screen

Entering the License Key

The license is normally supplied in a file named vcm.key which the %vcins procedure sets VC/m up to access.

If not, the first time you start VC/m you may receive a message saying 'VC/m is not installed or not licensed' and will be asked to enter your license key.

You will be presented with the following screen:

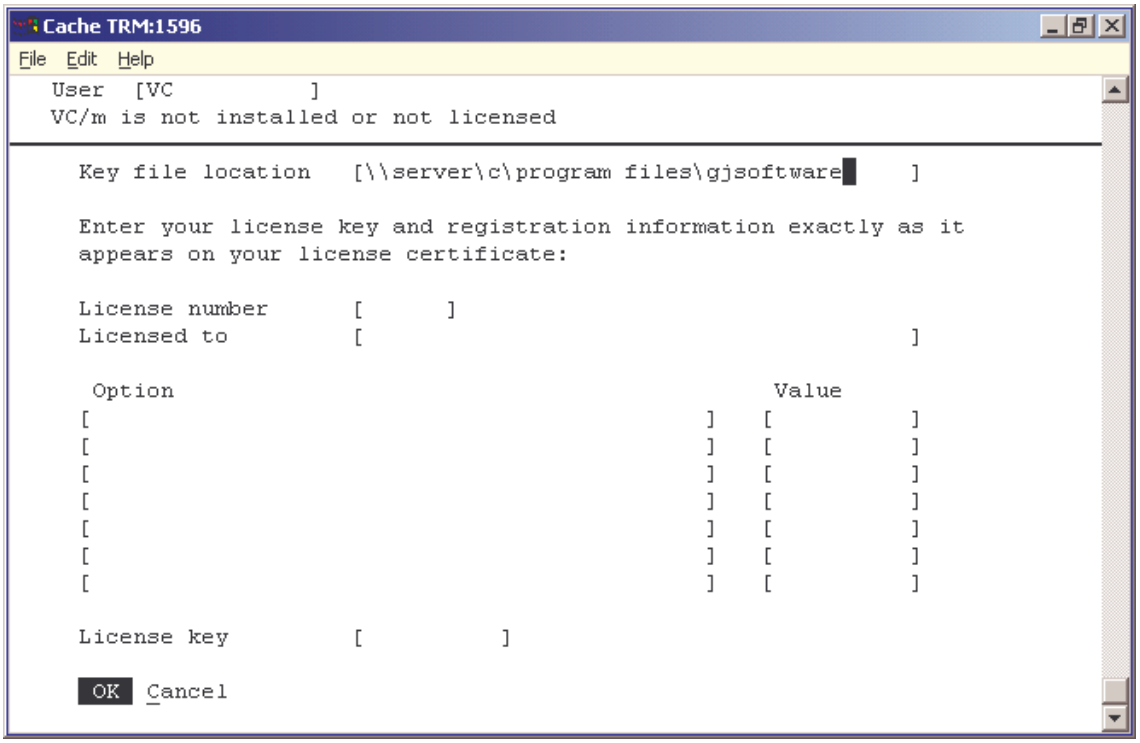


Figure 4.3 License key

Enter the location of your license key. If you are running in a networked environment (i.e. developers with their own Caché instances connecting to VC.m using ECP) it is recommended that you enter the path for the license key location using the whole network path. Do not use a relative path specification. The details of your license will be displayed.

If you are using Caché on Windows and enter a UNC path, ensure that the Caché service is running under an account that has sufficient permissions. The default service account, LocalSystem, is unable to access UNC shares; terminal mode access to VC/m off the Caché console will work but the browser interface will produce a licensing error.

When you have entered the location of the license key, and selected 'OK' to save the information, use the Exit option and select Yes to exit.

4.4 Configuring the Server for Access via the Browser Interface

System Requirements

To enable clients using the browser interface to access the VC/m server, the following must be installed and configured:

1 One of the following web servers: IIS (Microsoft Internet Information Server) or Apache. The web server can be on a different machine from the VC/m server.

2 CSP, WebLink or serverLink interface.

The following combinations are currently supported:

M Implementation	Web Server	Interface
Caché	IIS or Apache	CSP, WebLink or serverLink
DSM	IIS or Apache	WebLink
M21	IIS or Apache	serverLink
MSM	IIS or Apache	WebLink
GT.M	IIS or Apache	serverLink

CSP (Caché Server Pages) was introduced with Caché 4.0 and is an integral part of that and subsequent versions. The web server component of CSP is installed by default when Caché is installed on the same server. It can also be installed on a separate web server. CSP is the preferred option for the interface to Caché because it receives active development and support from InterSystems.

WebLink is a legacy InterSystems product. Where available, it is to be preferred to serverLink because it is faster. It must be installed on the web server.

Note: WebLink is supplied with Caché, but by default it is not installed.

serverLink is supplied by George James Software as part of the VC/m software.

If the web server is on a different machine from the one you installed the VC/m software on, copy the VC/m installation directory and subtree onto the web server using a transfer technique that respects any need to change line-break characters in ASCII files. Perform this copy operation after you have run %vcins.

Once the web server and interface are correctly installed, follow the configuration instructions which are most appropriate for the combination which you plan to use.

Note: If you are using serverLink, you will need a Perl interpreter installed. On Unix platforms, this is normally pre-installed. A Perl interpreter for a Windows platform can be obtained at no cost from http://www.perl.org.

IIS Web Server with CSP Interface

IIS Web Server Configuration

1 If it is not already present, install CSP on the web server by performing a Custom install of Caché, opting to install only the CSP components there.

2 Start the web server management console.

3 Under Default Web Site, create a new virtual directory. The VC/m files which are provided are configured for /vcm, so 'vcm' is recommended as the name of the alias.

4 Map the virtual directory to the directory where VC/m is installed. By default, this will be C:\Program Files\George James Software\vcm.

5 Set the access rights for the virtual directory to Read, and the execute permissions to Scripts and Executables.

CSP Interface Configuration

6 If you have not already done so, enable browser access to VC/m. Enter the following at the Caché command prompt in the VCM namespace:

```
do ^%vcins
```

Press <enter> to move through the options until you reach:

```
Enable CSP web-access to VC/m?  <N> Y
```

Default through subsequent prompts.

7 In Caché Configuration Manager, define a CSP application named '/vcm'. It should use the VCM namespace and its Caché Physical Path should point to the directory where VC/m is installed. By default, this will be C:\Program Files\George James Software\vcm.

8 In the CSP Web Gateway Management pages, extend the Server Response Timeout from its 60 second default to a higher value, perhaps 1800 seconds. Do this either on the Default Parameters page, or if you don't want the new value to apply to other CSP applications hosted by the same web server you should do the following:

   a. Define a new Server Access entry named VCM, copying an existing entry that connects to the correct Caché server (e.g. LOCAL) and then setting the high Server Response Timeout on the new entry.

   b. Define an Application Access entry for '/vcm' and specify that it should use the Default Server named VCM.

IIS Web Server with WebLink Interface

IIS Web Server Configuration

1 Start the web server management console.

2 Under Default Web Site, create a new virtual directory. The VC/m files which are provided are configured for /vcm, so 'vcm' is recommended as the name of the alias.

3 Map the virtual directory to the directory where VC/m is installed. By default, this will be C:\Program Files\George James Software\vcm.

4 Set the access rights for the virtual directory to Read, Scripts and Execute.

WebLink Interface Configuration

5 If you have not already done so, enable browser access to VC/m. Enter the following at the M prompt:

```
do ^%vcins
```

Press <enter> to move through the options until you reach:

```
Enable WebLink web-access to VC/m?  <N> Y
```

Default through subsequent prompts.

6 If it is not already there, install WebLink on the web server.

7 Copy the file 'mgwms32.dll' from your M installation on the VC/m server to the directory where VC/m is installed.

8 Connect to WebLink in your VC/m virtual directory. For example, if your virtual directory is called 'vcm' and is on the local machine, use the following URL: http://127.0.0.1/vcm/mgwms32.dll.

9 Select 'Configure Server Access', enter a new server name (recommended: vcm) and press 'Configure Server'. NB while you can use any name for the server here, we recommend vcm. If you use something different then the default.htm launch page will need to be edited accordingly.

Use the following settings for your new configuration:

```
Connection_Method: TCP Sockets

Server_Type: Intersystem Cache 3.x

Maximum_String_Length: 32000

IP_Address: the IP address of your VC/m Server

Base_TCP_Port: 1972

PCD_Namespace: %SYS

Default_User_NameSpace: the namespace where the VC/m routines reside on
the VC/m server, e.g. VCM
```

10 Save the configuration.

11 From the WebLink main menu, select 'Connect to a Server'. From the drop-down list for 'Server Name', select the server which you have just created. Click on 'Connect'. If the installation is correct, a test page will be displayed.

IIS Web Server with serverLink Interface

serverLink Interface Configuration

serverLink comprises both a set of routines which are supplied and installed alongside the standard VC/m routines and a Perl script.

1 If you have not already done so, enable browser access to VC/m. Enter the following at the M prompt:

```
do ^%vcins
```

Press <enter> to move through the options until you reach:

```
Enable serverLink web-access to VC/m?  <N> Y
```

Default through subsequent prompts.

2 Install a Perl interpreter on the web server host if one is not already installed.

3 Each web server which connects to serverLink needs to have a pass phrase and a description set up. The pass phrase identifies and authenticates the web server to serverLink. The description is used in the serverLink log to identify the source of the connection.

Note: Pass phrases are stored in plain text in the M database and on the web server. They are passed in plain text between the two systems via a TCP socket connection. This can be vulnerable to packet-sniffing etc.

To set up the pass phrases, enter the following at the M prompt in the namespace or UCI where VC/m is installed:

```
do setup^serverLink
```

For each server, enter the pass phrase and then the description, as indicated by the prompts. The pass phrase is a string of up to 60 characters which should be secret. For the description, the URL of the virtual directory which is script-mapped is recommended.

To list the servers which have been set up, enter '?' at the pass phrase prompt. To delete a pass phrase, enter a space in the description field. To exit the set-up, press <return> at the pass phrase prompt.

For example:

```
do setup^serverLink

Configure authenticated web-servers
Pass-phrase?  From Russia with Love
Web-server description http://joe/vcm
Pass-phrase? <return>
```

4 From the VCM directory or namespace, start serverLink as a background job, using the following command:

```
job start^serverLink(6500,0)
```

If you want to stop serverLink, enter the following command. All serverLink daemons will stop within a few minutes:

```
d setStop^serverLink()
```

You may want to add the serverLink start command to your system start-up procedure. For Caché this line can be added to the routine 'ZSTU' in the %SYS namespace to ensure that serverLink is started automatically when Caché is restarted.

IIS Web Server Configuration

5 Start the web server management console.

6 Under Default Web Site, create a new virtual directory. The VC/m files which are provided are configured for /vcm, so 'vcm' is recommended as the name of the alias.

7 Map the virtual directory to the directory where VC/m is installed. By default, if this is the same machine as the VC/m server, this will be C:\Program Files\George James Software\vcm.

8 Set the access rights for the virtual directory to Read, Scripts and Execute.

9 Open the properties window for the VC/m virtual directory.

10 Select the 'Virtual Directory' tab and click the 'Configuration' button.

11 Select the 'App Mappings' tab and click 'Add'.

12 Enter the following values, amending the paths as necessary:

Executable:	`c:\perl\bin\perl.exe "C:\Program Files\George James Software\VCm\serverLink.cgi" "127.0.0.1:6500:`*passPhrase*`"`
Extension:	.DLL
Method exclusions:	
Script engine:	Checked
Check that file exists:	Unchecked

where *passPhrase* is a secret phrase that will be set up on the M server for this web server.

Apache Web Server with CSP Interface

Apache Web Server Configuration

1 If it is not already present, install CSP on the web server by performing a Custom install of Caché, opting to install only the CSP components there.

2 Using the web server management console, or amending the Apache configuration files by hand, publish as '/vcm' the directory where VC/m was installed.

CSP Interface Configuration

3 If you have not already done so, enable browser access to VC/m. Enter the following at the Caché command prompt in the VCM namespace:

```
do ^%vcins
```

Press <enter> to move through the options until you reach:

```
Enable CSP web-access to VC/m?  <N> Y
```

Default through subsequent prompts.

4 In Caché Configuration Manager, define a CSP application named '/vcm'. It should use the VCM namespace and its Caché Physical Path should point to the directory where VC/m is installed. By default, on UNIX /Linux it will be '/usr/vcm' and on Windows 'C:\Program Files\George James Software\vcm'.

5 In the CSP Web Gateway Management pages, extend the Server Response Timeout from its 60 second default to a higher value, perhaps 1800 seconds. Do this either on the Default Parameters page, or if you don't want the new value to apply to other CSP applications hosted by the same web server you should do the following:

   a. Define a new Server Access entry named VCM, copying an existing entry that connects to the correct Caché server (e.g. LOCAL) and then setting the high Server Response Timeout on the new entry.

   b. Define an Application Access entry for '/vcm' and specify that it should use the Default Server named VCM.

Apache for UNIX Web Server with WebLink Interface

Note: The directory paths below relate to Apache on a Red Hat Linux system. The location of these files will vary on other UNIX platforms.

Apache for UNIX Configuration

1 Check that mod_rewrite and mod_cgi are installed in the Apache web server by examining the file /etc/httpd/conf/httpd.conf.

2 Edit /etc/httpd/conf/httpd.conf (or access.conf on some systems). Add the following directive for the vcm directory. Replace «*installation directory*» with the name of the directory where you installed VC/m.

```
# VC/m
Alias /vcm «installation directory»
<Directory «installation directory»>
Options None
AllowOverride All
</Directory>
```

For example:

```
# VC/m
Alias /vcm /usr/vcm
<Directory /usr/vcm>
Options None
AllowOverride All
</Directory>
```

3 Configure the web server to translate nph-mgwcgi to mgwms32.dll.

4 In the VC/m installation directory make a copy of the file '.htaccess.template' named '.htaccess'.

5 Edit the file '.htaccess' and change it to reference the correct IP address and port number of the M server.

6 Restart the Apache server.

For Red Hat Linux:

```
kill -USR1 'cat /var/run/httpd.pid'
```

For other UNIX implementations, try:

```
kill -USR1 'cat /usr/local/apache/logs/httpd.pid'
```

WebLink Interface Configuration

7 If you have not already done so, enable browser access to VC/m. Enter the following at the M prompt:

```
do ^%vcins
```

Press <enter> to move through the options until you reach:

```
Enable WebLink web-access to VC/m?  <N> Y
```

Default through subsequent prompts.

8 If it is not already there, install WebLink on the web server.

The network service demon must also be started.

9 Copy the file 'nph-mgwcgi' to the /cgi-bin directory under Apache.

10 Copy the file ‘mgwms32.dll’ from your M installation on the VC/m server to the directory where VC/m is installed.

11 Connect to WebLink in your vcm virtual directory. For example, if your virtual directory is called ‘vcm’ and is on the local machine, use the following URL: http://127.0.0.1/vcm/mgwms32.dll.

12 Select ‘Configure Server Access’, enter a new server name (recommended: vcm) and press ‘Configure Server’. NB while you can use any name for the server here, we recommend vcm. If you use something different then the default.htm launch page will need to be edited accordingly.

Use the following settings for your new configuration:

```
Connection_Method: TCP Sockets

Server_Type: Intersystem Cache 3.x

Maximum_String_Length: 32000

IP_Address: the IP address of your VC/m Server

Base_TCP_Port: 1972

PCD_Namespace: %SYS

Default_User_NameSpace: the namespace where the VC/m routines reside on
the VC/m server, e.g. VCM
```

13 Save the configuration.

14 From the WebLink main menu, select ‘Connect to a Server’. From the drop-down list for ‘Server Name’, select the server which you have just created. Click on ‘Connect’. If the installation is correct, a test page will be displayed.

Apache for UNIX Web Server with serverLink Interface

serverLink Interface Configuration

serverLink comprises both a set of routines which are supplied and installed alongside the standard VC/m routines and a Perl script.

1 If you have not already done so, enable browser access to VC/m. Enter the following at the M prompt:

```
do ^%vcins
```

Press <enter> to move through the options until you reach:

```
Enable serverLink web-access to VC/m?  <N> Y
```

Default through subsequent prompts.

2 Install a Perl interpreter if one is not already installed. Unix platforms normally have one installed by default.

3 If the path to the Perl interpreter is not /usr/bin/perl, edit the first line of serverLink.cgi. Do not remove the '#' comment symbol.

4 If you are running on a version of GT.M that resolves routine names only to a maximum length of 8 characters, rename serverLink.m to serverLi.m. Use '^serverLi' instead of '^serverLink' in the following instructions.

5 Each web server which connects to serverLink needs to have a pass phrase and a description set up. The pass phrase identifies and authenticates the web server to serverLink. The description is used in the serverLink log to identify the source of the connection.

Note: Pass phrases are stored in plain text in the M database and on the web server. They are passed in plain text between the two systems via a TCP socket connection. This can be vulnerable to packet-sniffing etc.

To set up the pass phrases, enter the following at the M prompt in the namespace or UCI where VC/m is installed:

```
do setup^serverLink
```

For each server, enter the pass phrase and then the description, as indicated by the prompts. The pass phrase is a string of up to 60 characters which should be secret. For the description, the URL of the virtual directory which is script-mapped is recommended.

To list the servers which have been set up, enter '?' at the pass phrase prompt. To delete a pass phrase, enter a space in the description field. To exit the set-up, press <return> at the pass phrase prompt.

For example:

```
do setup^serverLink

Configure authenticated web-servers
Pass-phrase?  From Russia with Love
Web-server description http://joe/vcm
Pass-phrase? <return>
```

6 From the VCM directory or namespace, start serverLink as a background job, using the following command:

```
job start^serverLink(6500,0)
```

For GT.M, a performance benefit can be obtained by starting up multiple serverLink daemons. Simply execute this command, say, five times.

If you want to stop serverLink, enter the following command. All serverLink daemons will stop within a few minutes:

```
d setStop^serverLink()
```

You may want to add the serverLink start command to your system start-up procedure. For Caché this line can be added to the routine 'ZSTU' in the %SYS namespace to ensure that serverLink is started automatically when Caché is restarted.

Note: The directory paths below relate to Apache on a Red Hat Linux system. The location of these files will vary on other UNIX platforms.

Apache for UNIX Configuration

7 Check that mod_rewrite and mod_cgi are installed in the Apache web server by examining the file /etc/httpd/conf/httpd.conf.

8 Edit /etc/httpd/conf/httpd.conf (or access.conf on some systems). Add the following directive for the vcm directory. Replace «*installation directory*» with the name of the directory where you installed VC/m.

```
# VC/m
Alias /vcm «installation directory»
<Directory «installation directory»>
Options None
AllowOverride All
</Directory>
```

For example:

```
# VC/m
Alias /vcm /usr/vcm
<Directory /usr/vcm>
Options None
AllowOverride All
</Directory>
```

9 In the VC/m installation directory make a copy of the file '`.htaccess.template`' named '`.htaccess`'.

10 Edit the file '.htaccess' to reference the correct IP address and port number of the M server, plus a secret pass phrase. For example:

```
"serverLink.cgi?127.0.0.1:6500:From Russia with Love"
```

11 Make serverLink.cgi executable but not editable:

```
$cd /usr/vcm
$chmod 555 serverLink.cgi
```

12 Restart the Apache server.

For Red Hat Linux:

```
kill -USR1 'cat /var/run/httpd.pid'
```

For other UNIX implementations, try:

```
kill -USR1 'cat /usr/local/apache/logs/httpd.pid'
```

Apache for Windows Web Server

Configuration for Use with serverLink Interface

Note: There is a bug in Apache 1.3.22 (win32) relating to mod_rewrite rules which means that this solution does not work with this version of Apache.

1 Check that mod_rewrite and mod_cgi are installed in the Apache web server by examining the file C:\Program Files\Apache Group\APACHE\conf\httpd.conf.

2 Edit C:\Program Files\Apache Group\APACHE\conf\httpd.conf (or access.conf on some older systems). Add the following directive for the vcm directory:

```
# VC/m
Alias /vcm "c:/Program Files/George James Software/VCm"
<Directory "c:/Program Files/George James Software/VCm">
AllowOverride All
</Directory>
```

3 In the VC/m installation directory make a copy of the file '`.htaccess.template`' named '`.htaccess`'.

4 Edit the file '.htaccess' to reference the correct IP address and port number of the M server, plus a secret pass phrase. For example:

```
"serverLink.cgi?127.0.0.1:6500:From Russia with Love"
```

5 Restart the Apache server using the following command:

```
C:\Program Files\Apache Group\APACHE\apache -k restart
```

Checking the Browser-Interface Installation

1 Test that the web server is correctly configured by connecting to the virtual directory. For example, for a virtual directory called 'vcm' on the local system, use the following URL:

```
http://127.0.0.1/vcm/default.htm
```

2 Test that the browser interface is working by connecting to it. Click on the appropriate 'VC/m Client' button on the web page displayed above. This will launch the VC/m browser interface in a new window.

3 If you want to be able to launch the VC/m browser interface direct from the desktop, see the information in the section titled "Interfaces to the Browser Interface"on page 119 about using .hta files.

Troubleshooting

- If you are using Apache as your web server, you may need to take extra steps to ensure it serves VC/m's .htc files with the correct MIME type. See Microsoft article http://support.microsoft.com/kb/306231 for further details. The problem has been observed with older Apache installations. If the .htc files are not served correctly to Internet Explorer the selector field on the VC/m toolbar fails to work. The following script error also occurs when defining a transfer route:

```
Can't move focus to the control because it is invisible, not enabled, or
of a type that does not accept the focus
```

WebLink Guide

If the VC/m interface is not launched and you are using WebLink, try looking at the WebLink System Management page. For a virtual directory called 'vcm' on the local system, use the following URL:

```
http://127.0.0.1/vcm/mgwms32.dll
```

The WebLink configuration is saved in a file called mgw.ini in the WINNT directory (or equivalent). If the configuration does not save, you need to alter the permissions on the account IIS used for anonymous access to allow it to write to this file.

The page default.htm specifies the following string to launch VC/m:

```
http://127.0.0.1/vcm/mgwms32.dll?MGWLPN=vcm&MGWAPP=vcm
```

MGWLPN is the name that was specified under 'Configure Server' on the WebLink System Management page.

The value of MGWAPP is always vcm. This is specified by the global ^MGWAPP in the VC/m namespace.

ServerLink Guide

serverLink.cgi is a perl script. When it is transferred using FTP, it must be transferred as ASCII.

The .htaccess includes a RewriteRule which translates all requests to mgwms32.dll to the serverLink.cgi.

The second parameter of start^serverLink is the serverLink logging level. If serverLink is not working, try setting this to 99 and look at what appears in ^serverLink("log"

The page default.htm specifies the following string to launch VC/m:

```
http://127.0.0.1/vcm/mgwms32.dll?MGWLPN=vcm&MGWAPP=vcm
```

MGWLPN is the namespace name.

4.5 Backing Up your Installation

Once in use, the VC/m libraries and configuration management database contain information that is of significant value. As a minimum, the following globals should be backed up:

`^%vc*`	in the manager UCI, namespace or directory
`^vcli`	in the designated VC/m library locations

Note: The manager globals and the library globals should always be backed up and restored together.

In practice it is simpler to back up the operating system files which contain the database.

4.6 Additional VC/m Installations

VC/live

A VC/live system is installed in the same way as a VC/m server. A sub-set of the configuration options are applicable.

Appendix III: Installation and Configuration Checklists has a checklist which should be used to ensure that all of the necessary steps are completed when installing the VC/live software for the first time.

More detailed information can be found on p. 137.

Task Server

M Environment

When the task server is run on a machine which is separate from the main VC/m server, there are some differences in the configuration of the M environment.

1 It is essential that the globals ^%vc* are mapped to the main VC/m system and that the global ^%gjtask is local, since different data is stored in each system.

2 Additionally, the task server job (%vczn) must be running in the background. This should be set to start automatically when M starts up.

4.7 Basic M Guide

Caché

M Installation and Back Up

The Caché configuration is stored in a file called CACHE.CPF in the installation directory.

The database is stored in files called CACHE.DAT. The [database] section of the CACHE.CPF lists all the directories where these are located.

Obtaining an M Prompt on a Windows System

When Caché is running, a blue cube is visible in the system tray in the bottom, right-hand corner of the screen.

Right-click on this cube. Look at the name of the 'Preferred Server', and if it does not correspond to your VC/m server, select the correct one from the list.

Once the preferred server is correct, select 'Terminal' from the same right-click menu.

Restoring Routines

To restore routines, enter the following at the M prompt:

```
do ^%RI
```

If Caché reports that this file is not a %RO output file, say 'Yes' to override and use the file with %RI. Choose Caché mode for the restore.

Syntax errors will be reported, but these can safely be ignored.

DSM

M Installation and Back Up

By default, M is installed in a directory called [DSM]. The database is stored in files with a .GLS extension.

Obtaining an M Prompt

To obtain an M Prompt, type the following at the DCL command prompt:

```
DSM
```

For operations which require full access privileges, use the /MANAGER switch:

```
DSM/MANAGER
```

Restoring Routines

To restore routines, enter the following at the M prompt:

```
do ^%RR
```

Syntax errors will be reported, but these can safely be ignored.

GT.M

Restoring Routines

To restore routines, enter the following at the M prompt:

```
do ^%RI
```

When prompted for the output directory, enter the name of the VC/m installation directory (e.g. (/usr/vcm/). On Unix platforms it is essential to include the trailing slash (/) otherwise the routines will be loaded into an incorrect location.

Once loaded, the routines *must* be compiled. (Do not rely on GT.M's Auto-Link functionality as this does not handle source files with names greater than 8 characters correctly.) Use the following command to compile all routines in the VC/m installation directory:

```
$mumps *.m
```

Syntax errors will be reported, but these can safely be ignored.

MSM

Restoring Routines

To restore routines, enter the following at the M prompt:

```
do ^%RR
```

Syntax errors will be reported, but these can safely be ignored.

M21

Restoring Routines

The routines should be restored into a manager, i.e. [MGR,xxx], UCI.

To restore routines, enter the following at the M prompt:

```
do ^%RR
```

Syntax errors will be reported, but these can safely be ignored.

Global Protection

VC/m uses and updates globals stored in the manager UCI. You should ensure that the protection codes for these globals (^%vc*) are such that they can be read, written and deleted from all UCIs where VC/m will be used.

5 System Manager's Guide

5.1 Configuration Procedure

Planning

Before you configure and use VC/m, it is essential that you plan how you intend to use it and design the configuration management procedures which you want to use.

Appendix I: Installation and Configuration Planning gives guidance on the questions which need to be addressed before proceeding.

Appendix II: Charts for Configuration Planning has charts which can be used to help in your configuration planning.

It is also recommended that a diagram is drawn showing all the logical locations which will be controlled by VC/m. Lines should then be drawn between each location to represent the transfer routes to be used. This will also enable you to consider what dependencies should be set up between locations and any other controls which may be required.

These charts and diagrams can then be used as a basis for both setting up and maintaining your VC/m system.

Configuration

Appendix III: Installation and Configuration Checklists has 2 checklists which should be used to ensure that all of the necessary steps are completed when configuring the VC/m software for the first time.

More detail on some of these steps is given in the following sections.

XML Definitions

Many of the main VC/m configuration tables can be defined and managed using XML formatted configuration files. Elements of the configuration can also be abstracted using XSL, making it easy to define templates. This is useful where there are many repetitive elements or patterns in the definitions, such as a large number of sandboxes or regularly added library release locations.

XML configuration files are easy to set up even for newcomers to XML. However, the mechanism is not recommended for initial prototyping setup of VC/m when concepts are being learned and workflow strategies experimented with. Furthermore, use of XML files requires a commitment to maintaining this approach. Direct changes to the configuration tables made through the browser or character interface will be overwritten if configuration globals are subsequently reloaded from the XML.

The following configuration tables can be maintained using the XML configuration file:

- Users
- Physical locations
- Logical locations
- Location classes
- Transfer routes
- Systems

In order to run the batch file that transforms XML definitions for VC/m use, Java 1.4 or later must be installed. Java 1.4 includes the xalan XSLT processor used by the mechanism. Later Java versions require additional jars to be installed. Read the comments in the batch file for further instructions.

The files for the XML schema are in the Configuration subdirectory of the VC/m installation directory. vcmConfigurationExample.xml contains examples of how to define physical locations, logical locations and routes.

To start using the configuration files:

1 Copy vcmConfigurationExample.xml to vcmConfiguration.xml.

To add or modify the definitions:

1 Edit vcmConfiguration.xml to suit your purposes.

Note: Physical addresses for M type locations should always be entered in upper case.

2 Run vcmConfiguration.bat. This will read vcmConfiguration.xml and create a global save file named vcmConfiguration.g. Check for errors and check that the .g file is created.

3 Load the global save file into VC/m using your standard global restore utility.

4 If you need to validate the entries, open them using the Master Files options and move down through the screens. Any errors will be reported.

The import of the .g file does not perform any deletions. To delete a definition:

1 Use the Setup subtree of the browser interface or the Master Files options of the Manager menu in the terminal interface to delete the item.

2 Delete the definition from vcmConfiguration.xml so that it will not be recreated when the batch file is next run.

5.2 Recommendations for Best Practice

User IDs and Access Codes

We recommend the use of 'NOBODY' as the standard access codes for locations or transfer routes which are not available to anyone. This can be used, for example, to prevent a location from being displayed in the browser interface.

Similarly, we recommend the use 'EVERYBODY' as the standard access code for locations or transfer routes which are available to everyone. Every user should have this code in their access list. The code can then be used, for example for auto-transfer routes to ensure that they are always triggered.

We also recommend the use of 'SUPERUSER' as an access code for unusual, potentially risky routes. No user should have this as part of their access list. Instead, there should be a separate username of 'SUPERUSER' which has this access code. This prevents these routes from being used accidentally.

In general, it is best to use access codes which describe a user role.

Locations and Location Classes

We recommend the use of upper camel notation for location names, eg WaitingForTesting.

5.3 Component Types

Each component type comprises an entry in the component type table and a driver routine. Both parts are essential to enable VC/m to handle components of this type.

Not all component types are supported in all physical locations. For example, it would not be sensible to permit an executable file to be placed in an M routine directory.

The table entry defines the code used for the component type, and the sequence in which different components will be transferred.

The driver routine performs the physical manipulation of the component for VC/m. Each time VC/m needs to copy, delete or otherwise access a component, it calls the appropriate function in the component driver routine.

Standard Component Drivers

The following generic component types are available:

Component Type	Component Description	Purpose
BIN	Binary Files	To manipulate binary files
T	Text Files	To manipulate text files

The following component types are available for M systems:

Component Type	Component Description	Purpose
G	M globals	To manipulate M global sub-trees
GZ	Binary globals	To manipulate M global sub-trees which contain control characters
R	M routines	To manipulate M routines

Note: Types G and GZ are used for any part of a global, anything from a single node to a whole global.

The following component types are available for DSM systems:

Component Type	Component Description	Purpose
P	Pcode routines	To manipulate DSM p-code

The following component types are available for Caché and Ensemble systems:

Component Type	Component Description	Purpose
BAS	Basic routines	To manipulate Caché Basic routines (Caché 5 and later)
CDL	Caché object class definitions	To manipulate Caché object class components stored in an M directory or a text file (Caché 4)
CLS	Caché object class definitions	To manipulate Caché object class components using XML format (Caché 5 and later)
CMB	M/SQL menu objects	To manipulate M/SQL menu objects stored in ^mobject using the 'old' M/SQL import and export functions
CMQ	M/SQL queries	To manipulate M/SQL queries stored in ^mql using the 'old' M/SQL import and export functions
CMR	M/SQL reports	To manipulate M/SQL reports stored in ^mreport using the 'old' M/SQL import and export functions
CMT	M/SQL tables	To manipulate M/SQL tables stored in ^mdd using the 'old' M/SQL import and export functions
CMV	M/SQL views	To manipulate M/SQL views stored in ^mdd using the 'old' M/SQL import and export functions
CMW	M/SQL forms	To manipulate M/SQL forms stored in ^mform using the 'old' M/SQL import and export functions
CRT	M/SQL menus	To manipulate M/SQL menus stored in ^mmenu using the 'old' M/SQL import and export functions
CSP	Caché server pages	To manipulate Caché server pages (Caché 5 and later)
CSR	Caché server rules	To manipulate Caché server rules (Caché 5 and later)
EWD	Enterprise Web Developer run-time components	To manipulate Enterprise Web Developer (a.k.a. eXtc Web Developer) components
HL7	Ensemble HL7 schemata	To manipulate Ensemble HL7 schemata (Ensemble only)
INC	Caché .INC files	To manipulate Caché .INC files stored in ^ROUTINE(0,"INC",1) or ^rINC
INT	INT routines	To manipulate Caché .INT files
MAC	Caché and M/SQL .MAC files	To manipulate Caché and M/SQL .MAC files stored in ^ROUTINE(0,"MAC",1) or ^rMAC
MMB	M/SQL menu objects	To manipulate M/SQL menu objects stored in ^mobject using the 'new' M/SQL import and export functions
MMQ	M/SQL queries	To manipulate M/SQL queries stored in ^mql using the 'new' M/SQL import and export functions
MMR	M/SQL reports	To manipulate M/SQL reports stored in ^mreport using the 'new' M/SQL import and export functions
MMT	M/SQL tables	To manipulate M/SQL tables stored in ^mdd using the 'new' M/SQL import and export functions
MMV	M/SQL views	To manipulate M/SQL views stored in ^mdd using the 'new' M/SQL import and export functions
MMW	M/SQL forms	To manipulate M/SQL forms stored in ^mform using the 'new' M/SQL import and export functions
MRT	M/SQL menus	To manipulate M/SQL menus stored in ^mmenu using the 'new' M/SQL import and export functions
MVB	Caché MultiValue Basic programs	To manipulate Caché MultiValue Basic programs (Caché 2008.1 and later)

Component Type	Component Description	Purpose
MVI	Caché MultiValue intermediate routines	To manipulate Caché MultiValue intermediate routines (Caché 2008.1 and later)
PRJ	Projects	To manipulate Caché projects (Caché 5 and later)
RUL	Ensemble rule definitions	To manipulate Ensemble rule definitions (Ensemble only)
WLD	WebLink Developer run-time components	To manipulate WebLink developer run-time components

Component Type Table

The component types defined for VC/m are held as a global:

```
^%vcvc("CT",sequence)=component type
```

where *sequence* gives the sequential order in which VC/m transfers all the components of that type

component type holds the component type code

Note: When a group of objects is transferred, the transfer sequence applies to the components within each object and not to the sequence of objects within the group.

The transfer sequence of the following component types will be defined automatically on installation:

Component Type		Transfer Sequence
R or INT	M routines	1
G	M globals	2
GZ	Binary globals	3
BIN	Binary files	4
T	Text files	5

Others are added depending on the Caché version.

You must ensure that a value is specified for each of the component types which you wish to manage. If you do not, the integrity of your system may be compromised.

For example:

```
set ^%vcvc("CT",9)="P"
```

will cause VC/m to transfer P type components after the above component types.

In most circumstances, the transfer sequence of components is often not significant and so any sequence of numbers may be used, provided each one is unique.

After amending the ^%vcvc("CT" table you should DO ctxref^%vcins

Component Date /Time Stamps

When a binary or text component is transferred by VC/m, it is possible to preserve the modification date /time stamp which the operating system records for the file. This is essential if 'time stamp' is used as the dependency check. It is also useful if users wish to check whether two files are the same by looking at the operating system directory listings. This is set up as follows:

1 Install a Perl interpreter.

Note: On Unix platforms, this is normally pre-installed. A Perl interpreter for a Windows platform can be obtained at no cost from http://www.perl.org.

2 Set up the path and command to invoke the Perl interpreter. In the browser interface this is done under Setup, Properties, Environment. If this interface is not available, set the following global node directly, e.g.:

```
set ^%vcvc("perlCommand")="c:\program files\perl\perl.exe"
```

Comment Lines

When a routine component is physically copied to an M or F type location, VC/m adds comment lines. A routine component is one of type R, INT, WLD, INC, MAC or BAS.

The VC/m comment lines are regenerated when a developer opens a routine which is under VC/m control. Even if a developer deletes or modifies the lines and saves the amended routine, the next time it is opened in Studio, the information is correct.

When a component is transferred *from* an M or F type location, VC/m removes the comment lines.

When a routine is transferred to a location of any other type, the VC/m comments are removed.

The comment lines are generated by the comment line generator. The content of the lines can be changed by using a component driver callout.

More detailed information can be found in the VC/m Reference Manual.

Managing Routine Components

M routines can be managed as either component type R or INT. The functionality of these is the same.

The INT component type provides a more consistent name for the management of Caché INT components.

Managing Global Components

VC/m can be used to manage any part of a global, anything from a single node to a whole global.

Two component drivers are available for globals. Type G is the usual driver. Type GZ should be used for globals which contain control characters. The main difference in processing between the two is when writing to the F storage format. Components of type G are written out in an ASCII format text file. This is inappropriate for a global which contains control characters.

An error is given when a user tries to check in a global which contains control characters but is registered as type G. This behavior can be over-ridden by setting the following node:

```
set ^%vcct("G","param","allowControlChars")=1
```

Global components can be entered with or without a close bracket at the end of a subscript. In the VC/m database, the component name is stored without the "^" character but with the closing bracket, e.g. A("ABC").

Managing Text and Binary Files

Operating System files are either of type T or BIN.

Type T (text file) is used for files which contain text which can be read with a simple text editor. When using FTP, these would be transferred as ASCII.

Type BIN (binary file) is used for files which contain binary data. The data is not meaningful if the file is opened with a simple text editor. With FTP, these would be transferred in binary mode.

The browser interface gives more accurate information if VC/m knows whether a file extension is of type T or BIN. This is specified by setting the following global:

```
set ^%vcvc("fileTypes",component_type,extension)=""
```

For example:

```
set ^%vcvc("fileTypes","T","html")=""

set ^%vcvc("fileTypes","BIN","pdf")=""
```

Managing Caché MAC and INC Components

Caché MAC and INC components are managed in a very similar way to standard M routines.

For development, they should be placed in a physical location with a storage format of M. Whenever a MAC or INC component is transferred to an M-type location, it is automatically compiled.

Managing M/SQL Components

VC/m includes two sets of component drivers for M/SQL components. These correspond to the two types of import and export functions built into M/SQL.

The first set (whose names begin with M) uses the ^mxdd import and export functions to transfer the components. These externalize all the definitions.

The second set (whose names begin with C) uses the old M/SQL import and export functions to transfer the M/SQL components. These do a more physical copy of the M/SQL globals.

The main difference between the two is that the second set preserves the RowIds of the tables and forms. These drivers should be used when hard-coded references to RowIds have been used in the code. The first set is preferred in other cases, particularly if development is doing done in more than one place.

For each location where the M/SQL components can be run, configure the location definition so that these components have a storage format of "M" and a physical address of the namespace or directory where they should be placed.

If the first set of component drivers is used, the following global node must be set, depending on the version of FDBMS which is being used. For version 14.x or lower:

```
set ^%vcvc("msql_version")="F14"
```

For version 15 or higher:

```
set ^%vcvc("msql_version")="F15"
```

Managing Caché Object Class Definitions

VC/m includes two component drivers for Caché object class definitions. The CDL component driver should be used for Caché 4, and the CLS driver for Caché 5 onwards.

Note: The CDL component driver manipulates the System property of classes it controls in order to prevent unauthorized changes being made in Object Architect.

Each of the components must be set up with a type of "CDL" or "CLS". The component name is made up of the package name, followed by a dot, followed by the Caché class name, e.g. Package.Object.

For each location where the Caché classes can be run, configure the location definition so that all CDL or CLS components have a storage format of "M" and a physical address of the namespace where they should be placed.

In addition, for CDL components, a transfer call-out is required which will compile the components after they have been transferred. On screen 1 of the Properties option in VC/m Set-up, add the following to the Per-transfer, initialization:

```
init^%vc1xCDL
```

To run the compilation in the background rather than in the foreground, add the following to the Per-transfer, termination:

```
compile^%vc1xCDL
```

Note: For CDL components the compile is run as a background job and often takes some considerable time after VC/m has made the transfer.

Note: Classes which are generated from Caché Server Pages should be managed as CSP components, not as classes.

Note: In order to display the classes in Cache 4's Object Architect on the client PCs, set it to display System Classes.

Managing Caché Server Pages and Rules

VC/m has component drivers for Caché Server Pages and Rules which compile the pages as well as managing the source files.

When more than one CSP application is defined for a single namespace, VC/m cannot deduce which one the CSP and CSR components it is managing belong to, and WRITE $$appUrl^%vc1xCSP in the namespace returns null. In such a case, define which application to use by setting URL (with trailing "/") in ^vcvp("cspAppURL") in the namespace, e.g. ^vcvp("cspAppURL")="/csp/vcmdemo/samples/dev1/". Note, there is no %-prefix on this global name.

If you need to manage pages for more than one application per namespace, the CSP source files should be managed as text components (type “T”).

If the CSP driver is used, each location where the pages can be run should be configured so that these components are in a physical location with a storage format of ‘M’. The component name is a directory path and filename without the .csp extension. The path is always relative to the base directory where the CSP application is defined for that namespace.

If the T driver is used, the components should be placed in a physical location with a storage format of ‘text’. The component name is a directory path and filename with the extension .csp.

Creating Customized Component Drivers

The driver routine performs the physical manipulation of the component for VC/m. Each time VC/m needs to copy, delete or otherwise access a component, it calls the appropriate function in the component driver routine. Each function is simple and can be written directly in the M programming language.

A programmer can create a customized component driver to handle components in ways other than those provided by the standard VC/m component drivers. In order for VC/m to recognize the new driver, an entry must be set-up in the component driver table. A one to three-character uppercase code should be used to identify the new component type and a routine created whose name begins '%vc1x' and is followed by the component type code. For example, a new component type 'ABC' requires a component driver named '%vc1xABC'. A number of functions with standard line labels must be incorporated into the component driver.

Component type codes beginning with "Y" are available for customers to use for their own purposes. All other component type codes are reserved for drivers written and maintained by George James Software.

5.4 Physical and Logical Locations

Storage Formats

A physical location is defined as having one of the following storage formats:

Storage Format	Description	Notes
F	Packed sequential files	stores components in sequential files in the host operating system
FX	XML files	Stores components in text files written in Caché XML export format
L	VC/m library	stores any type of component in the global ^vcli
M	M routine and global directory	stores M routines, globals and other M items in an executable form
bin	Binary files	stores any files which contain control characters
text	Text files	stores files which can be viewed as text only
vss	Microsoft Visual SourceSafe database	stores files in the Microsoft Visual SourceSafe library

Note: A physical location only has one storage format. Therefore, if a logical location has both text and binary components which are stored in the same directory, two physical locations (i.e. with different names) must be set up which map to the same physical address.

Physical Addresses

The physical address may depend on the storage format, the host operating system and the M implementation on which VC/m is running.

F, text and bin Formats

For text, binary and sequential files, the physical address is the full path of a directory specification on the host operating system. When running in a networked DOS /Windows environment, you are advised to use the network path name. The following table gives some examples:

Operating System	Example
DOS /Windows	\\Server1\DriveD\Dev\Project4
Unix /Linux	/usr/dev/project4/
OpenVMS	DISK$DATA1:[DEV.PROJECT4]

L and M Formats

For VC/m library and M routine and globals formats, the physical address depends on the M implementation. The following table lists the possibilities:

M Implementation	Physical Address	Example
Caché	Namespace	USER
DSM	UCI,VOL	MGR,DEV
GT.M	(see below)	
MSM	UCI,VOL	MGR,DEV

Note: Where a UCI name is required, the volume set is optional unless multiple volume sets are in use and access is required across volume sets.

L Format in GT.M

The physical address of an L format location in GT.M is in the form:

```
vcmInstallationDirectory,globalDirectoryFile
```

where *vcmInstallationDirectory* is the directory where the VC/m routines reside

globalDirectoryFile is the name of the .gld file which maps to the database which has been created for the VC/m library

For example:

```
/usr/vcm,/usr/vcm/library/mumps.gld
```

M Format in GT.M

The physical address of an M format location in GT.M is in the form:

```
routineDirectory,globalDirectoryFile[,objectDirectory]
```

where	*routineDirectory*	is the directory where VC/m reads and writes source code for this location
	globalDirectoryFile	is the name of the .gld file where VC/m reads and writes global data for this location if VC/m is being used to manage globals
	objectDirectory	is the location where object code is written when VC/m writes a routine to this location

For example:

```
/usr/joe/work,/usr/joe/work/mumps.gld,/usr/joe/object
```

If *objectDirectory* is not specified, VC/m will not compile routines when they are written to this location.

Remote Node Specification

This format is used for an M location on a remote node. It is also used as part of the remote protocol for text and bin formats on a remote node. It can only be used when a task server is running.

The physical address is a comma-delimited list which specifies how to access the M system on the remote machine. The format depends on the M implementation.

For a Caché to Caché system it is specified as follows:

Piece Number	Name	Usage
1	directory	the directory path of the Caché database belonging to the namespace (see piece 3) in which the task will be performed
2	system	the node name (Caché networking) of the Caché server where that database is mounted
3	namespace	the name of the Caché namespace in which the task will be performed, i.e. one which uses the database specified under 'directory' above
4	manager directory	the directory path of the database on 'system' in which the task server runs

For a DSM to DSM system it is specified as follows:

Piece Number	Name	Usage
1	Uci	the UCI which the job will be run in
2	Vol	the volume set which the job will be run in
3	Guci	
4	Gvol	
5	manager uci	the UCI which the task server will run in
6	manager vol	the volume set which the task server will run in

For a MSM to MSM system it is specified as follows:

Piece Number	Name	Usage
1	Uci	the UCI which the job will be run in
2	Vol	the volume group which the job will be run in
3	manager uci	the UCI which the task server will run in
4	manager vol	the volume group which the task server will run in

Remote Protocol

This format is used for a text or bin format on a remote node. It can only be used when a task server is running.

A physical address which uses the remote protocol is made up of three parts, joined as follows:

protocol://remote_node_specification:local_physical_address

The protocol is 'remote' or 'REMOTE'.

The remote_node_specification is a comma-delimited list which specifies how to access the M system on the remote machine. The format is described above, under ‘Remote Node Specification’. Alternatively, it can be entered as @physical_location_code, where physical_location_code is a defined physical location with type M that already has the necessary remote node specification as its address. This mechanism is particularly useful in situations where remote_node_specification itself needs to contain a colon (:) character, for example as part of Windows directory paths for Caché databases.

The local_physical_address is the physical address of the location, given in the format which you would use for a physical address on a local machine. This is used to specify the physical address when accessed from the remote machine.

vss Format

For Visual SourceSafe library formats, the physical address is $/ followed by the project name, e.g. $/vcmLibrary. If there is no project, it is simply $/.

Dependency Checks

When a location is used as a dependency on a transfer route, VC/m may need to compare each component at this location with the same component at the 'from location'. The dependency check determines the mechanism which is used:

Dependency Check	Behavior
Timestamp	Compares the external date /time stamps
Compare	Compares each byte
Internal	Compares VC/m's own internal date /time stamps

The 'Compare' check will be the most accurate but it will also be much slower than the others.

The 'Timestamp' check will only work for components which have an external date /time stamp. For example, the operating system usually records the date and time when a file is modified, but an M implementation does not record the date and time when a global is modified.

The 'Internal' check is usually the preferred option. However, this should not be used if an editor is being used which does not have an interface to VC/m. Without an editor interface, the VC/m date /time stamp will not be updated when the component is changed.

Note: For the 'Internal' check, the components do not physically have to exist at the location.

Relaxing Control: Uncontrolled Locations

An uncontrolled location is a physical location whose contents VC/m does not expect to have full knowledge of, nor exclusive control of.

Uncontrolled locations can be referenced within VC/m in exactly the same way as controlled locations, namely as targets of component masks on logical locations. The difference is that VC/m does not perform any checks on the physical contents of an uncontrolled location before making a transfer. In particular, if the transfer would overwrite components at that location, this is allowed to happen. By contrast, if VC/m is transferring to a controlled location, it expects that every component at the location has been registered. If the transfer would overwrite any components at that location which have not been registered (i.e. components that do not belong to an object), the transfer is not allowed. Similarly, if the transfer of an object makes a component redundant, VC/m will physically delete the component from a controlled location, but not from an uncontrolled location.

When an object is transferred to a location whose component masks map to controlled physical locations it is normal to make the route activate the object at the destination location. VC/m has placed a component at a controlled physical location, so should record which object the component belongs to there. If it does not, and the location is single-version, a subsequent transfer involving the same component will not be possible.

A transfer to a location mapping to uncontrolled physical locations may or may not activate the object at the destination. If it does, VC/m will apply the normal displacement constraints on subsequent transfers to this location. For instance, if the location is single-version and the subsequent transfer is of a predecessor version or a version on a different branch then the transfer will be disallowed. Likewise, displacement of a master version will be prevented.

> *Note: It is not meaningful to set up uncontrolled library locations. If objects are not marked as active when they are transferred to the library, it will not be easy to extract them from the library.*

> *Note: When components are removed from an object, VC/m will not be able to delete them from an uncontrolled location automatically.*

Library Locations

A VC/m system should have at least one library which is used to store the master copies of all object versions. This is a set of one or more multi-version physical locations which each have a storage format of either L or vss.

In order to access all the master copies using the VC/m system, there should be a multi-version logical library location which maps to the library set of physical locations. All component types should map to one of the physical library locations. All the systems which are in use should belong at this location.

When objects are needed for development, they are checked out from this multi-version location. When development is completed, the objects are checked into it.

Single-version library locations can also be defined. A single-version location can be used, for example, to keep a record of a particular version of the application or to hold the most recent version. These kind of locations are particularly useful because they do not have any significant overhead on disk space. A single-version library location should be defined with all the same component mappings as the multi-version library location. Unless it has a more specialized purpose, all the systems should also belong there.

Sequential File Locations

A sequential file location is one in which at least some of the components are stored in format ‘F’. It is usually used either as a temporary location or as a means of releasing software.

The names of the files can be defined using the Transfer Files option in VC/m Setup.

If the location is for temporary files, it should be configured with the physical locations as uncontrolled. The commonest way to configure the location is with T type components in a physical location with storage format ‘text’; BIN type components in a physical location with storage format ‘bin’ and all other components in a physical location with storage format ‘F’. Objects are not usually made active in this kind of location, thus allowing earlier versions of an object to be transferred to the location after later versions.

If the location is to be used for release to VC/live, all components must be mapped to a physical location with storage format ‘F’. If they are to be used for the site’s own release procedures, the components types should probably be mapped as described above for temporary locations. Objects for release should usually be activated in the location.

Locations without Physical Addresses

It is possible to define a location which does not map to any physical locations. This can be thought of as a kind of virtual location. When an object version is transferred to it, no components will physically be moved. However, the statuses of the object version will be updated in accordance with the transfer route. This kind of location cannot, of course, be used as the source location for a transfer route if the components must physically be moved to the destination location.

This kind of location is suitable for transfers which indicate the next step in the development cycle, without the need for the object versions to exist in a location. An example is a transfer from a ‘ready for testing’ step to ‘accepted’.

Generic Locations

Generic locations are standardized locations which can be used to simplify the setting up of transfer routes.

In order to use generic locations, they must first be defined in the location file. They should be defined as multi-version locations which do not map to any physical locations. The standard names must be used.

%ANY is a generic location which functions as a wildcard for all existing locations. It can be used in the 'from location' and the 'to location' field of a transfer route.

%FROM is a generic location which represents the 'from location' of a transfer route. Similarly, %TO is a generic location which represents the 'to location' of the transfer route. They can be used in the dependent locations, new active locations, new master location and message fields of a transfer route.

%FROM and %TO can be used in the definition of any transfer route. However, they are particularly useful when %ANY is used in either the 'from location' or the 'to location' field.

Copying Another Logical Location

The Copy option under Logical location maintenance replicates the contents of a logical location. If a physical location is copied, this option can be used to create a new location that represents the contents of that physical location. The procedure is as follows:

1 Physically make the copy of the area.

2 Set up a new physical location with the address of the new directory.

3 Set up a new logical location that maps onto the new physical location.

4 Select the copy option and copy from the logical location that mapped onto the original physical location to the logical location just created.

VC/m will populate the new location with the same version information as was at the old location. Note that it does not actually physically copy any components when you do this.

The NEW Location

When a new object is created by using the Object registration file in the VC/m character-based interface, it is initially placed in a nominal location of NEW with a version of 0.

In order to be able to set up transfer routes for newly-created objects, you first need to set up this location. The name must be in upper-case. It should be a single-version location, with all the systems which exist. It can be set up as a location without physical addresses.

You will also need to define at least one check-out transfer route from NEW so that newly-created objects can be transferred.

5.5 Transfer Routes

Function Codes

The **function code** identifies the type of transfer for which a transfer route is used. A number of standard codes are defined in VC/m, and installation-specific ones can also be used.

The following standard function codes are defined in VC/m:

Code	Used by Function	Notes
XFER	Transfer Bulk transfer Install Archive and Purge	Standard transfer functionality.
OUT	Check-out	Standard transfer functionality. By convention, this is used when objects are checked out from the library and the version number is incremented.
IN	Check-in	Standard transfer functionality. By convention, this is used when objects are checked in to the library.
STATUS		Standard transfer functionality. By convention, this is used when setting a status date on the object versions in the transfer. The transfer is often only a logical transfer, rather than physically moving or copying the objects.
AUTOT		An automatic standard transfer triggered by the dependency conditions being satisfied.
AUTOS		For backward compatibility only.

When installation-specific function codes are used, they are created when a transfer route is set up. In order to use a transfer route with an installation-specific function code, a menu option must be added using the Menu maintenance. This also needs to be added if the STATUS function code is to be used.

More detailed information can be found in the VC/m Reference Manual.

Using different function codes means that it is possible to define more than one transfer route between any two locations. These routes can have different characteristics, e.g. different access codes, so they can be used for different purposes. It can also reduce the number of transfers of a particular type, especially XFER. This will increase the likelihood of the locations defaulting correctly when the standard transfer function is used. It also helps users to distinguish between routes intended for different purposes. However, the character-based Bulk transfer option can only be used for routes with a code of XFER.

Note: CHMAINT must not be used as a function code since this has a pre-defined meaning within VC/m.

Dependent Locations

A dependent location is a condition that an object version can only be transferred if the version is already active at the dependent location. The object version which is to be transferred from the 'from location', not the resulting version at the 'to location', is the one which is used to make this comparison.

Dependent locations provide a mechanism whereby an object has to meet certain conditions before it can proceed to the next step in the development or release process. For example, a dependency can be set up such that an object version cannot be checked back into the library unless it is active at location TESTED. The quality manager may be the only user authorized to transfer objects from DEV to TESTED, thus ensuring that untested software cannot be placed in the library. The dependency is only applicable if the object version belongs to a system at the dependent location.

Location classes can also be used instead of locations in the dependent locations field for a transfer route.

The dependency can be defined so that it is satisfied if the version itself is at the specified location (absolute dependency) or so that it is satisfied if that version or a later one is at the specified location (relative dependency).

For example, if location A contained version 1 of an object and location B contained version 2, an attempt to transfer version 1 from A to another location using a route with a relative dependency on location B would succeed because the version at B is a later one than the version being transferred. (It can be assumed that version 1 was at some time at location B, but was displaced by version 2) If there was an absolute dependency on location B, the dependency would not be satisfied, because version 1 is not the version at location B.

By default, dependencies are relative. To define an absolute dependency, suffix the location code with # in the dependent locations list for the transfer route. In the example below, there are relative dependencies on DEVA and DEVB and an absolute dependency on LIB.

```
Dependent locations  [DEVA,DEVB,LIB#           ]
```

Negative dependencies can also be used, so that, for example, the transfer route could be set so that a version could only be archived if it were not in the live or the development area. A negative dependency is specified by prefixing the location code or location class with an ' character. A negative dependency is satisfied whenever the object version does not have a status of active at the location, i.e. it may have an error status or no status.

There is also a dependency called EMPTY. This dependency is satisfied if the object version is empty (i.e. it contains no components). The negation of the EMPTY dependency can also be used.

When the latest object version at the dependent location has the same version number as the object version to be transferred from the 'from location', VC/m compares each component. The dependency is only satisfied if each component at the dependent location is the same as or a later version than the one at the 'from location'. This mechanism is used to validate that a set of locations are kept in-line with each other. The precise kind of comparison which VC/m uses depends on the physical location. It is defined in Physical Location maintenance.

Installation-specific Dependencies

An installation-specific dependency is in the form of an extrinsic function in the M programming language.

The function must be accessible from every area where VC/m can be run. The simplest way to achieve this is to use a routine which starts with the percent character (%).

Usage:

```
set ok=$$label^routine(fcode,lcode1,lcode2,obv1,.err)
```

Inputs:

fcode	=	Function code for the transfer
lcode1	=	From location code
lcode2	=	To location code
obv1	=	Object version at from location

Output:

$$	=	1\error message = Change request is not valid 0 = Change request is valid
err	=	Error message if dependency is not satisfied

Some pre-defined functions to use or learn from are available in %vc1xvf.

New Active Locations

In general, it is desirable that an object version should be active at all the locations at which it exists. The ‘to location’ should therefore be included in the list of new active locations.

If the ‘to location’ is not included in the list of new active locations, the components will be transferred to the location, so they exist there, but they will no longer be registered to VC/m. VC/m will have no record of the object version at that location. This may be useful, for example, for a scratch location.

Other locations may be made active when a transfer is made. This is generally used for locations where the object version will exist after the transfer. For example, when an object is checked into the library, it may be made active in several single-version library locations as well as the multi-version library location. In this case the object version will exist in the new active location as a result of this transfer. Alternatively, the object version might be made active in a location where it exists as the result of a previous transfer.

An object version may also be made active in a location where it does not physically exist. It cannot then be used as the source of a transfer, but it can be used to satisfy dependency checks for other transfers.

An object version may also be de-activated at a location. This does not mean that components will be deleted, but the object version will no longer be recorded in VC/m’s database at that location. This is generally used when the object version is still active at another location which maps to the same physical locations, or when the components were never physically transferred to the location.

New Master Location

Certain transfer routes move the master copy of the object version. In general, this should only be done when the route is a check-out or a check-in, i.e. to move the object version in and out of the library for development work. It may occasionally be used to move the master copy within the development cycle.

When a transfer route has a new master location, this location should always be included in the list of new active locations.

The VC/m system does not check the location of the master version before a transfer is made. This could mean, for example, that a developer can check in from their sandbox even when they do not have the master copy. On an installation where this situation can arise, it should be guarded against by the use of a transfer dependency of *not* the 'to location' or the use of an installation-specific dependency.

When transfer routes are designed, the location of the master copy should always be kept in mind. Some of the possible pitfalls are as follows:

- a route which moves the master copy but does not make it active at its new location
- a route which de-activates the object version at the location where the master copy is
- a route which creates a new version but does not move the master copy, leaving no master copy of the new version
- a route of type 'Move' which moves the object version from its master location or from another location which maps to the same physical locations without moving the master version

All of these situations are undesirable for the integrity of the VC/m database. It is the responsibility of the person setting up the transfer routes to ensure that these situations cannot arise.

When the master copy is moved by a transfer which is a Copy, unless the new active locations field explicitly defines the new status of the 'from location', the object version will still be active at the 'from location'.

Create New Version

When new development work is being done, a new version should be created. When the Create new version? prompt is set to Y, the version number of the object version is incremented, so that the version at the ‘to location’ is a successor of the version at the ‘from location’.

Generally, the version number should only be incremented on a check-out route. It is possible to configure VC/m to increment the version number on check-in instead, but this is not advised.

When the version number is incremented, a new master location must be included as part of the transfer route. This master location should be one of the locations at which the new version will be active. If there is no new master location, an object version will exist with no master copy.

Copy Only if Changed

This attribute means that the components which belong to an object will only be transferred if they have been changed, i.e. have different date /time stamps in each location. If the objects comprise many components, this can significantly improve the performance of object transfers. It is particularly useful if the transfers are being performed via a slow modem line.

In order to be able to make use of this facility, it is essential that components are only edited by a routine editor (or maintenance object) that has been interfaced with VC/m. If any routine editors or component maintenance functions do not have a VC/m interface, this option *must not* be used.

If a transfer is to a remote location and VC/m is used to install the transfer files, the transfer route on the originating system and the install transfer route on the destination system should be set up in the same way. If the originating system creates transfer files that only contain changed components, while the destination system expects the transfer file to contain all components, an attempt to install the transfer file on the destination system will result in a fatal error.

If the transfer files are installed using a routine restore facility at the destination, rather than a VC/m install facility, the routine restore will just restore the components which are put in the transfer file on the originating system.

In order to ensure system integrity, some VC/m functions ignore this attribute and copy all components, whether they have changed or not. The following table summarizes the occasions when all components are copied:

Occasion	Notes
Auto Status Change	Auto Status changes do not perform physical transfers. This attribute is therefore irrelevant.
Archive and Purge	Transfer to archive files will always copy all components. Otherwise, if versions have been skipped, restoring objects from the archive could be difficult.
Cancel	When Cancel reinstates a previous version, all components are copied to ensure maximum system integrity.
Diff	When two object versions are compared, all components of each version are compared even if their date /time stamps are the same.
Hot Backup	If a hot backup is used with the Install function, a complete copy of each object will be made to ensure that a full recovery can be made if necessary.
Type G Components	Because there is no standard method of editing global sub-trees, and therefore no simple way of determining whether they have changed from one version to another, they will always be copied as part of a transfer. Note that custom component types will only be copied if they have changed, and so their maintenance tools must be properly interfaced to VC/m.
Multi-Version Locations	All components will be transferred to a multi-version location (for example, the library), as most drivers for these types of location will perform component comparisons and will only store discrete copies or deltas for components that have changed from one version to the next.

Audit Trail Messages

When a transfer is performed, a message is written to the audit trail using the 'Message' field from the transfer route maintenance screen.

A number of variables which will be substituted when the message is written can be used. These are as follows:

Variable	Translation
%Action	'Added' or 'Re-transferred'
%Ancestor	the object version which the new object is created from. This will only differ from %ObjVer on a route which increments the version number.
%ByChReq	if the transfer is of a change request rather than a single object, this holds the change request code
%ChReqs	a list of all change request codes for this object
%FROM	the 'from' location of the transfer
%Function	the function code of the transfer, e.g. OUT or XFER
%ObjVer	the object version which results at the end of the transfer
%TO	the 'to' location of the transfer
%Type	'Moved' or 'Copied'

These can be combined in various ways, e.g.:

```
%Type from %FROM to %TO (%ByChReq)

Checked-out from %Ancestor to %TO (%ChReqs)

%Action (%Function) to %TO
```

Moving from a Library Location (De-activating)

When a transfer route with a transfer of type Move is set up, the transfer will physically delete the components of the object from the source location.

Since the library is the main database for the master copies of all object versions, it is very unlikely that you actually wish to delete the components from here.

When there is a single-version library location, there may be situations when you wish to, conceptually, move the contents from this location. To do this, instead of setting up a Move, set up a Copy route, which de-activates the version in the single-version location. To do this, enter the location name, prefixed with a minus sign, in the new active locations field.

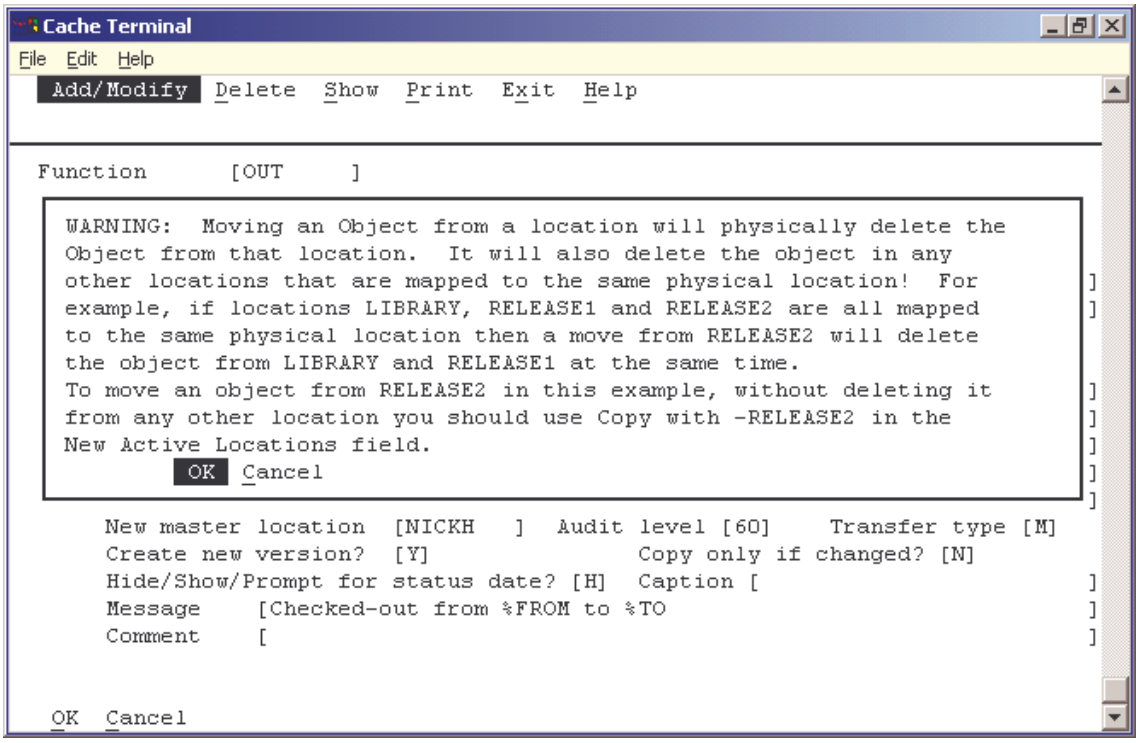


Figure 5.1 Moving from the library

Automatic Transfers: AUTOT Function Code

An automatic transfer is a transfer which is triggered when its dependency conditions are satisfied. It is made automatically by the VC/m system, rather than manually by a user using one of the transfer functions. It is defined by the use of the reserved function code AUTOT. Each time VC/m transfers an object version, it checks through its list of automatic transfers and makes any ones whose conditions are satisfied.

An automatic transfer depends on the access levels of the user who made the transfer which it is triggered by. Its access levels should therefore be set so that all users have access to it.

Automatic transfers are useful when several transfers have to be made at the same time. They can be used, for example, for updating all the developers' sandboxes when a change is checked in to the library. In this case, the automatic transfers use the 'to location' of the original transfer as their 'from location'. They can also, for example, have different function codes from the original transfer route.

The most important consideration when setting up an automatic transfer is the dependent locations. These must be viewed from two angles. Firstly, under what conditions should the transfer be made? As usual, the dependencies may be that the object version is active at that location, or that it is not active. Secondly, what condition will stop the transfer from being repeated ad infinitum? If there is no such condition, VC/m will keep re-transferring the object version, only failing after the hundredth transfer. Generally this can be addressed by including a dependent location of 'not' the 'to location'.

5.6 Model Transfer Routes and Configurations

A number of model transfers routes and configurations are available to help you. Please contact GJS support for information.

5.7 The Audit Trail

The Audit Trail is the essential reference to how a VC/m system is being used. It contains a permanent record of the activity of the system. It records the following information:

- Transfer of an object
- Transfer of a change request
- Creation (and deletion using the Programmer option) of an object
- Cancellation of an object version
- Merging of 2 object versions
- Addition and deletion of a component from an object
- Addition and deletion of an object from a change request
- Creation of a change request
- Copying or deletion of version information from a location
- Certain compilation errors

Each audit trail entry has an associated message category. This can be used to control the display of the audit trail.

For transfers, the message category and the text of the message are defined as part of the transfer route.

For other events, the message level is defined on the Audit trail screen of the Properties in Setup. The message text is pre-defined.

The display properties for each message category are defined on the Message Categories screen in Setup.

5.8 Objects

Variant Codes

By default, any letters, numbers and punctuation characters, except for "." (dot) and "/" (forward slash) and "*" (asterisk) can be used for the variant code.

The default code which will be given to a new object can be defined on screen 6 of the Properties option in VC/m Set-up. If this is left unchanged, "1" is used.

It is possible to set validation so that variant codes are only allowed if they match an existing system code. This is done by setting the following global node:

```
set ^%vcvc("VariantIsSystem")=1
```

5.9 Change Requests

Change Request Callouts

By default, any value can be entered as a change request code. There is no validation on this item and new change requests can be created at will. Installation-specific validation can be attached to the change request so that it can be validated against an external system. This validation is written in the M programming language.

Installation-specific conditions can also be added to a change in the change request status.

As well as validation, installation-specific code can be called at the time when a change request is saved. This can be used, for example, to update an external system.

To enable a change request callout, enter the line label of the M code to be invoked on screen 3 of the Properties option in VC/m Set-up.

More detailed information can be found in the VC/m Reference Manual.

Change Request Selection Callouts

These callouts are used by the change request selection dialog in the browser interface.

By default every change request is displayed and can be selected. The user can specify selection criteria which will restrict the change requests which are displayed, making it easier to find the change request which they require. These selection criteria are known as filters.

Installation-specific code can be added which will provide default values for the selection filters. This code is written in the M programming language. The user can still manually specify their own filters.

To enable default values for the selection filters, set the following global node:

```
set ^%vcvc("changeRequestSelectDefaults")="label^routine"
```

More detailed information can be found in the VC/m Reference Manual.

It is also possible to override the behaviour of the filters. This means, for example, that the user can be prevented from seeing and selecting certain change requests. The default filtering rules are provided by the code at filter^%vc029. Any installation-specific code will replace this functionality. The code checks each change request and includes or excludes it from the list which will be displayed.

To enable installation-specific filtering rules, set the following global node:

```
set ^%vcvc("changeRequestSelectFilter")="label^routine"
```

More detailed information can be found in the VC/m Reference Manual.

5.10 Log-in and Access Controls

Access Codes

Access codes are used to control which users are allowed to use a transfer route. They can also optionally be used to control access to a menu or menu option and to determine which locations are displayed on the browser interface.

An access code can be a role which is common to several users, or it can be a code which is unique to a particular user.

Each access code is a string of uppercase and /or numeric characters. If a code is entered in lowercase, it will automatically be converted to uppercase. There is no defined list of access codes. They are defined simply by being used, so care must be taken that the typing is accurate.

Each transfer route has an associated access code which a user must have in order to use it. If access security is used for menus or locations, then these items will also have an access code set up.

If there is more than one access code, these must be separated by a comma.

Each user ID has an access code or codes for that user. The user has access if the access code for the item contains at least one code in common with the access codes of the user. For example, if the route from LIB to DEV has the access code DEV, a user with access codes of DEV,QUAL,TEST would be able to make the transfer, but a user with access codes of MAN,QUAL,TEST would not.

Access codes can also include logical AND and logical OR. This notation allows great flexibility in allocating access codes to functional groups and in allocating users to one or more of those groups.

In the access code for the item, codes within parentheses are evaluated as a logical AND. Codes outside parentheses are interpreted as OR. For example, the access code (DEV,TEST),QUAL means '(DEV and TEST) or QUAL'.

The following table gives some examples of user access codes and item access codes. A *tick* signifies a true evaluation and the user has access, and a *cross* that they do not.

Item Access Codes	**Meaning**	**User Access Codes**			
		DEV,1	DEV,3	SYS,DEV,1	QUAL,MAN
SYS,DEV,QUAL	SYS or DEV or QUAL	✓	✓	✓	✓
(DEV,1),QUAL	(DEV and 1) or QUAL	✓	×	✓	✓
(SYS,1),(DEV,3)	(SYS and 1) or (DEV and 3)	×	✓	✓	×

Access Controls

Transfer Access

Each transfer route must have at least one access code assigned to it. Access control therefore applies to all transfers.

More detailed information can be found in the VC/m Reference Manual.

Menu and Menu Option Access

The Menu maintenance in VC/m Set-up can be used to restrict access to both standard and customized VC/m menus and menu options.

By default, all users have access to all options.

More detailed information can be found in the VC/m Reference Manual.

Location Display

Access codes can be used to restrict the locations which are displayed on the browser interface. By default, all locations are displayed for all users. The primary purpose of these access restrictions is to increase the speed of the interface.

More detailed information can be found in the VC/m Reference Manual.

Cancel a Check-out

By default, only the user who checked an object version out is able to cancel the check-out. However, individual users can be given permission to cancel any check-outs. This may be useful, for example, for a team or project manager.

More detailed information can be found in the VC/m Reference Manual.

User IDs and Terminal-Mode Interface Log-in

Each user of the VC/m system should have a user ID to determine their access levels.

Each user ID must be unique after conversion to be all upper case.

If VC/m is installed on Windows 95 or later, UNIX /Linux or OpenVMS, VC/m checks the account name which has been used to log onto the operating system. When starting the character-based interface, if this account name corresponds to a VC/m user ID, the user is logged into VC/m automatically without the need to re-enter their user ID.

If the host operating system does not automatically provide a user ID for VC/m to use, or the account name does not correspond to any user ID in VC/m, the user will normally be prompted for a user ID when they start the VC/m character-based interface.

The account name will be defaulted or prompted for as described above for any character-based entry to VC/m. This means that the access controls will apply even if a user invokes a menu or menu option directly.

If a unique VC/m user ID is not provided by the operating system, for example because all the users log in with the same user ID, it is possible to allocate a unique user ID for a discrete terminal session outside VC/m. This removes the need for VC/m to prompt users for their user ID, while still implementing the user access rights as defined by the VC/m system manager.

To do this, set the following global node, for example in the log-in process, for each user who uses VC/m:

```
set ^%vcvc("user",$job)=User ID
```

Note: VC/m only uses this user ID if it has been set up as a valid VC/m user ID. If the value of this node does not correspond to any VC/m user ID, VC/m will prompt for the user ID to be entered.

You should ensure that this global node is deleted at the end of each user's session, otherwise the next user invoking VC/m with the same job number ($job) may receive a default ID belonging to the previous user.

There is also an unconditional user ID main menu which forces a user to log in by always prompting for the user ID. To use this, enter the following command at the M prompt:

```
do ^%vcx
```

Log-in with IIS

If Internet Information Services (IIS) is used on Windows, the authentication to the browser VC/m interface can be integrated with the Windows authentication, eliminating the VC/m Login dialog popup. To set this up, do the following:

1 In IIS, open the Properties for your VC/m virtual directory.

2 Select the 'Directory Security' tab.

3 Click on the 'Edit' button for 'Anonymous access and authentication control'.

4 Uncheck 'Anonymous access' and make sure that 'Integrated Windows authentication' is checked.

5.11 Menu Customization

Customized Menus

The Menu maintenance in VC/m Set-up can be used to create new, customized menus which provide tailored sets of options for use by different groups of users. Menus for both the character-based and the browser interface are created in the same way. It can also be used to set access levels for menus.

The procedure for setting up customized menus has two stages:

1 Create the new menu.

2 Make the menu available to users.

A menu name can be any string of letters, numbers and punctuation characters. Any menu name which includes at least one lowercase letter is a standard VC/m menu.

More detailed information can be found in the VC/m Reference Manual.

Creating a New Menu

To create a new menu:

1 If a similar menu exists which can be used as a basis for the new one, use the Copy option in Menu maintenance to copy this to a new name.

2 Use Add /Modify to edit the menu (or create a new one).

3 Enter the name of the menu.

4 Select the type to be Menu and enter a description. If required, enter the access codes for the menu.

5 Use the Menu edit bar to edit, add and remove items from the menu. Each item can be a sub-menu or a menu option. The details of the item to be changed are displayed in the lower part of the screen. The menu options can be defined directly. Each sub-menu must itself be defined using the above procedure.

6 When the changes have been completed, select the OK button from the Menu edit bar.

More detailed information can be found on p. 110 and in the VC/m Reference Manual.

Making a New Character-Based Menu Available

Users can reach a new menu in two different ways. When a new menu has been created, it can be made available in one or both of these ways.

The first option is to replace one of the menus on the standard VC/m menu. In order to replace a standard VC/m menu, all of the menus above the menu must be replaced with customized copies as well.

The procedure to replace one of the second-level menus is as follows:

1 Copy the standard VC/m top-level menu (vc) to another name using the Copy option on Menu maintenance.

2 Use Add /Modify to edit the new menu and replace the standard second-level menu with the customized one.

3 Use screen 1 of the Properties option in VC/m Set-up. Under Custom Menus, add the name of your top-level menu as the value for Main menu.

In order to replace a menu at a lower level, this procedure must be repeated for each level in turn. It is easiest to start at the lowest level and work up to the top-level menu.

The second access option is to invoke the menu directly. In order to replace a standard second-level menu when it is invoked directly, use screen 1 of the Properties option in VC/m Set-up.

More detailed information can be found in the VC/m Reference Manual.

Making a New Browser-Interface Menu Available

For the main menu:

```
set ^%vcvc("mnuMain")="menu_name"
```

where menu_name is the name of the customized menu.

For the main menu when disconnected:

```
set ^%vcvc("mnuDisconnected")="menu_name"
```

where menu_name is the name of the customized menu.

Setting Access Levels for a Menu

To set access levels for a menu:

1. Use Add /Modify to add a new menu or modify an existing one.
2. Enter the name of the menu.
3. If the menu is not a standard VC/m one, select the type to be Menu and enter a description.
4. Enter the access codes.
5. If the menu is not a standard VC/m one, select the OK button from the Menu edit bar.

More detailed information can be found in the VC/m Reference Manual.

Customized Menu Options

The Menu maintenance in VC/m Set-up can be used to set access levels for menu options. These apply to both the character-based and the browser interface. It can also be used to add new, customized menu options to menus.

A menu option name is the line label of the M code to be invoked (e.g. ^ABC or LABEL^ABC). This can include parameters where appropriate. Those starting with ^%vc or fn are standard VC/m menu options.

More detailed information can be found in the VC/m Reference Manual.

Adding a New Menu Option

Menu options can only be added to customized menus, so the first stage is to set up a customized menu, if one does not already exist. Once this has been done, edit the menu to add the new menu option.

More detailed information can be found on p. 108.

Any M routine can be added to a VC/m character-based menu, whether a standard VC/m menu option or not.

Certain standard VC/m menu options can take parameters. These may be included when setting up customized menu options:

Menu Option	Line Label	First Parameter
Standard transfer	^%vc220	Function code, e.g. XFER
Change request maintenance	^%vc330	Change request type

Any valid JavaScript code can be added to a VC/m browser-interface menu. There is also a function, fnMenuRun("routineName"), which will run an M program on the server. Any output from the program will be interpreted as html in a new browser window.

When a customized menu option is added to a menu, there is no direct way to add access codes. If access controls are required, it must also be set up as a separate menu item with its own access codes (see below).

Setting Access Levels for a Menu Option

To set access levels for a menu option:

1 Use Add /Modify to add a new menu option or modify an existing one.

2 Enter the line label for the menu option, including any parameters.

3 If the menu option is not a standard VC/m one, select the type to be Function and enter a description.

4 Enter the access codes.

More detailed information can be found in the VC/m Reference Manual.

5.12 Peripheral Devices

In order to use a printer or other output device, e.g. a file, it must first be defined in the “Printers” option in the VC/m Set-up. Each output device is identified by a logical name. This need not be the same as its $IO value.

The output device definition links to the device type table. Several pre-defined devices are available. If these are not exactly as required, they should be copied and then amended as necessary.

VC/m will run on any terminal which conforms to ANSI standard 3.64 (e.g. VT220, VT420, and most PC terminal emulators).

Each terminal is identified by its $IO value. If an entry is not defined in the terminal file, VC/m assumes that the device is of the type DEFAULT, as defined in the device type table. DEFAULT should be configured to support all the terminals which will be used with VC/m over a terminal server or network.

5.13 Integration with Other Development and Support Tools

Controlling Component Editing

Controlling whether or not a component can be edited at a location is an important part of an integrated VC/m solution. It is a general principle that when component is edited, the copy which is being edited must be the master copy.

For text and binary components, this is achieved by setting the file to be read-only or not at the operating system level. A file is read-only unless it is the master copy (i.e. checked-out). This behavior is controlled by:

```
set ^%vcvc("setReadOnly")=1
```

More sophisticated interfaces are also available or can be created.

For M routines etc, VC/m provides a variety of ways to interface to the editors.

No standard methods are available for controlling access to global components, so strict procedures must be followed for these.

Interfacing from an Editing Tool

Once an object has been checked-out for development, there are a whole range of editing tools which may be used. VC/m provides a variety of ways to interface with these.

At the simplest level, an interface to VC/m only has two requirements:

Firstly, a check must be made that the copy of the component which is being edited is the master copy, i.e. that it is checked out to the location.

Secondly, the date /time stamp must be updated when the component is saved. VC/m keeps a record of the last changed date /time stamp for every object. When any change to a component is saved, VC/m needs to be informed of this.

Interfaces are available from various common development products. Additionally, a number of function calls are provided which allow users to create an individual editor interface which is suited to their needs.

Interface from Serenji

Serenji is an editor and debugging environment for M systems, supplied by George James Software. It can be used with Caché, DSM, GT.M and MSM. VC/m comes with a set of hooks to integrate with Serenji.

With the hooks installed, a Serenji user who tries to edit a routine which has not been checked out, will be unable to open it for editing. They will be asked whether they wish to open it in read-only mode.

To set up the Serenji interface, enter the following at the M prompt on the VC/m server:

```
do setup^%vc670
```

You can choose whether or not to allow users to edit components which have not been registered to VC/m. You can also choose whether or not users will be allowed to edit components in namespaces that are not M-type physical locations in VC/m.

If you wish to customize the interface, see the Serenji documentation for further information.

Interface from Caché Studio (5.x or later) or Ensemble Studio

Caché 5 Studio is an editor which is provided with Caché 5 and runs on Windows. VC/m comes with a set of hooks to integrate with this.

On Caché 5.0 VC/m places some classes into CACHELIB in a package named %VCm. After every Caché 5.0.x update you must run the following command in the VCM namespace to reload these classes:

```
D Cache5^%vcins()
```

Starting with Caché 5.1 these classes have been relocated into a package named VCmStudio in the VCM namespace. Use package mapping to make this package accessible to all namespaces that require it. A single mapping can apply to all local namespaces.

A Studio add-in named "VC/m Status of Project" provides a summary of the VC/m status of the current project and its members. It also gives access to the VC/m transfer dialog so that appropriate operations can be performed. For example, new items can be registered to VC/m.

A Studio template named "VC/m Operations" provides similar information about a single item. However, a Studio limitation means that no template can be invoked on a source-controlled item which has not been checked out. This template cannot consequently be used on checked-in items.

To configure, perform the following steps:

Pick the VC/m source control class

On 5.0 this is done as follows for each namespace in turn:

1. Connect Studio to the namespace.
2. On Studio's Tools menu, choose Source Control, then Source Control Class. In the dialog which appears, highlight %VCm.Studio.SourceControl and click OK.

On 5.1 and later and on Ensemble the procedure is:

1. Open the System Management Portal, go to Configuration and select the Studio Source Control Settings.
2. For each namespace, set VCmStudio.SourceControl as the class to use.

Set the parameters controlling how VC/m operates in each namespace

The parameters are held per namespace. For each VC/m-controlled namespace, perform the following steps from Studio while connected to the namespace:

1. On the Tools menu, choose Add-Ins, then Add-Ins. From the dialog, select VC/m Configuration and click OK.
2. On the configuration page, select the VC/m location which corresponds to this namespace. Enter a VC/m change request which will be used by default whenever a Studio source control operation in this namespace requires one, e.g. on checkout. In the transfer routes section, select the source or destination locations (normally a library location). Alter the transfer function codes if necessary. For each change request type, you can specify how VC/m links such change requests to Studio projects of the same name.

The following options are available for linking a change request to a Studio project of the same name:

Option	Behavior
No linkage	A project whose name matches a change request of this type will undergo no special integration with VC/m.
Refresh	When opening a project whose name matches a change request of this type, the project's contents will be redefined as the Studio-applicable components of the objects on the change request, unless the project's description does not contain a VC/m marker in the first line (see General tab of Project Settings dialog in Studio). When refresh occurs, a message is written to the Studio Output pane and the project's description text is updated. Additionally, when Check Out, Check In or Get Latest is performed from Studio, the project name supplies the change request to be used if one is required (e.g. during check-out, when a new version is created), in preference to the change request specified on the VC/m configuration dialog.
Create and Refresh	Behavior is as described in the previous paragraph. Additionally, provided an additional VC/m setting has been made (see below), projects will automatically be created in this namespace for every change request of this type having at least one component present in the corresponding location.

To enable "Create and Refresh" behavior, a transfer callout is required. In the browser interface's Setup folder, under the Transfer section of the Properties folder (or on screen 1 of the Properties option in VC/m terminal mode Set-up), add the following to the "Transfer termination" ("Per-transfer, termination" in terminal mode) field:

For Caché 5.0:

```
##class(%VCm.Studio.SourceControl).afterTransfer()
```

For Caché 5.1 and later:

```
##class(VCmStudio.SourceControl).afterTransfer()
```

This callout is defined automatically on new VC/m installations on Caché.

Once these settings have been made, the next transfer of any type to this location will create all required projects, regardless of whether or not the transfer is associated with the corresponding change requests.

In some situations, Studio needs to access the standard VC/m browser interface and requires a suitable URL. A default one is set during installation of VC/m. If this is not correct, it can be altered on screen 6 of the Properties option in VC/m Set-up.

For an individual user, if the Windows user name does not match the VC/m user name, set the Username in the Caché Connection Manager dialog. On Caché versions after 5.1 you will need a corresponding login account.

Interface from Caché 4.x Studio

Caché 4.x Studio is an editor which is provided with Caché and runs on Windows. VC/m comes with a set of hooks to integrate with this.

With the hooks installed, when a Caché Studio user opens a component, they see a dialog box which gives the status and other information about the component which they are about to edit. If it has not been checked out, it will be opened in read-only mode.

The Caché 4.x Studio interface needs to be set up on each of the VC/m Windows clients. It makes an entry in the registry. An uninstall file is also provided.

The registry install and uninstall files are named VcmCacheStudio*.reg. They are plain text files which can easily be edited.

The files assume that the Caché configuration is named CACHE. If it is different on the VC/m client, then the .reg files must be edited before they are applied. The files contain details of how to do this.

Once any necessary changes have been made to the files, the interface can be installed on a client simply by double-clicking on the appropriate .reg file, and agreeing to add the information to the registry.

There are some built-in switches which make commonly-requested changes to the standard functionality. These can be enabled and disabled by running the following and responding to the prompts:

```
do setup^%vc620
```

There is also a hook which allows site-specific checks to be made.

Interface from M/SQL

M/SQL is a development tool used on older Caché systems. VC/m comes with a set of hooks to integrate with M/SQL. To set this up:

1 Log into %SYS and run ^PROTECT. For the directory containing the FDBMS software (normally in c:\cachesys\mgr\fdbms), give all globals Read/Write/Delete access.

2 Create a namespace called %FDBMS, mapped to database FDBMS.

3 Log into namespace %FDBMS and run ^%msql.

4 Edit the form 'BaseTable'.

Add the following routine trigger to the Post Retrieval Trigger (Pst Ret Trigger) for the form:

```
checkout^%vc1msql("MMT",{BaseTabName})
```

If there are other triggers here, this should be inserted as the last one. Once this change has been made, re-compile the form.

Note: Do not change the type of form compilation. If it defaults to OLD, leave it as OLD. If it defaults to NEW, leave it as NEW. If the form compilation type is NEW, you may need to set the Use Local Storage argument to NO in order to avoid a STORE error during compilation.

5 Edit the form 'Form Definition'

Add the following routine trigger to the Post Retrieval Trigger (Pst Ret Trigger) for the form:

```
checkout^%vc1msql("MMW",{FormName})
```

If there are other triggers here, this should be inserted as the last one. Once this change has been made, re-compile the form.

6 Edit the form 'Menu Object Definition'

Add the following routine trigger to the Post Retrieval Trigger (Pst Ret Trigger) for the form:

```
checkout^%vc1msql("MMB",{MenuName})
```

If there are other triggers here, this should be inserted as the last one. Once this change has been made, re-compile the form.

7 Add the following call to the Per-transfer, termination call-out on screen 1 of the Properties option in VC/m Set-up:

```
import^%vc1msql
```

8 Change the active namespace /directory configuration back to the normal namespace /directory. If you do not, it will reset back the next time it is re-started.

Interfaces to the Browser Interface

VC/m's client integration kit facilitates the integration of VC/m's browser interface with client-side development tools such as Delphi, Visual Basic and Microsoft Word.

For some client programs, e.g. Microsoft Word, the VC/m client integration kit includes templates for integration, using macros and add-ins.

The main integration kit consists of two customizable .hta files. When these are invoked, they run within Internet Explorer in application mode.

The first file, vcm.hta, is a shortcut which launches the VC/m browser interface. This can be added, for example, to a Windows desktop, start menu or favorites list. It can also be added to any client-side program, such as Excel, which supports customizable menus. It can then appear as a menu option or button.

The second file, vcmClient.hta, is a shortcut which launches the VC/m browser transfer dialog. The name of the file to be transferred is required input. It can also take as input the type of transfer, the default 'From location' and the default 'To location'. This file can be added to any client-side program which supports customizable menus and can pass its current document or file name as an argument.

The default Windows setting for the .hta file type will cause it to run in Internet Explorer. When the files are run from within another application, the application to be used must be specified.

For Internet Explorer 5.x onwards, the application is:

```
mshta.exe
```

For Internet Explorer 4.x, the application is:

```
C:\Program Files\...\Iexplore.exe
```

Note: An .hta file contains standard html content. However, Internet Explorer treats these files in a different way from regular .htm files. mshta.exe, which is part of Internet Explorer, runs in application mode, which gives more control over what can be done programmatically. It also has a less strict security model, allowing commands such as window.close to execute without prompting the user for confirmation. To ensure that PC clients are not compromised by the delivery of .hta documents over the internet, Internet Explorer will only run in application mode if the .hta document is installed on the local or shared file system. This makes use of .hta documents safe. Within VC/m, the .hta files are used solely for launching the VC/m browser interface. The .hta file then immediately closes itself and plays no further part.

The files are found in the VC/m installation directory.

To use the VC/m client integration kit:

1 Make a copy of the file vcm.hta.template and name it vcm.hta.

2 Make a copy of the file vcmClient.hta.template and name it vcmClient.hta.

3 Edit both files and replace the IP address 127.0.0.1 with the IP address of your server.

4 If you are using a different URL to access VC/m, make any other necessary changes to the URL string in the files.

5 Place these customized files on each PC client, for example in C:\Program Files\George James Software\VCm. Alternatively, place them on a file server where the PC clients can access them. Remember that remote PCs may not always have access to a central file server and may therefore need to have their own local copies.

Interface from Microsoft Word

The Microsoft Word interface adds a VC/m menu and a VC/m toolbar to Word. By default, they both contain:

- VC/m: Invokes the VC/m browser interface
- Out: Invokes the check-out function for the current document
- In: Invokes the check-in function for the current document
- Transfer: Invokes the transfer function for the current document

The current open document can be transferred by using the Out, In and Transfer shortcuts. Defaults for these are set by customizing the interface. Any other object can be transferred by invoking the VC/m browser interface.

To set up the interface from Microsoft Word:

1 Install the two .hta files from the VC/m client integration kit on the PC client.

More detailed information can be found on p. 119.

2 Make a copy of the file vcmWord.dot.template and name it vcmWord.dot.

3 Open the template in Microsoft Word and then invoke the Visual Basic Editor (Tools, Macro, Macros, Visual Basic Editor, or Alt+F11.)

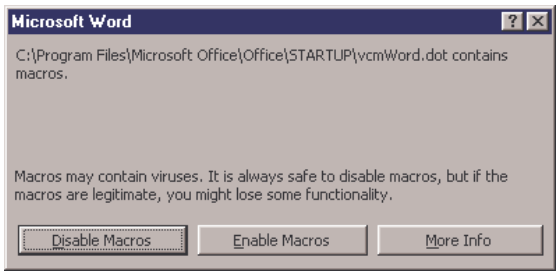
4 Customize the macros to specify the location where the VC/m client-side components are installed, e.g. C:\Program Files\George James Software\VCm.

5 Customize the constants vcmLibrary and mySandbox to be, respectively, the VC/m location names for the library and the work area where documents will be edited.

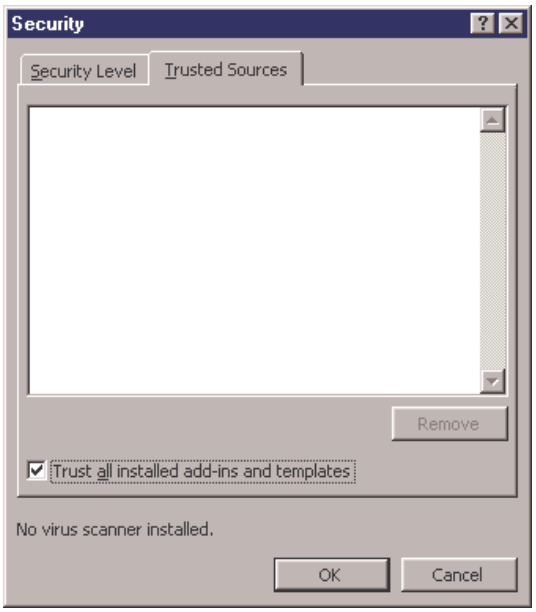
6 Typical check-in, check-out and transfer functions have been defined. If required, create additional ones suited to your VC/m configuration. If you create additional transfer functions, you should also add them to the custom VC/m toolbar and /or the VC/m pull-down menu which are part of this Word template.

7 Copy your customized version of vcmWord.dot to the location which the PC client's copy of Microsoft Word has configured as its Add-in startup directory. Typically this is C:\Program Files\Microsoft Office\Office\Startup.

8 When you open Word, you may get a message which says:



This message can be suppressed by adjusting the security settings for Word. From the menu, select Tools, Macro, Security. Select the Trusted Sources tab. Tick the box for Trust all installed add-ins and templates.


Security
Security Level
Trusted Sources
Remove
Trust all installed add-ins and templates
No virus scanner installed.
OK
Cancel

Interface from Delphi IDE

To set up the interface from Delphi:

1 Install the two .hta files from the VC/m client integration kit on the PC client.

More detailed information can be found on p. 119.

2 Add your vcmClient.hta file to the Delphi toolbar using the following values:

For Internet Explorer 5.x onwards:

```
Program:  mshta.exe

Working dir:

Parameters:
vcmClient.hta?function=XFER&lcode1=DEV&lcode2=TEST&ctype=BIN&component=
$EDNAME
```

For Internet Explorer 4.x:

```
Program name:  c:\Program Files\...\Iexplore.exe

Working dir:

Parameters:  vcmClient.hta?component= $EDNAME
```

The following parameter values are permitted:

Parameter	Value	Notes
function	Function code: XFER, IN, OUT, etc	Optional
lcode1	'From location'	Recommended
lcode2	'To location'	Optional
ctype	Comma-delimited list of component types	Recommended
component	$EDNAME is the name of the currently open Delphi unit *Note: There must be a space before the $ otherwise Delphi doesn't parse it correctly.*	Mandatory

3 If required, add vcm.hta to the Delphi toolbar using the following values:

For Internet Explorer 5.x onwards:

```
Program:  mshta.exe

Working dir:

Parameters:  vcm.hta
```

For Internet Explorer 4.x:

```
Program name:  c:\Program Files\...\Iexplore.exe

Working dir:

Parameters:  vcm.hta
```

Creating an Interface from an Editor

VC/m has several calls which can be added to a routine editor to create an interface from the editor to VC/m. These can provide various levels of sophistication, depending on the users' needs.

At the simplest level, an interface to VC/m only has two requirements. A check must be made that the copy of the component which is being edited is the master copy, and the date /time stamp must be updated when it is saved.

It is a general principle that when a routine or other component is edited, the copy that is being edited should be the master copy. If the copy being edited is indeed the master copy, editing can be permitted with no further intervention until the changes are saved.

VC/m keeps a record of the last changed date /time stamp for every object. When any change to a component is saved, VC/m needs to be informed of this.

More sophisticated interfaces can be implemented which provide additional functionality and make the use of VC/m even more convenient. For example:

- Automatic invocation of the VC/m check-out screen if the master copy is not at the location where the editor is being used.
- Automatic invocation of the VC/m object registration if the routine to be edited is new or is not already associated with an object. For new routines, VC/m modules can cause a skeleton routine to be generated automatically.
- Automatic notification to VC/m when a routine, or other component, is deleted.

A number of function calls are provided which allow users to create an individual editor interface which is suited to their needs. It is the user's responsibility to check that no copyright is infringed by amending an editor which is supplied by a third party.

Component Status

This function call returns a backslash delimited string that contains the status and other information about a component. The information can be used to determine what action or actions should be performed before editing of a component is permitted.

Usage:

```
set status=$$status^%vc1edt(ctype,cname)
```

Inputs:

ctype	=	Component type
cname	=	Component name

Output:

$$status	=	flag\obv\msg flag = status of component at the current location -1 if error in format of component name 0 if master copy of component is not at current location 1 if master copy of component is at current location 2 if component is not known to VC/m obv = object and version if known to VC/m, format is object/variant.version msg = error message if flag is -1

Update Date /Time Stamp for Component

This call updates the date /time stamp for a component at the current location. It enables a routine editor or other maintenance tool to inform VC/m that a component has been changed.

Usage:

```
do update^%vc1edt(ctype,cname)
```

Inputs:

ctype	=	Component type
cname	=	Component name

Output:

None

Create or Check Out an Object

This call enables the object that a component belongs to be checked-out. If the component does not belong to an object, it allows a new object to be created or allows the component to be added to an existing object.

In all cases the screen is cleared and the appropriate VC/m screen is invoked. On exit from this call, the screen is also cleared. The user may or may not actually check out the object when this call is invoked. The status of the component should therefore be re-checked on return from this call.

Usage:

```
do checkout^%vc1edt(%usr,ctype,cname)
```

Inputs:

%usr	=	User ID (must be in the VC/m user ID file)
ctype	=	Component type
cname	=	Component name

Output:

None

Is this a VC/m-Controlled Place?

This call enables a routine editor or other maintenance tool to determine whether or not the current UCI, namespace etc. is VC/m-controlled. If it is, only the master copy of a component should be edited here and VC/m should be informed whenever a component is changed in this place. If it is not a VC/m-controlled place, VC/m may still be aware of it but it is not necessary for routine editors to make any further calls to VC/m.

> *Note: The calls to status and update are both sensitive to whether or not a place is controlled by VC/m. They can therefore be called directly, without the need to use this call first.*

Usage:

```
set vcm=$$vcd^%vc1edt
```

Input:

None

Output:

$$vcd	=	Flag indicating whether this place is VC/m-controlled 1 = Controlled by VC/m 0 = Not controlled by VC/m

Delete a Component from an Object

This call enables an editor or maintenance tool to inform VC/m that a component has been deleted. VC/m does not actually physically delete the component when this call is made, but it will delete this component in other locations when the object that owned the component is subsequently transferred there.

Note: If the master copy of the object that owns the component is not at the current location, VC/m will not delete the component. No error message is given as this condition should be detected by calling $$status (component status) before permitting the deletion to occur.

Usage:

```
do delete^%vc1edt(ctype,cname)
```

Inputs:

ctype	=	Component type
cname	=	Component name

Output:

None

Integration with Visual SourceSafe

Visual SourceSafe (VSS) is a Microsoft version control product. It does not have the configuration management facilities which are an integral part of VC/m.

VC/m provides facilities for integration with a VSS database. This is primarily of use where VSS has been in use at a site prior to the installation of VC/m, and the users wish to retain the data in the VSS database. VC/m integrates with VSS by storing part or all of the library in the VSS database, rather than in its own library storage format. The physical location for the VSS database has a storage format of 'vss'.

VC/m uses a physical location with storage format vss in exactly the same way as it uses its own library, with storage format L. Multiple versions of objects can be placed into and retrieved from a VSS library. When placed in the library, each component is labeled with the object version which it belongs to.

The integration is set up as follows:

1 Using the Visual SourceSafe Administrator, create a user account which corresponds to the user account which VC/m will use to connect to VSS. For Caché on Windows, by default this user will be the local SYSTEM account, which is the user name of Caché background processes.

Do not specify a VSS password so that the Windows access security is used.

It is recommended that other user accounts are configured with read-only access.

2 If there is not an existing VSS database, create a new one. Use the Tools, Create Database option from the menu in the Visual SourceSafe Administrator function.

If required, projects can be created with VSS, although this is not necessary.

3 At an M prompt on the VC/m server, enter the following and then enter the appropriate information for your system in the specified format.

Note: By default, the directory containing the Visual SourceSafe script files will be the directory where VC/m is installed.

```
do setup^%vclvss

Directory containing Visual SourceSafe script files (NB avoid spaces in
path - use short name if necessary)? c:\progra~1\george~1\vcm\

Directory to contain temporary files (NB UNC not supported)? <:> c:\temp\

Path to Visual SourceSafe directory containing ss.exe? c:\program
files\microsoft visual studio\common\vss\win32\

Setup complete
```

4 Set up a VC/m library location for the VSS database.

More detailed information can be found on p. 77 and p. 78.

Note: When testing, it is important to test that each user has the correct access rights from both the character-based and the browser interfaces.

For extra security, a location can be set up which stores the information in VC/m's own library storage format as well as in the VSS database. This guards against data loss caused by corruption of the VSS database. Auto-transfers can be set up to keep the two databases in line.

5.14 Component and Object Take-On

Once the VC/m master files have been set up and any set-up options have been defined, you are ready to register your files and software in the library and let VC/m start to control your configuration management.

Before you load all your objects into the library, it is recommended that you create some temporary objects and transfer them through the development, test and release cycle in order to test that the transfer routes, dependencies and access codes have been set up in the way that you intended. You may find that you need some fine tuning of the master files before you are ready to load all your objects into VC/m.

There are three different strategies that can be adopted for loading your files into VC/m:

- The Big Bang approach

  Load all the files into the library at one time, and then check out all the objects that are currently being worked on.

- The incremental approach

  Define an object to VC/m only when there is a need to change it. This approach uses less time initially, but will not bring all software under VC/m control for some considerable time. Indeed, it is possible that some objects will never be taken on, because they never get changed.

- The combination approach

  The third strategy is to use the incremental approach initially, then, after a period of time, use the big bang approach to load in all the files that have not yet been loaded.

Taking On Non-R Type Components

Note: A routine called %vc413 is available which is a template for stages 2 and 3 below.

Procedure

1 Set up the logical location to load from. This must include a mapping for R type components to an M physical location.

2 Set up the following global nodes for the components to be loaded:

```
^%vcsel(listName)=numberOfComponentsInList\R
^%vcsel(listName,componentType_"."_fullComponentName)=""
```

For example:

```
set ^%vcsel("workfile1")="3\R"
set ^%vcsel("workfile1","MAC.abcdef")=""
set ^%vcsel("workfile1","T.cam/web/index.htm")=""
set ^%vcsel("workfile1","BIN.cam/images/logo.gif")=""
```

If the components are binary or text files, you can pipe a directory listing to a file, then open this in M and read in the contents.

3 If the object names do not correspond to the component names (which they probably won't if the components are binary or text files which include a path), write M code which will set up global nodes linking the component type, name and object name.

4 Set up a change request for the take-on.

5 Run the Load function from the Manager menu.

6 Enter the logical location from stage 1.

7 At the "Routine(s)" prompt, enter @ and the name of the list you created in stage 2, e.g. `@workfile1`. The prompt text changes. Press <return> to move on from here.

8 Fill in the other prompts as usual and run the Load function.

Specifying the Object Name when Loading

If you want to force the VC/m load function to populate objects according to a particular naming scheme, set up the following data structure for each component:

```
^%vcal2(2,component_type,component_name,"force")="\object\"
```

where:

'component_type' is the code for the component type

'component_name' is the name of the component

'object' is the name of the object

For example, `^%vcal2(2,"R","AAA","force")="\Start\"` will load the routine AAA into an object called Start.

5.15 Removing and Deleting

Removal and deletion occur at different levels within VC/m.

The Component option under Object maintenance is used to delete a component from the master location of an object version. The user is prompted as to whether the component should be physically removed from the location.

A component is physically deleted from any other controlled location only when an object version which does not include it is transferred to the location and displaces an object version which did include it.

All the components of an object can be physically removed by creating a new version of the object which contains no components. When this version is transferred to a location where the previous version existed, all the components at the location are removed.

Also, if an object is removed from a system which it previously belonged to, it can still be transferred to a location where that system is installed. The effect of this is to physically delete the components of the object from that location.

The VC/m menus offer several options for deleting logical entities.

The Version option under Location is used to de-activate a location. It can be used to inform VC/m that a complete location has been physically deleted, or to remove a redundant location which is one of several mapping onto the same physical location. It deletes all the information about object versions at that location from the VC/m database. The components at the location are not physically deleted, and the audit trail is not deleted.

The Delete option under Object is used to delete an object version which was created in error. An object version can only be deleted here if it has not been checked out or transferred to any location. The components at the location are not physically deleted, and the audit trail is not deleted.

The Cancel option on the Manager menu is used to delete all copies of an object version at all locations and replace them with the preceding version. For example, if an object is checked-out for development, but for some reason the change is not required, the version may be canceled. The components of the canceled version are physically deleted, and the old components are copied to the locations. The audit trail is not deleted.

An object version can only be canceled if it is the latest version. Earlier versions can therefore only be canceled if all succeeding versions have been canceled in turn. Also, an object version can only be canceled by the user who created it.

If version 0 of an object is canceled, all the information about that particular variant and its components will be deleted from the database.

The Delete function on the Manager menu it used to completely remove an object version from the VC/m database. All the location and component information for the object is deleted. The actual components associated with each object are deleted from library locations, but not from any other locations. If all versions of the object are deleted, the audit trail entries are also deleted. This function should be used with extreme caution.

5.16 Managing Concurrent Development

To enable concurrent development work to take place on an object, two new versions must be created from the same original version. These must be checked out to different locations. VC/m does not allow a new version to be created from one which is currently checked out, nor does it allow two versions of the same object to be checked out to the same location.

The following diagram shows the process:

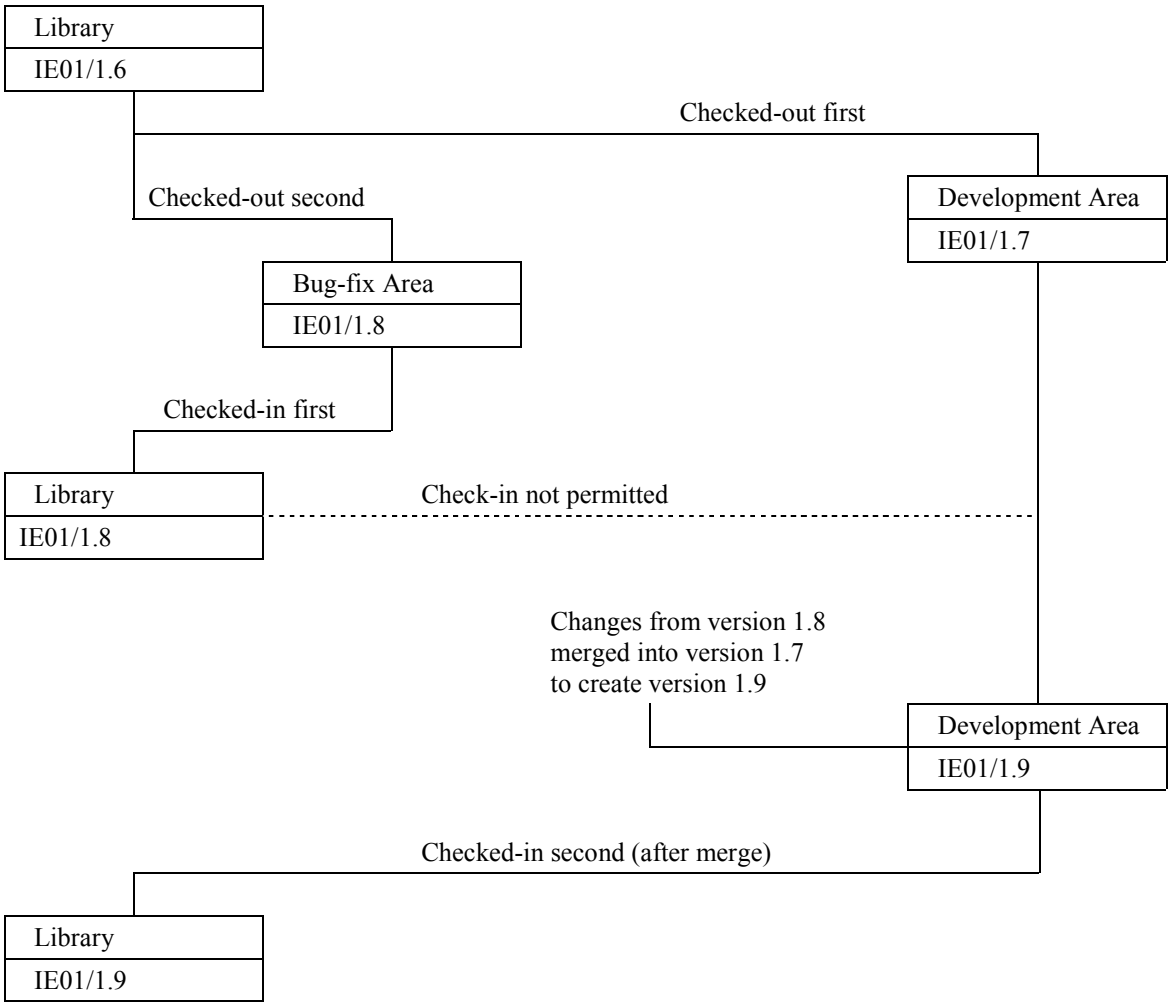


Figure 5.2 Concurrent development work

Note that if the object version is already checked out, VC/m will not permit it to be checked out again to the same location.

However, it will allow a new version of the object to be created and checked out, provided that it is checked out to an alternative location. For example, if object IE01/1.6 is currently the live version and IE01/1.7 has already been checked out to the normal development area, version 1.7 cannot be checked out. It is possible; however, to check out version 1.6 (thereby creating version 1.8), providing that it is not checked out to the same place as 1.7.

At a single version location, VC/m only allows an object version to displace another version if the earlier version is a direct ancestor of the later one.

If version 1.7 were to be checked in after version 1.8, the master version would lose the changes made in version 1.8. In this case, VC/m prevents the transfer because the version at the destination location is not an ancestor of the version being transferred.

If version 1.8 were to be checked in after version 1.7, the master version would lose the changes made in version 1.7. In this case, VC/m prevents the transfer because the version at the destination location is not an ancestor of the version being transferred. The ancestor of version 1.8 is actually 1.6, not 1.7.

This mechanism prevents concurrently developed versions from overwriting each other, but allows either version to be checked-in, and released, before the other.

> *Note: This validation is enforced for single-version locations, where one version will displace another version, but not for multi-version locations, where no displacement occurs. Concurrency conflicts will always be detected when an object is transferred to a single-version location, such as a live area. We recommend that concurrency is resolved prior to testing, by transferring all objects via a single version location. The test area itself is ideal for this purpose.*

When this situation is encountered, the changes made to the first version that is checked-in must be incorporated into the second version before it can be transferred. The diff function can be used to identify the differences. The actual merging of the software changes is a manual process. Once the changes to the first version have been incorporated into the second version, the merge function is used to inform VC/m that a merge has taken place.

When VC/m is informed that changes to the first version have been merged into the second version, it gives the second version a new version number which is higher than the first version, and makes that version its ancestor.

5.17 Housekeeping

To ensure that the VC/m system is kept in order, it is recommended that 2 tasks are run on a regular basis. The frequency with which this should be done will depend on the level of use and complexity of the system.

9 Run the Integrity checker with “Repair errors?” set to No. Any errors which are found should be investigated. Once any issues which are highlighted have been resolved, run the Integrity checker with “Repair errors?” set to Yes.

10 Run the Matrix report for all systems, objects and locations, but with Status set only to “E”. Any errors which are found should be investigated. Once any issues which are highlighted have been resolved, transfer each object along a route which will correct the error.

5.18 Managing Code on Other Machines

VC/m offers several ways of managing code which is on machines other than the VC/m server.

M Networking

In many cases the code can be managed simply by appropriate configuration of the M environment.

VC/m Networking: The Task Server

The Task Server is used to perform operations which for any reason cannot be done by the system which the user is connected to. Example uses of the task server include:

- Users connect to VC/m on a Unix system and need to move files on a Windows system.
- Users need to manage code on a different M implementation from the one they are connected to.
- Users manage code which needs to be compiled under a different Caché version from the one they are connected to.

Non-Networked Locations: VC/live

VC/live is a stand-alone installation of VC/m which is used for installing software on a machine which has no network connection to the VC/m server. It is typically used for transferring to a production server which is at another location or is firewalled from the development environment.

5.19 The Task Server

The Task Server is used to perform operations which for any reason cannot be done by the system which the user is connected to. Example uses of the task server include:

- Users connect to VC/m on a Unix system and need to move files on a Windows system.
- Users need to manage code on a different M implementation from the one they are connected to.
- Users manage code which needs to be compiled under a different Caché version from the one they are connected to.

The Task Server can run on any machine which has an M networking connection to the machine which the users are connected to. Most commonly, the users connect to the VC/m server and the task server runs on another machine. In some cases the users connect to another machine and the task server runs on the VC/m server. Sometimes users may connect to either machine and so a task server runs on both.

The task server is an M background job. It monitors the global ^%gjtask for tasks to be done. It then switches namespace to the place where the task is to be performed.

The physical location addresses must be defined using the remote protocol syntax.

Starting and Stopping a Task Server

To start a task server:

```
do fast1^%vczn
```

To stop a task server:

```
do fast0^%vczn
```

5.20 VC/live

VC/live is a stand-alone installation of VC/m which is used for installing software on a machine which has no network connection to the VC/m server. The install function is designed to be as automated as possible so that the risk of mistakes is minimized.

VC/live is a single-user version with limited functionality. There are no systems and no change requests.

Appendix III: Installation and Configuration Checklists has a checklist which should be used to ensure that all of the necessary steps are completed when configuring the VC/live software for the first time.

Starting VC/live

To start VC/live:

```
do ^%vcl
```

Starting VC/setup

To start VC/setup:

```
do ^%vcsetup
```

Using VC/live

Software is transferred from the VC/m server to the VC/live machine. The procedure has 3 steps:

1 The main VC/m server transfers the objects destined for the VC/live machine to a local sequential file location.

The transfer on the main VC/m server creates a series of files with the same name but different extensions. One of the files has an extension of $. This is the control file and it contains all the information which VC/live requires about the objects and the locations which they belong in. There is also one file for each type of component which is to be installed. The extensions for these are the component type code.

2 The user manually copies the files to a sequential file location on the VC/live machine using any available transfer protocol.

3 VC/live reads the sequential files and installs the objects from this sequential file location into the installation location.

Installation Mode

A VC/live machine can be configured to install the objects in one of three different modes. Firstly, an operator can use the interactive 'Install' option from the menu. Secondly, a non-interactive batch job can be run at an appointed time. This could be run, for example, when a system re-starts after a nightly shutdown for backup. Thirdly, a background install daemon can be set up which is running all the time and will install the objects non-interactively when they are placed in the sequential file location.

Hot Backups and Rollback

The install function includes an option for a hot backup of the objects which are being replaced. This means that, when a new version of an object is installed, the displaced version will be transferred from the 'to location' into the designated backup location. If the new version turns out to have a problem, the old version of the object can be quickly re-instated from the hot backup location, using the 'Rollback' option. This is particularly useful for a live system.

Locations and Routes

VC/m Server

A single-version sequential file location is set up for the objects which are to be transferred to the non-networked machine. All the components at this location must be mapped to one physical location with a storage format of 'F'. (This includes binary and text files.)

> *Note: In order for VC/live to function, it is essential that the corresponding sequential file location on the main VC/m server and the installation location on the VC/live machine have the same location name as each other.*

A route is defined to transfer the objects to this location.

If there is more than one VC/live installation location or more than one VC/live machine, a separate logical location is required for each one. However, one physical location can be used for files destined for more than one logical location. The install function will not allow a transfer file to be installed at a location for which it is not intended.

VC/live Machine

On the VC/live machine, a multi-version sequential location is defined for the objects which are awaiting installation. As on the VC/m server, all the components at this location must be mapped to one physical location with a storage format of 'F'. If there is more than one VC/live installation location, a separate sequential location must be defined for each one. The physical location can be the same.

A location is defined for the place where the objects are to be installed. This location *must* have the same location name as the corresponding sequential file location on the main VC/m server. If there is more than one area, a separate location is required for each.

A route is defined to transfer the objects from the sequential file location to the installation location. This route must have a function code of 'XFER'.

If the automated hot backups and rollback function are to be used, a multi-version sequential location is defined for the backup copy of the objects which are being replaced. Two transfer routes are required, both with a function code of 'XFER'. The first is a route from the installation location to the backup location. This enables creation of a backup copy of each object during installation. The second is a route from the backup location to the installation location. This enables the backed-up copies to be restored.

The installation parameters must be set up using the 'Install' option in VC/m Set-up. This option must be run in the location where the objects are to be installed to. If there is more than one location, change to each one in turn and run the option. If you are using hot backups, enter the code for the backup location in the appropriate field.

The default output device for the installation report can be defined on screen 4 of the Properties option in VC/m Set-up.

Configuring Non-Interactive Installation

In non-interactive mode, the install function runs silently. Installation messages are logged in the audit trail as normal. Any errors which relate to the install process itself, e.g. configuration and set-up errors, are logged in the audit trail under a pseudo-object called vcm$install.

So that the audit trail messages can be viewed, create a VC/m object called vcm$install/1.0. It does not need to be checked out or given any components.

The install process uses the user name of the current user. If the code is to be run in background, identify the process which will run the code and set up a user account with appropriate access.

Batch Mode

The following call is used to run the install process non-interactively:

```
do ^%vc490("XFER",installLocn,liveLocn,backupLocn,1,0,1)
```

This code can be called from a scheduled background job.

Install Daemon

To start the install daemon, run the following in the VC/m namespace or UCI on the VC/live machine:

```
job ^%vc490("XFER",installLocn,liveLocn,backupLocn,1,0,2)
```

To stop the install daemon:

```
do stop^%vc490
```

5.21 Operating System-Specific Information

Unix /Linux

Default Mode Settings

On Unix platforms, by default VC/m uses chmod to set the mode settings for all files to 666 for read /write and 444 for read-only. To change this behavior, set the following global nodes to a suitable callout:

```
set ^%vcvc("unixReadOnly")="label^routine"

set ^%vcvc("unixReadWrite")="label^routine"
```

Each callout must be written as a line label or routine call.

Usage:

```
do label^routine(file)
```

Inputs:

file	=	Full file specification

Output:

None

5.22 M Implementation-Specific Information

GT.M Information

Case Sensitivity on OpenVMS

The OpenVMS implementation of GT.M is not case-sensitive for routine names. All VC/m routines are therefore installed as uppercase routines. This does not affect the operation of VC/m in any way.

VC/m, however, *is* sensitive to upper and lowercase routine names. If you use VC/m to create and manage lowercase routines, remember that these are stored in uppercase by GT.M and are not distinguished from a routine with the same name that is in a different case.

MSM Information

Partition Size

VC/m requires approximately 12 Kbytes of partition space for its own operation. In addition, when a transfer is performed to or from an M location, enough free partition space must be available to load the routine into the partition. You should therefore configure your partitions so that they are at least 12 Kbytes larger than the size of the largest routine that you will transfer with VC/m.

Support for Diff on MSM-PC Systems

If you are using MSM-PC on a DOS platform, it is possible to invoke and run the DOS fc utility directly from VC/m by using the dosshell utility supplied by Octopus GmbH.

If this utility is not available, the Diff function can still be used, but VC/m only creates a DOS batch command file which must be executed independently to obtain the Diff output.

To enable use of this utility, set the following global node:

```
set ^%vcvc("dosshell")=1
```

To disable use of this utility, execute the following command:

```
kill ^%vcvc("dosshell")
```

Support for VC/m on MSM-UNIX Platforms

Where VC/m is used on MSM-UNIX systems, if the developers do not have sufficient access rights to execute UNIX shell scripts, VC/m can be configured to avoid this restriction.

The procedure to do this is as follows:

1. Create a UNIX directory called /vcm and in that directory create a file called vcshell. It should contain the following lines:

   ```
   # VC/m UNIX access shell script
   # VC/m version 1.3
   $*
   exit $?
   # end
   ```

2. Ensure that the /vcm directory and the vcshell script file have appropriate access and execute file permissions.

3. In the Manager UCI, set up the following global node:

   ```
   set ^%vcvc("nounixaccess")="/vcm/vcshell"
   ```

To disable this feature, execute the following command in the Manager UCI:

```
kill ^%vcvc("nounixaccess")
```

Known Problems

There is a compatibility issue with some versions of MSM such that after a Routine Restore has been performed, the ANSI cursor addressing functions on device 1 do not operate correctly. The cursor is positioned wrongly and occasionally spurious numbers appear on the screen. This can manifest itself when VC/m is first installed and used.

The problem can be resolved by shutting down MSM completely and restarting it. Logging out and back in is not sufficient to resolve the problem.

Appendix I: Installation and Configuration Planning

Before you configure and use VC/m, you should plan how you intend to use it and design the configuration management procedures that you want to use.

The following topics should be considered as part of this process:

VC/m Server

The VC/m and library are usually installed on the main development server.

1 Which machine will be used?

2 What operating system is on the server?

3 What M implementation and version are on the server?

4 Are there any global protection restrictions for % globals?

Other Machines

5 What other machines will VC/m interact with?

For each one:

- What operating system is running?
- What M implementation and version?
- Is there an M networking connection to the server? What kind?
- Do all machines where VC/m will be used have access to all the physical locations which VC/m will control, or are there restrictions?
- Are there remote sites to be controlled? How will the files and software be transferred there?

Locations and Routes

A common configuration for VC/m is:

- Check out to a development area
- Transfer to a test area
- Check in to the library
- Transfer to the live area

6 Which physical areas will be used for development?

Note: You will need at least 2 development areas if you wish to allow concurrent development. Where possible, there should be an area for each developer.

7 Which areas will be used for testing?

8 Which areas are live areas?

9 Are there any other areas which need to be kept up-to-date with the current copy of the application?

10 What procedures does the new system need to reflect? Should it replicate the current procedures, follow a modified version or be a complete re-design?

11 What logical stages are there /should there be in the development and release process?

12 Which users will be allowed to perform which stages of the procedure?

13 Are there any other constraints etc which you would like to be incorporated into the VC/m configuration?

14 Which single-version location will be used to detect concurrent development work?

15 Do you need to manage code from your MGR or SYS area? Do you use any special procedures and /or configurations for this at present?

16 What naming conventions will be used for physical locations, logical locations and users?

Objects, Change requests, etc

17 Which of the following are part of your source code and need to be managed by VC/m?

- Routines
- % Routines
- Globals
- % Globals
- Globals with control characters
- Text files, e.g. JavaScript files
- Binary files, e.g. executable files
- Screens
- System parameters
- Command files
- CSP pages
- CachéObjects class definitions
- MAC files
- M/SQL components
- Other. Please specify.

18 Do you want to group any software components together as single objects for configuration management purposes?

19 Should the software be divided into systems? If so, how?

20 Are different variants relevant for any of the objects?

21 Is every change in your application driven by a project, bug report or similar? What number ranges are used for these?

22 If not, what change request types will be used?

23 What naming conventions will be used for objects, systems and change requests?

External Applications

24 Do you use any application development tools such as M/SQL, DASL, MDM, WebLink, CSP, CachéObjects, ESI Objects, CyberTools, etc? Please specify.

25 What editors do developers use?

26 Do you want to interface to any project control or bug tracking systems?

Take On

After take-on, any software which is not yet in live will need to be marked as "checked-out". Some additional software may need to be identified as "being tested" or "on hold".

27 How is your current work in progress identified?

28 Approximately how many components need to be registered with VC/m and loaded into the library?

29 How will the software be taken-on? Big-bang, incremental or combination?

Back Up

30 How and when will the system be backed up?

Appendix II: Charts for Configuration Planning

Access

Access code within VC/m	Description /use

Users

Naming convention:

User ID within VC/m	Full name	Access code(s)	Password	Cancel objects for other users? (Y /N)

Systems

Naming convention:

System name within VC/m	Description /use	Logical locations

Physical Locations

Note: One suggested naming convention is to use lowercase for physical location names and uppercase for logical location names.

Naming convention:

Physical location name within VC/m	Actual physical location	Storage format (M /L /F /text /bin)	Control led? (C /U)	Multiple versions? (Y /N)	Description /use	Logical locations

Logical Locations

Note: One suggested naming convention is to use lowercase for physical location names and uppercase for logical location names.

Naming convention:

Logical location name within VC/m	Description /use	Allowable systems	Multiple versions? (M /S)	In location classes	Component mask	Physical location
NEW						

Location Classes

Naming convention:

Location class name within VC/m	Description /use	Logical locations

Transfer Routes

Fn. code	From loc.	To loc.	Access code /codes	Depend-ent locations	New active loc(s)	New master locatio n	Create new version? (Y /N)	Trans-fer type (C /M)	Audit level	Descrip-tion /use
	NEW									

Change Request Types

Type	Description /use	Prefix /Suffix	Statuses

Variants

Naming convention:

Variant code within VC/m	Description /use

Objects

Type of object	Criteria for combining components into an object	Possible components included	Naming convention

Modules

Module name within VC/m	Description /use	Form of last used object suffix	Skeleton object generator

Appendix III: Installation and Configuration Checklists

VC/m Server Installation Checklist

Install M

If you have not already done so, install M on the machine which is to be the VC/m server. If this process is not familiar, please consult George James Software for help.

Completed ☐

Install the VC/m Program Files

Copy the files from the CD or zip file to the installation directory.

More detailed information can be found on p. 27.

Completed ☐

Prepare the M Environment

Set up the namespaces or UCIs which are required for the VC/m software and library. Set up mappings to the VC/m routines and globals for each area where VC/m will be used. If your M implementation supports global protection, you need to ensure that all VC/m globals have the correct protection attributes.

More detailed information can be found on p. 28.

Completed ☐

Install the VC/m Routines

The VC/m routines are supplied in a routine save file. These need to be installed using the appropriate routine restore utility for your M implementation.

More detailed information can be found on p. 31.

Completed ☐

Run the Installation Script

To complete the software installation, run the VC/m installation program and select the options for your M implementation and operating system.

More detailed information can be found on p. 33.

Completed ☐

Check the Installation

Start VC/m to check that the software has been installed correctly.

More detailed information can be found on p. 35.

Completed ☐

Set Up the Browser Interface

If the browser interface is to be used with the system, configure the web server and then set up the interface to allow this.

More detailed information can be found on p. 37.

Completed ☐

Set Up Backup Procedures
Make sure your valuable data is backed up regularly.

More detailed information can be found on p. 51.

Completed ☐

Essential VC/m Configuration Settings

Once VC/m has been installed, a number of VC/m master files must be set up before it can be used. The following is a suggested procedure to follow for doing this.

Check the Component Type Table

The component type table specifies the sequence in which components of each type are transferred by VC/m.

> ***Warning: If this node is not set up for each component type which you use, VC/m will ignore components of that type. This may result in the possible loss of integrity of your software! If you are in any doubt, please contact George James Software for support and advice.***

More detailed information can be found on p. 66.

Completed ☐

Set Up User Accounts

Using User maintenance, set up a user account for each of the users.

A user account will have been created automatically for the user who first installed VC/m. This user should set up accounts for other users.

More detailed information can be found in the VC/m Reference Manual.

Completed ☐

Set Up Systems

If your software comprises more that one logical system, you may want to group objects by system. Use System maintenance to define these. They can be linked to locations later when the locations are set up.

More detailed information can be found in the VC/m Reference Manual.

Completed ☐

Identify or Create Physical Areas

If they do not already exist, you should create the physical areas that you will be managing using VC/m.

You will probably also need to create new database files /data-sets for the library.

Completed ☐

Set Up the Library

The main VC/m repository is stored in a location which maps to a physical location with a storage format of L or vss. This is used to store unchangeable master copies of each object version. You are advised to define a multi-version location which maps to this physical location.

More detailed information can be found in the VC/m Reference Manual and on p. 83.

Completed ☐

Set Up Other Locations

Use Location maintenance to set up all the locations that you require and map each one onto the physical locations.

More detailed information can be found in the VC/m Reference Manual.

Completed ☐

Set Up Location Classes

If required, group some or all of the logical locations into location classes. This is done using Location Class maintenance.

More detailed information can be found in the VC/m Reference Manual.

Completed ☐

Set Up Transfer Routes

Use Transfer Route maintenance to set up all the transfer routes which are required, with the appropriate function codes, status changes, access codes and dependencies.

If the character-based interface is used to register objects, you will need to define at least one transfer route from NEW so that newly-created objects can be transferred.

More detailed information can be found in the VC/m Reference Manual.

Completed ☐

Set Up Change Request Types

Use Change Request Type maintenance to set up at least one change request type.

More detailed information can be found in the VC/m Reference Manual.

Completed ☐

Additional VC/m Configuration Settings

VC/m provides many options for customizing the way in which it operates for each specific situation. The following options should be set up if required.

Set Up Platform Specific Settings

The Environment item under Properties within the Setup folder of the browser interface (also Screen 3 of the Properties option in terminal-mode VC/m setup) allows you to enter settings which are required on specific operating systems or M implementations.

Completed ❑

Component Date /Time Stamps

When binary and text components are transferred, it is possible to preserve the modification date /time stamp.

More detailed information can be found on p. 67.

Completed ❑

Define Settings for Component Types

Certain component types require callouts or specific settings. A callout may be required, for example, to compile the components after transfer.

More detailed information can be found on p. 70 - 75.

Completed ❑

Define Transfer and Component Driver Callouts

Setup, Properties, Transfers in the browser interface (or Properties option in VC/m terminal-mode setup) enables you to define a number of hooks where your own code can be invoked. You can use these to perform special processing such as updating indexes, menus, invoking custom installation procedures, etc.

More detailed information can be found in the VC/m Reference Manual.

Completed ❑

Define Filenames for Sequential File Locations

For each sequential location, the naming conventions used for the files can be defined. If they are not defined explicitly, they will default to the location code suffixed with a three-digit sequence number.

Setup, Properties File Locations in the browser interface (Transfer Files option in VC/m terminal-mode setup) can be used to define explicitly the prefix and ranges for each sequential file location.

More detailed information can be found in the VC/m Reference Manual.

Completed ❑

Define Audit Trail Message Settings

Setup, Properties, Audit Trail in the browser interface (screen 5 of the Properties option in VC/m terminal-mode setup) enables you to define the message category of each type of audit trail message. Message categories themselves are defined in the browser interface under Setup, Message Categories.

More detailed information can be found in the VC/m Reference Manual.

Completed ❑

Set Up VC/Live Locations
Optionally, VC/live can be used on remote production systems.

Completed ☐

Define Variant Settings
It is possible to define the default value for the variant code when a new object is created. It is also possible to set validation so that variant codes are only allowed if they match an existing system code. See Setup, Properties, Objects in the browser interface.

Completed ☐

Define Change Request Callouts
The Properties option in VC/m Set-up enables you to define a number of hooks where your own code can be invoked. You can use these to perform special processing such as validation of change request codes.

More detailed information can be found on p. 102.

Completed ☐

Set Up Automatic Allocation of User IDs
If you use the terminal-mode interface and your host operating system does not automatically provide a user ID for VC/m to use, or all your users log in with the same user ID, it is possible to externally define a user ID for a discrete terminal session.

More detailed information can be found on p. 106.

Completed ☐

Customize Menus and Set Access
User access levels can be defined for the VC/m menus and menu options. The menus themselves can also be customized.

More detailed information can be found on p. 105.

Completed ☐

Define Default Values for Functions
Screen 4 of the Properties option in VC/m Set-up contains default values for a number of terminal-mode functions (Archive and Purge, Compare, Installation and Automatic Release). These can be set up if required.

More detailed information can be found in the VC/m Reference Manual.

Completed ☐

Define Other Settings
There are many settings which can be defined using the Properties screens in VC/m Set-up. You are advised to check through these for anything else which may be relevant to your installation.

More detailed information can be found in the VC/m Reference Manual.

Completed ☐

Define Device Types
You may need to set up the device characteristics of terminals and printers that you plan to use with VC/m's terminal-mode interface. The Device Types menu option in VC/m Set-up enables you to do this.

More detailed information can be found in the VC/m Reference Manual.

Completed ☐

Define Output Devices

For each printer or other output device which will be used with VC/m's terminal-mode interface, create an entry in the printer file (using the Printers option in the VC/m Set-up menu).

More detailed information can be found in the VC/m Reference Manual.

Completed ☐

Define Terminals

For each terminal which will be used with VC/m's terminal-mode interface and is not of the DEFAULT type, create an entry in the terminal file (using the Terminals option in the VC/m Set-up menu).

More detailed information can be found in the VC/m Reference Manual.

Completed ☐

Define an Interface to the Development Tool or Tools

VC/m gives the facility to interface to a greater or lesser extent with the editor which is used by developers. Some interfaces also allow you to control their precise behavior using various settings.

More detailed information can be found on p. 112.

Completed ☐

VC/live Installation Checklist

The VC/live software is the same as the standard VC/m software, although a different license is required.

Install M

If you have not already done so, install M on the machine which is to be the VC/live server. If this process is not familiar, please consult George James Software for help.

Completed ☐

Install the VC/live Program Files

Copy the files from the CD or zip file to the installation directory.

More detailed information can be found on p. 27.

Completed ☐

Prepare the M Environment

Set up the namespaces or UCIs which are required for the VC/live software and library. Set up mappings to the VC/live routines and globals for each area where VC/live will be used. If your M implementation supports global protection, you need to ensure that all VC/live globals have the correct protection attributes.

More detailed information can be found on p. 28.

Completed ☐

Install the VC/live Routines

The VC/live routines are supplied in a routine save file. These need to be installed using the appropriate routine restore utility for your M implementation.

More detailed information can be found on p. 31.

Completed ☐

Run the Installation Script

To complete the software installation, run the VC/live installation program and select the options for your M implementation and operating system.

More detailed information can be found on p. 32.

Completed ☐

Check the Installation

Start VC/live to check that the software has been installed correctly.

More detailed information can be found on p. 35.

Completed ☐

Set Up the Browser Interface

If the browser interface is to be used with the system, configure the web server and then set up the interface to allow this.

More detailed information can be found on p. 36.

Completed ❑

Set Up Backup Procedures
Make sure your valuable data is backed up regularly.

More detailed information can be found on p. 51.

Completed ❑

VC/live Configuration Settings

Once VC/live has been installed, a number of VC/live master files must be set up before it can be used. The following is a suggested procedure to follow for doing this.

Note: A VC/live implementation has no systems or change requests.

Check the Component Type Table

The component type table specifies the sequence in which components of each type are transferred by VC/live.

> ***Warning: If this node is not set up for each component type which you use, VC/m will ignore components of that type. This may result in the possible loss of integrity of your software! If you are in any doubt, please contact George James Software for support and advice.***

More detailed information can be found on p. 66.

Completed ☐

Set Up User Accounts

Using User maintenance, set up a user account for each of the users.

A user account will have been created automatically for the user who first installed VC/live. This user should set up accounts for other users.

More detailed information can be found in the VC/m Reference Manual.

Completed ☐

Identify or Create Physical Areas

If they do not already exist, you should create the physical areas that you will be managing using VC/live.

Completed ☐

Set Up Locations

Use Location maintenance to set up all the locations that you require and map each one onto the physical locations.

The names of the production locations must exactly match the names used on the VC/m server.

More detailed information can be found in the VC/m Reference Manual.

Completed ☐

Set Up Transfer Routes

Use Transfer Route maintenance to set up all the transfer routes which are required, with the appropriate function codes, status changes, access codes and dependencies.

More detailed information can be found in the VC/m Reference Manual.

Completed ☐

Define Installation Settings

If the install process is going to be used to install files received from remote locations, the ‘install from’ and ‘install to’ locations must be specified in each M routine location where the install function will be used.

Switch to each M routine directory in turn, and use the Install option in VC/m Set-up to define the ‘from’ and ‘to locations’.

More detailed information can be found in the VC/m Reference Manual.

Completed ☐

Set Up Additional Settings

Set up any of the additional settings listed for VC/m which may be required.

More detailed information can be found in the VC/m Reference Manual.

Completed ☐

Define Output Devices

At least one printer or other output device must be defined which will be used for the VC/live installation report. Create an entry in the printer file (using the Printers option in the VC/live Set-up menu).

More detailed information can be found in the VC/m Reference Manual.

Completed ☐

Upgrade Checklist

Back Up your System
Before commencing any upgrade procedure, take a full backup of your whole system.

Completed ❑

Install the VC/m Program Files
Copy the files from the CD or zip file to the installation directory.

More detailed information can be found on p. 27.

Completed ❑

Install the VC/m Routines
The VC/m routines are supplied in a routine save file. These need to be installed using the appropriate routine restore utility for your M implementation.

More detailed information can be found on p. 31.

Completed ❑

Run the Installation Script
To complete the software installation, run the VC/m installation program and select the options for your M implementation and operating system.

More detailed information can be found on, p. 32.

Completed ❑

Check the Installation
Start VC/m to check that the software has been installed correctly.

More detailed information can be found on, p. 35.

Completed ❑

Set Up New Features
Consult the release documentation and this manual for details of how to configure the new features which you require.

Completed ❑

Glossary

absolute dependency	a particular object version must be active at a dependent location
access code	a code which determines which users have access to an item
activate	set the status to active for an object version at a location
active	a status of an object version at a location which means it can be used as the source for a transfer and to satisfy location dependencies
ancestor	an object version from which another version was created
change request	the specification of a change which is to be made to the application, including the objects and the reason
check in	transfer the master copy of an object version into the library
check out	transfer the master copy of an object version out of the library
component	a physical unit of an application, for example a file or a routine A component is **registered** to VC/m and **belongs** to an object version.
component type	a code which specifies the kind of component, e.g. binary file, text file or M routine
concurrent development	work on two different aspects of an application which takes place at the same time
controlled location	a location which VC/m expects to have full knowledge of
copy	transfer an object version to a new location and leave a copy at the first location
date /time stamp	the unique date and time which VC/m records for each version of a component
de-activate	remove the active status for an object version at a location
dependency	a condition on a transfer
dependent location	a condition on a transfer in which an object version must be active at the dependent location in order for the transfer to be valid
displace	remove an object version from a location and replace it with a newer version
error	a status of an object version at a location which means it is not a complete or integral copy
exist	for a component, it is physically present at the location. For an object version, all its components are physically present and it is recorded in VC/m's database
function code	a code which specifies the kind of transfers for which a transfer route is used
inactive	another word for error
the library	the central storage location for all objects, used to store reference copies of the objects, as well as copies of previous versions
library location	a logical location which maps to the library
loaded	another word for registered
location	a conceptual place where an object version is located, e.g. live, development, testing, tested A location **maps** onto one or more physical locations.
location class	a group of locations
logical location	another name for a location

master a status of an object version at a location which means it is the primary copy of the object

module a template for the creation of a new object and its components

move transfer an object version to a new location and then remove it from the first location

multi-version location a location which can hold more than one version of an object at the same time

object a logical unit of an application combining an object base and variant
An object **belongs** to a system.

object base a grouping which forms a unitary part of an application, synonymous with an object when there is only one variant

object version a logical grouping of one or more physical components at a specific stage of development

obsolete another word for error

physical location the physical place where an object actually resides, e.g. an operating system directory or an M namespace
A physical location is **declared** to VC/m.

predecessor another word for ancestor

program another name for an object

relative dependency a particular object version or a later one must be active at a dependent location

registered a component belongs to an object version which is active at a location

remote system a system which is physically separate from the VC/m system, i.e. without a network connection

sandbox a physical development area which is unique to one developer rather than being shared

single-version location a location which can only hold one version of an object at any one time

status the status of an object version at a location is master, active or error

status date a date which can be used as a condition on a volume transfer

storage format a code which determines the way in which components are stored in a physical location

successor an object version which was created from another version

system a group of objects which are installed together to form a working set of software
A system is **installed** at a location.

transfer move or copy an object version between two locations

transfer route a path that an object is moved along

uncontrolled location a location which VC/m does not expect to have full knowledge of

variant a high-level design difference in an object which can exist in parallel with other variants over an extended period of time

version an instance of an object at a particular stage of development
also used for an instance of a component at a particular stage of development

version number an number which indicates that a small change has been made to an object

Index

A

Absolute dependency 90, 171
Access codes 12, 14, 18, 104, 171
Active status 11, 13, 17, 18, 82, 90, 171, 172
Ancestor 8, 16, 18, 132, 133, 171
Archive ..... *See* Purge
Audit trail 4, 89
Automatic transfers 21
AUTOT function code 89

B

Backing up 51
Baseline creation 3
Binary file 63
Bulk transfer 89

C

Caché 64
Cancel 95, 131
Change request 3, 7, 19, 22, 102, 171
Change request status 7, 22
Change request type 7, 22
Change request validation 102
Check-in 3, 9, 89, 133
Check-out 3, 4, 9, 89, 124, 125, 132
Component 5, 6, 8, 16, 17, 63, 66, 76, 82, 113, 124, 125, 126, 171, 172
Component driver 5, 63, 76
Component type 5, 63, 66, 76, 124, 125, 126, 161, 168, 171
Component type table 63, 66
Component type, customized 95
Concurrent development 3, 132, 171
Controlled location 8, 16, 82, 125, 131, 171
Copy 12, 16, 63, 76, 87, 95, 171, 172
Copy only if changed 95

D

Date /time stamp 95, 113, 124, 125
Defaults 8, 90, 163, 164
Deletion 11, 16, 24, 126, 131
Dependency ..... *See* Transfer route dependency
Dependent location 13, 18, 90, 171, 172
Device type 111, 164
Diff 95, 143, 164
Displacement 16, 132, 133, 171
DOS 143
DSM 55

E

Editor 95, 113, 124, 125, 126
EMPTY location 90
Error status ..... *See* Inactive status

F

Force install ..... *See* Install
Force rollback ..... *See* Rollback
Function code 12, 89, 171
Function defaults 164

G

Getting around VC/m 24
Global access and protection 58, 159, 166
Global protection ..... *See* Global access and protection
Globals 33, 51, 159, 166
Greystone M ..... *See* GT.M
GT.M 56, 142

H

Hot backups 4, 95

I

IN function code 89
Inactive status 11, 171, 172
Install function 89, 95, 139, 164
Installation of VC/m 35, 66

L

Library, the 3, 4, 9, 13, 51, 77, 82, 90, 171
Loading components 128
Location 8, 171
Location class 8, 171
Logical location ..... *See* Location

M

M global 66, 95
M implementation 33, 78, 159, 166
M routine 63, 66
M routine directory location 63, 77
M/SQL 64, 114, 118
Main menu 23, 106, 108
Master status 9, 11, 16, 17, 18, 19, 83, 113, 124, 133, 172
Matrix of version by location 4
Menu access 104
Menus 14, 24, 104, 105, 108, 110, 171
Merge 132, 133
Micronetics M ..... *See* MSM
Module 15, 172
Move 12, 16, 172
MSM 57, 58, 143
Multi-version location 8, 16, 17, 95, 133, 172

N

Navigation ..... *See* Getting around VC/m
NEW location 88

O

Object 6, 172
Object base 6, 10, 172
Object version 6, 172

Obsolete status ..... *See* Inactive status
OUT function code 89
Output device 165, 169
Overdue checked-out objects report 4

P

Parallel development ..... *See* Concurrent development
Physical location 8, 63, 87, 171, 172
Physical location type 78
Printer 165, 169
Program ..... *See* Object
Purge 95, 164

R

Relative dependency 90, 172
Release 3, 4, 89, 164
Release control sheet 4
Release date 13
Remote locations 95
Rollback 4, 138
Routine editor ..... *See* Editor

S

Sequential file location 8, 77, 163
Single-version location 8, 16, 18, 133, 172
Skeleton routine generator 15, 124
Starting VC/m 23
STATUS function code 89
Successor 172
Summary of transfers 4
System 10, 172

T

Take on 128
Terminal 106, 111, 164, 165
Text file 63
Tied location 22
Transfer file ranges 163
Transfer route 12, 13, 14, 16, 17, 18, 88, 89, 90, 95, 104, 162, 171, 172
Transfer route dependency 13, 18, 59, 89, 90, 128, 171

U

Uncontrolled location 8, 16, 82, 172
UNIX 106, 143
User exits 163, 164
User ID ..... *See* Users
Users 14, 18, 104, 106, 131, 164

V

Valid transfer 18, 22
Variant 6, 10, 172
VC/live 4
Version 6, 172
Version suppression ..... *See* Default for version suppression
VMS 106

X

XFER function code.....89