GEORGEJAMES
SOFTWARE

VC/m

VC/m Reference Manual
Version 3.2

Notices

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\ VCmReferenceManual.doc	30 August 2005	Gail Treves-Brown	
1	VCmReferenceManual.doc	16 July 2009	John Murray	VCmManual.doc/1.6
2	VCmReferenceManual.doc	14 September 2009	John Murray	VCmManual.doc/1.7

VC/m Reference Manual version 3.2

Contents

1 VC/m's Browser Interface Reference

1.1 The Browser User Interface

The Main Window

At the top of the main VC/m window is the menu bar. Below this is the select toolbar, which has 2 sections: the selection criteria and the work list.

Below the menu and toolbars, the main window has three panels. The central section has a left and right-hand panel, similar to the Windows Explorer interface. The left-hand panel is the folders panel and the right-hand panel is the main panel. The third panel, at the bottom, is the audit messages panel.

The panels can be resized by dragging. The folders panel and the audit messages panel can be shown or hidden using the options on the View menu.

Selection Criteria

Selection criteria filter the items which are displayed in the main panel for the object, component and change request views of a location.

The selection criteria are entered in the text box at the left-hand end of the select toolbar. The ten most recently used criteria are available from the drop-down list next to this.

The selection criteria can be specified in a number of standard ways:

Method	Example
Select with a wildcard	ABC*
Select with a single-character wildcard	ABC2?
Select and de-select with a wildcard	AB*;'ABC*
Multiple independent criteria	ABC*;BBC*

For objects, a special selection notation enables the highest versions at the location to be selected. Instead of naming the version number (e.g. A*/1.3), use the letter 'h' (e.g. A*/1.h). Note that this refers to the highest version which is active at that location. There may be other higher versions which are not at that location, for example, ABC/1.3 may be the highest version in LIB, although ABC/1.4 is under development in DEV.

Object selection criteria can also optionally have a suffix of '*@location*' where *location* is a location code at which the object(s) must be active in order to qualify for the selection or de-selection.

Clicking the 'Select' button or pressing the return key will cause the specified selection criteria to be applied to the items which are displayed in the main panel. Clicking the 'Select' button will also refresh this screen.

The status of the items which are displayed can also be controlled using the filter buttons to the right of the text box. When selected (depressed), items of the corresponding status will be included in the selection. Hover the pointer over a button for a description.

Double-clicking a filter button toggles its state **and** initiates the search.

Selection criteria may be copied from one folder to another of the same type by holding the Control key down when clicking on the target folder.

The Work List

The work list is at the right-hand end of the select toolbar. It allows a selection of items to be grouped and then have an operation applied to them. It can be used to add components to an object version or to add object versions to a change request.

The work list button has a list icon and shows the number of items in the list. Hovering over the button displays the items currently in the work list.

To add an item to the work list, first select the item in the main panel. Then drag the item to the work list button on the select toolbar. Alternatively, select 'Edit /Copy to Work List' from the menu, or press Ctrl-C on the keyboard. The number of items in the work list will be updated.

To paste the items from the work list, first select the item which they are to be added to in the main panel. Then select 'Edit /Paste from Work List' from the menu, or press Ctrl-V on the keyboard.

To clear the contents of the work list, click the 'Clear' button next to the work list or select 'Edit /Clear Work List' from the menu.

Folder and Main Panels

After logging in, the folder panel displays a tree of folders (options). By default, the following folders appear at the top level of the tree:

- Locations
- Location classes
- Objects
- Change requests
- Audit trail
- Setup

This list is folders is user-definable. Access controls can also be set, and custom forms and methods can be added. (The list of locations can be restricted by setting appropriate access codes in location maintenance.)

When a folder is selected, the main panel displays information about the contents of that folder. The information which is displayed depends on the type of folder. Some folders, for example a location folder, have a choice of views which display different information in the main panel.

The main panel can be printed or print-previewed using the option from the File menu.

By default 1000 items are displayed in the main panel. This value can be altered by opening the Setup folder, then the Properties subfolder, then clicking on Browser Interface.

Location Folder

When a location is selected, the main panel displays information about the location, including a description, the physical addresses and any systems which belong at the location. If a physical location is currently inaccessible (e.g. because a server is offline or a database is unavailable), it will be displayed with the name struck through.

Each location folder can be opened out to show the next level of the tree. The next level has at least three folders (views): objects, change requests, and one for each component mask the location has. Each of the views displays the contents of the location in a different way.

Object Folder

General View

When the top-level object folder is selected, all the object versions which are in the VC/m database and which match the selection criteria are displayed in the main panel. The selection criteria can be changed on the select toolbar.

The icon for each object version is colored to show the status of the object version.

Color	Status	Explanation
Green		Not used for this view
Amber	Registered	An active version
Red		Not used for this view
Gray	Superseded	A later version exists with this as the ancestor
White		Not used for this view

Location View

When the object folder is selected for a location, all the object versions which are at that location and which match the selection criteria are displayed in the main panel. The selection criteria can be changed on the select toolbar.

The icon for each object version is colored to show the status of the object version at that location.

Color	Status	Explanation
Green	Checked out	A master version at a location where is can be edited
Amber	Registered	An active version which cannot be edited at this location
Red	Error	Usually caused because a transfer failed
Gray	Superseded	A later version exists here or elsewhere with this as the ancestor
White	Empty	The object version has no components at this location

Change Request Folder

General View

When the top-level change request folder is selected, all the change requests which are in the VC/m database and which match the selection criteria are displayed in the main panel. The selection criteria can be changed on the select toolbar.

The icon for each change request is colored to show the status of the change request.

Color	Status	Explanation
Green		Not used for this view
Amber	Registered	At least one object version belongs to the change request
Red		Not used for this view
Gray		Not used for this view
White	Empty	No object versions belong to the change request

Location View

When the change request folder is selected for a location, all the change requests which are at that location and which match the selection criteria are displayed in the main panel. The selection criteria can be changed on the select toolbar.

The icon for each change request is colored to show the status of the change request at that location.

Color	Status	Explanation
Green	Checked out	At least one object version in the change request is checked out at this location
Amber	Registered	At least one object version in the change request is active at this location, but none of the object versions are checked out to it
Red		Not used for this view
Gray		Not used for this view
White		Not used for this view

Component Folder

The component view is only available for a location. There is no single component view for a location. Instead, there is a branch for each of the component masks which are defined for the location. To see a component view, open the component tree in the folder panel and select one of the branches. All the components with that component mask, which are at that location and which match the selection criteria are displayed in the main panel. The selection criteria can be changed on the select toolbar.

The icon for each component is colored to show the status of the component at that location.

Color	Status	Explanation
Green	Checked out	A master version at a location where is can be edited
Amber	Registered	An active version which cannot be edited at this location
Red	Unregistered	Not registered to any object in VC/m at this location
Gray	Superseded	A later version exists here or elsewhere with this as the ancestor
White		Not used for this view

Audit Trail Panel

After each operation, the audit trail panel displays the message which is written in the audit trial. These scroll when the space runs out.

If the message is a persistent one, double-clicking on any part of the message line except the Item column opens a window showing additional information. Double-clicking on the entry in the Item column opens the corresponding Object or Change Request property dialog.

1.2 Operations

Object Transfers

Before a transfer can take place, a transfer route must first have been set up. This route defines the exact processing which will take place.

A transfer can be invoked for a single object version, a change request, or a multiple selection. When the change request is used, all the relevant object versions on the change request will be transferred.

A transfer can be initiated in either of two ways.

The first method is to select the item(s) in the main panel and then drag and drop onto the 'to location' area in the folder panel. A dialog will appear.

Alternatively, a transfer can be initiated by using the menu. Select the item(s) in the main panel and then select 'Transfer' from the menu or from the right-click context menu off the selection. Choose the menu option for the kind of transfer to be performed. The same dialog will appear.

The dialog for an object transfer has the following fields:

- Object

  Read-only display of the object version.

- Components

  Read-only display of the components of this object version.

- Systems

  The systems to which the object version belongs. Depending on the configuration options for the installation, this list may or may not be altered.

- Function

  The type of transfer to be performed. The displayed value can be altered by selecting from the drop-down list.

- From

  The location code from which the change request or object version is to be transferred. The displayed value can be altered by selecting from the drop-down list.

- To

  The location code to which the change request or object version is to be transferred. The displayed value can be altered by selecting from the drop-down list.

- Change request

  For transfers which increment the version number, at least one change request must be specified. Click the 'Add' button to select from a list of existing change requests, or to add a new one. Alternatively, type the change request code into the first text field and click 'Add'. To remove a change request from the list, highlight the change request and click 'Remove'.

The dialog for a change request transfer has the following fields:

- Change request

  Read-only display of the change request code.

- Components

  Read-only display of the object versions belonging to this change request.

- Function

  The type of transfer to be performed. The displayed value can be altered by selecting from the drop-down list.

- From

  The location code from which the change request or object version is to be transferred. The displayed value can be altered by selecting from the drop-down list.

- To

  The location code to which the change request or object version is to be transferred. The displayed value can be altered by selecting from the drop-down list.

- Transfer mode

  Select the transfer mode. The default value is 'interactive'.

Option	Description
Batch Transfer	Transfers all the objects, reporting in the audit trail panel after each one. Continues after any error which may be encountered.
Interactive	Transfers all the objects, reporting in the audit trail panel after each one. Stops and consults the user interactively if an error occurs.
Background	Calls a background job to transfer all the objects. Transfers will only be displayed if the audit trail panel is refreshed. Continues after any error which may be encountered.

For transfers which do not increment the version number, the fields for 'Function', 'From' and 'To' must be filled in. Where possible, VC/m offers a default value. Even when drag-and-drop is used, these values can be altered.

Once the dialog has been completed, click 'OK' to make the transfer. Unless background mode was chosen the audit trail messages for the transfer and for any auto-transfers which are triggered will appear in the audit trail panel at the bottom of the window. For background mode transfers, review audit messages under the folder in the tree.

Object Creation and Single Component Registration

A new object is created by transferring a component along a valid route. This may be a route which is part of the standard development cycle, or it may be a route which is specifically for this purpose.

Open the 'component' view of the location where the new component is physically present. Unregistered components are shown in the main panel with a red icon.

Select the component and transfer it to the 'to location' by using drag-and-drop or the 'Transfer' option on the menu.

Exceptionally, the object field in the transfer dialog is not read only. A default name for the object is provided. This can be altered prior to clicking 'OK'. When the user clicks 'OK', the new object is created and the component is registered to it as part of the same operation.

If more than one component is to be added to an object version, the work list must be used. First create the object and register the first component using the procedure given above. Then follow the procedure for adding components to an object using the work list (see below).

Addition and Removal of Components

Components can be added to or removed from an object version only while it is checked out.

To add a component to an object version, select the component in the main panel and add it to the work list. If more than one component is to be added, the others can also be added to the work list. Once the list has been completed, select the object version in the main panel. Paste the items from the work list onto the object version.

To remove a component from an object, select the object view. In the main panel, open out the object and select the component. Then select 'Delete Component' from the menu or press the delete key.

To physically delete a component, follow the above procedure to unregister it from the object version. Then select the component again and repeat the procedure to delete it.

Change Request Entry and Deletion

A change request is used to group objects together so that they can easily be moved as one unit. When an object selector prompt is offered, a change request can be entered instead, prefixed with @.

Change Request Entry

To set up a new change request, select 'Change Request /New …' from the menu. Alternatively, the same dialog can be reached from the transfer dialog by clicking 'Add' and then 'New'.

To edit an existing change request, select 'Change Request /Open …' from the menu. Highlight the change request in the list and click on 'Select'. The change request dialog will open.

The dialog has the following fields:

- Change request

  Enter the unique identifier for the project, task, error reference, etc. If the change request type is set up to allocate numbers automatically, leave this field blank.

  Note: The wording of this label may vary depending on what is entered in the field for Prompt text under Change request type maintenance.

- Type

  This field is used to categorize change requests by type for reporting purposes. Select a type from the drop-down list. These types are set up in the Change Request Type master file.

- Title

  Enter a descriptive title for the change request.

- Description

  Details about the change request can be entered here.

- Owner user id

  Enter the user who is responsible for this change request. The default value is the current VC/m user.

- Planned end date

  Enter the date by which the work on the object versions is planned to be completed. This is the estimated date at which the objects can be returned to the library. It is used by the 'Overdue Checked-out Objects' report. The default value is today's date.

Click 'Save' when all the information has been entered. If the change request number is allocated automatically, it will now be displayed.

Click 'Close' to finish editing the change request.

Change Request Deletion

To delete a change request, highlight the change request, then select 'Change Request /Delete …' from the menu. Alternatively, press the <Delete> key.

Deleting a change request does not delete its member objects.

Addition and Removal of Object Versions from a Change Request

Object versions can be added to or removed from a change request at the time of a transfer. They can also be added or removed when a transfer is not being invoked, as described below.

To add an object version to a change request, select the object version in the main panel and add it to the work list. If more than one object version is to be added, the others can also be added to the work list. Once the list has been completed, select the change request in the main panel. Paste the items from the work list onto the change request.

To remove an object version from a change request, select the change request view. In the main panel, open out the change request and select the object version. Then select 'Edit /Remove Object from Change Request' from the menu or press the Delete key.

1.3 Reports and Utilities

All the reports are presented in Excel format. They can be reached by selecting 'View' and then the appropriate option from the menu. Fill in the dialog and click 'OK'. When prompted, choose between opening and saving the Excel spreadsheet.

SQL Queries

VC/m includes Caché class definition files for the VC/m database. This allows Caché users to define their own SQL queries on the database.

To set up the facility:

1 Load the VC/m class definitions into the VC/m namespace on the VC/m server.

Start a terminal session in the VC/m namespace. Enter the following command:

```
set ok=$system.OBJ.LoadDir(fullpath,"c")
```

where '*fullpath*' is the full path of the directory where VC/m is installed

2 On the PC where the query is to be set up, define an ODBC data source which uses the InterSystems ODBC driver to connect to the VC/m namespace on the VC/m server. If the driver is not installed on the PC, it needs to be installed from the Caché installation disk. Ideally the same version should be used as the version where VC/m is running.

Verify that basic connectivity works by using the Test Connection button on the ODBC setup dialog.

Note: You should ensure that your Caché installation is secured against unauthorised user of ODBC access.

To run an SQL query:

1 Submit the query through the ODBC data source set up above. For example, in Excel 2002, you can use the Data\Import External Data\New Database Query… feature (which uses Microsoft Query).

An example query to list all the objects and change requests checked in to the location NEXT between 1st July and 30th September 2003:

```
SELECT oval.objectVersion, ov.primaryChangeRequest,
oval.lastUpdateDateTime

FROM SQLUser.ObjectVersion ov, SQLUser.ObjectVersionAtLocation oval
WHERE  {fn CONCAT(ov.object,{fn CONCAT('/',{fn CONCAT(ov.variant,{fn
CONCAT('.',ov.version)})})})} = oval.objectVersion

AND oval.location='NEXT'

AND oval.lastUpdateDateTime>20030701000000

AND oval.lastUpdateDateTime<20031001000000
```

Notes on the syntax:

The FROM clause assigns aliases of 'ov' and 'oval' respectively to the tables 'ObjectVersion' and 'ObjectVersionAtLocation'.

The WHERE clause joins these two tables. This clause is relatively complex because the 'ObjectVersion' table stores the object base, object variant and object version separately rather than in a single column as the 'ObjectVersionAtLocation' table does.

Note: This join is needed in order to access the primaryChangeRequest column in the 'ObjectVersion' table.

The first AND clause restricts the output to objects which are present in the location NEXT.

The second and third AND clauses check the date /time when each object was most recently transferred into NEXT. This column, oval.lastUpdateDateTime, is presented as a string rather than as an SQL timestamp, but it can be treated as a large integer.

2 VC/m's Character-Based Interface Reference

2.1 Starting VC/m

Starting VC/m's Character-Based Interface

To run VC/m, log into the M system and enter the following command:

```
do ^%vc
```

or

```
DO ^%VC
```

In order to use VC/m, you must have a valid user ID set up. If you are using Windows 95 or later, UNIX /Linux or OpenVMS, the account name that you logged in with is usually used. If you are not logged into your operating system, you are prompted for a user ID. In all cases, the user ID and associated access levels must be set up by the VC/m system manager before you can use VC/m.

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

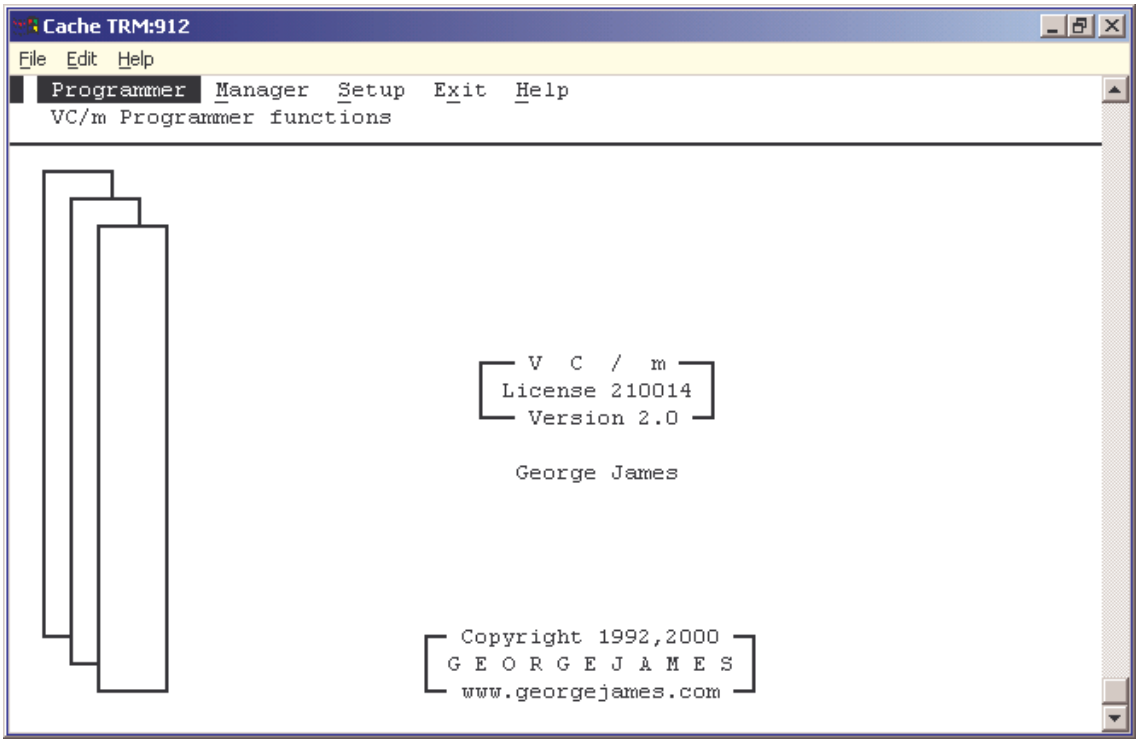


Figure 2.1 VC/m main menu screen

Starting VC/live's Character-Based Interface

To run VC/live, log into the M system and enter the following command:

```
do ^%vcl
```

or

```
DO ^%VCL
```

In order to use VC/live, you must have a valid user ID set up. If you are using Windows 95 or later, UNIX /Linux or OpenVMS, the account name that you logged in with is usually used. If you are not logged into your operating system, you are prompted for a user ID. In all cases, the user ID and associated access levels must be set up by the VC/m system manager before you can use VC/live.

You will be presented with the following screen, after prompting for your user ID if this is necessary:

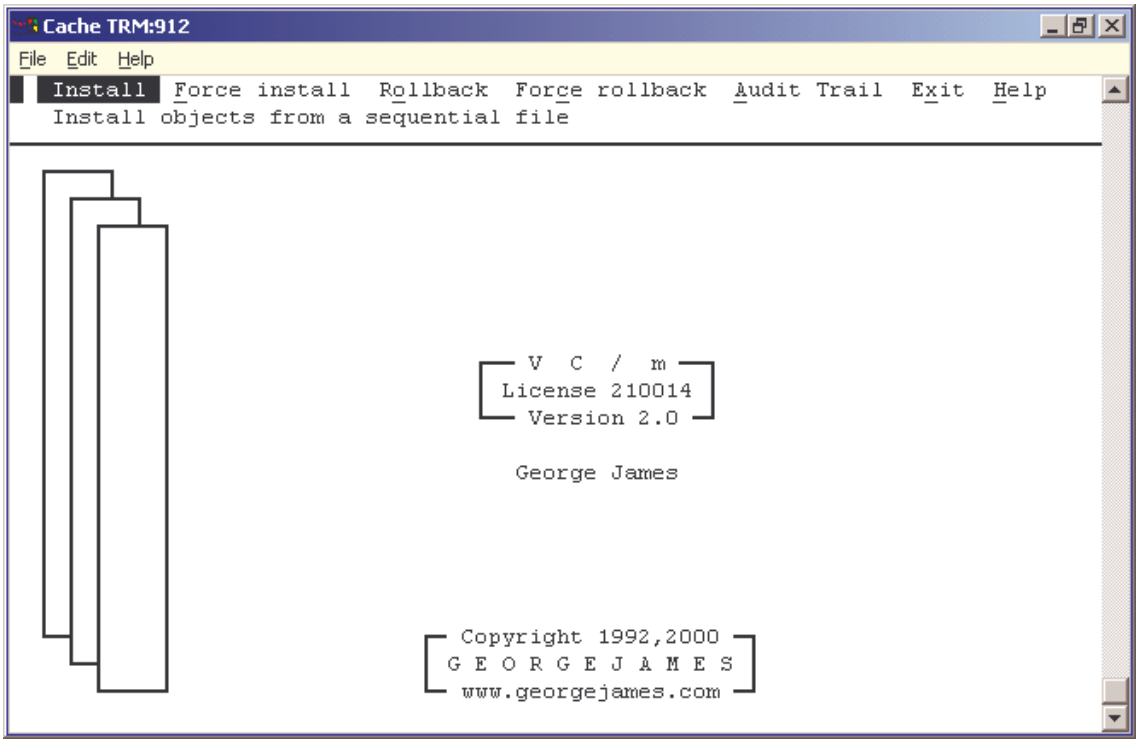


Figure 2.2 VC/live main menu screen

Other Entry Points to the Character-Based Interface

Invoking a Second-Level Menu Directly

It is possible to skip the main menu and go directly to one of the second-level menus.

For the programmer menu, enter one of the following commands at the M prompt:

```
do ^%vcp
```

or

```
DO ^%VCP
```

For the manager menu:

```
do ^%vcm
```

or

```
DO ^%VCM
```

For the set-up menu:

```
do ^%vcsetup
```

or

```
DO ^%VCSETUP
```

Invoking a Customized Menu Directly

To invoke a customized menu directly, enter the following command at the M prompt:

```
do run^%vc("^%vcmenu(""menu"")")
```

where *menu* is the name of the VC/m menu to be invoked

Invoking a Menu Option Directly

It is possible to invoke most VC/m menu options directly from the M command line, or from an operating system command file or script.

To invoke a standard or a customized menu option directly, use the following command:

```
do run^%vc("option")
```

where *option* is the routine label of the VC/m menu option to be invoked

Note: The routine label must include the ^ character as part of the label.

The following can also be used for a standard VC/m menu option:

```
DO RUN^%VC("option")
```

A current list of menu options which can be invoked directly can be found by pressing a help key (?, F1 or PF1) at the Menu /function prompt under Menu maintenance.

2.2 The Character-Based User Interface

Menus

All the menus within VC/m use a menu bar style which conforms to de-facto industry standards. Each menu has a series of options arranged horizontally on the first line of the screen. If there are more options than will fit on the screen, the menu will scroll horizontally as the cursor is moved across.

The default option is highlighted in inverse. The cursor keys (left-arrow and right-arrow) can be used to move the highlight from one option to the next. If the last option on the menu is highlighted and the right-arrow key is pressed, the first option will be highlighted, and vice versa. The second line of the screen contains a description of the menu option which is highlighted.

There are three ways to select an option from the menu bar:

- Position the highlight over the desired option and press the <return> key.
- Position the highlight over the desired option and press the down-arrow key.
- Press the key corresponding to the underlined character in the desired option (shortcut key).

Every menu bar has two standard options: Exit and Help. If Exit is selected, VC/m goes back to the previous menu. The up-arrow key has the same effect as selecting the Exit option. If Help is selected, it displays a page of help text which describes the functions that are available.

For further information, the IBM SAA CUA (System Application Architecture - Common User Access) specification contains a very comprehensive description of menu bars.

Input Fields

All input fields are represented by a prompt, normally to the left of the field, and a reply area enclosed by square brackets. For example:

```
Object              [                          ]
```

If the field has a default, this will be displayed in the reply area. To accept the default, press the <return> key.

To delete the contents of a field enter a single space followed by <return>. If the field is mandatory, VC/m will not allow it to be deleted.

If the field already has an entry, this can be edited by using the cursor keys to move to the appropriate place, and then re-typing. The default is over-type mode. Use <ctrl-a> to toggle between overtype and insert mode. The field contents will be deleted if the user starts typing immediately the field is reached, without moving the cursor.

Some fields have context sensitive help or a selection list. This can be invoked by pressing a help key. The help keys are ? on any keyboard, F1 on a PC keyboard and PF1 on a VT keyboard.

To go back to a previous field, press the up-arrow key.

Selection Lists

At most input fields where one or more items can be chosen, a selection window can be invoked to browse through and make a selection.

To invoke a selection window, enter a help key while in the input field. (The help keys are ? on any keyboard, F1 on a PC keyboard and PF1 on a VT keyboard.)

There are two types of selection window. The single-item selection window is used to browse through a list and select one item.

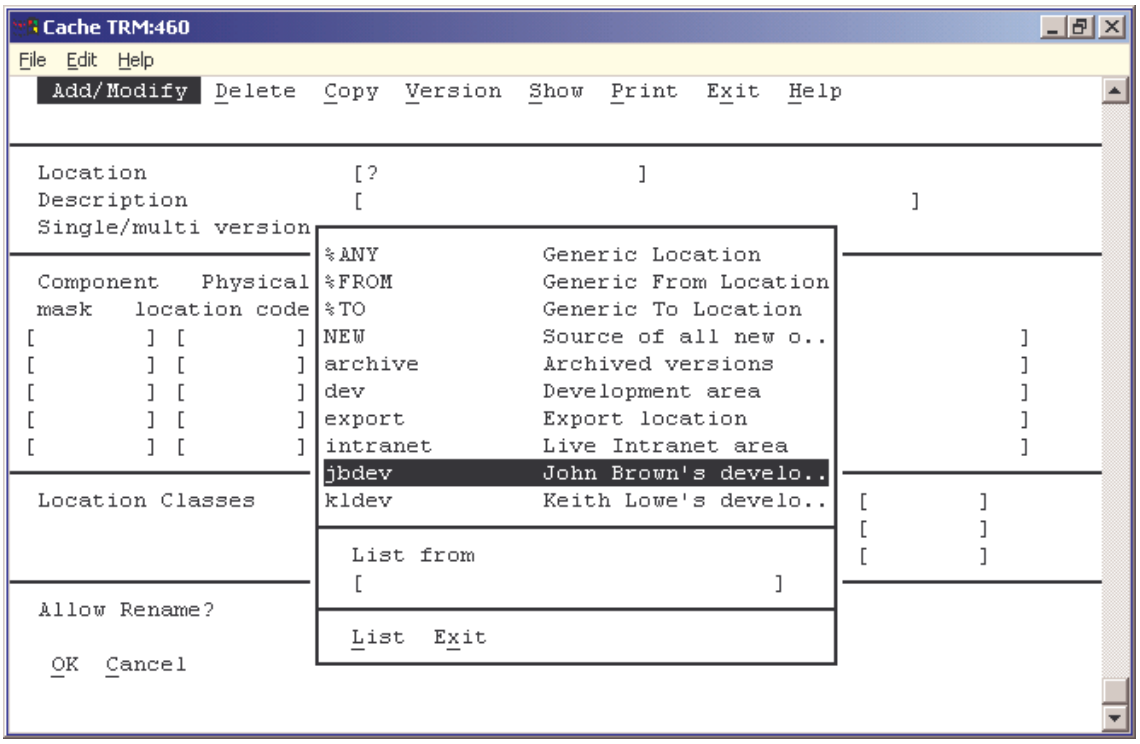


Figure 2.3 Single-item selection window

The second type of window is used when more than one item can be selected. The window contains two main panels. The left-hand panel displays the list of possible selections. The right-hand panel displays the items which are currently selected.

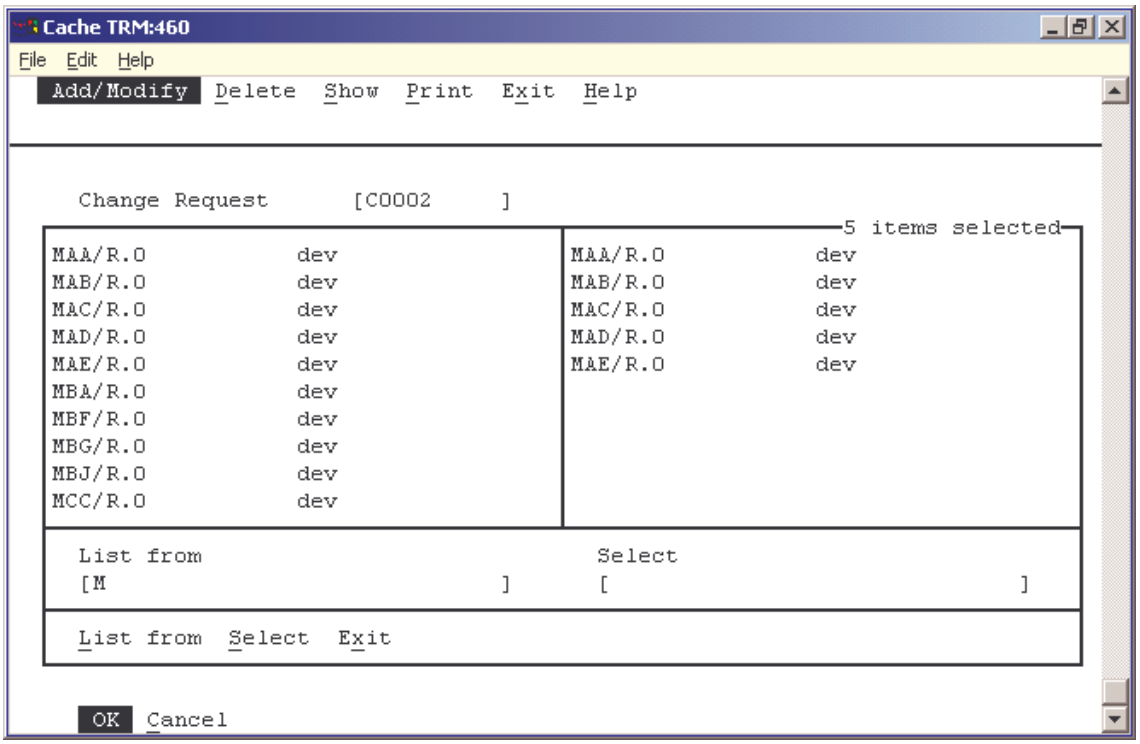


Figure 2.4 Multi-item selection window

In both types of window, the space bar or <return> are used to select the item which is highlighted. Cursor up, cursor down, page up and page down are used to scroll through the list of items.

In a single-item selection window, selecting an item closes the window and moves onto the next input field on the original screen. X (exit) can be used to exit the window without making a selection.

In a multi-item selection window, the left and right cursor keys are used to move between the two panels. When an item is selected in the left-hand panel, it is added to the list in the right-hand panel. When an item is selected in the right-hand panel, it is removed from the list. When the selection is complete, type X (exit) to return to the original screen. The cursor remains in the input field until <return> is pressed.

A multi-item window also has a Select prompt, which is reached by typing S. This enables items to be selected and /or de-selected from the right-hand list using wildcards and ranges. A minus sign before the items is used to de-select them.

More detailed information can be found on p. 27.

In both types of window, the List from prompt can be used to display the possible selections beginning from a specific point. Type L to go to the prompt, enter the characters to begin from and then <return>. Note that items are listed in the following order: numbers, punctuation characters, upper-case letters from A to Z, then lower-case letters from a to z.

The following table summarizes the keystrokes which can be used within a selection window:

Enter	Select the highlighted item
Space Bar	Select the highlighted item
Cursor Up	Scroll up the list by one line
Cursor Down	Scroll down the list by one line
Cursor Left	Move from the right-hand window to the left-hand window
Cursor Right	Move from the left-hand window to the right-hand window
Page Up	Scroll up the list by one page
Page Down	Scroll down the list by one page
L	Go to the List from prompt
S	Go to the Select prompt
X	Return to the original screen

Date Fields

Each VC/m installation is set up to accept either US format or UK format dates.

Whatever format the used to enter a date, it will be reformatted and displayed in the standard format of DD-MMM-YY.

When a 2 digit year is entered (i.e. the century is not specified), VC/m defaults the century using the 2 digit year pivot point. This value can be set on screen 6 of the Properties option in VC/m Set-up.

Valid entries for US format dates include the following:

Date	Description
T	Today
+5	Today plus 5 days (i.e. 5 days in the future)
-5	Today minus 5 days (i.e. 5 days ago)
mm/dd/[cc]yy	Day, month and 2 or 4 digit year as specified
mm/dd	Day and month as specified, current year as default
/dd	Day as specified, current month and year as default
dd-mmm-[cc]yy	Day, alphabetic month (Jan, Feb, Mar, etc.) and 2 or 4 digit year as specified
dd-mmm	Day and alphabetic month as specified, current year as default
mmdd[cc]yy	Day, month and 2 or 4 digit year as specified

Valid entries for UK format dates include the following:

Date	Description
T	Today
+5	Today plus 5 days (i.e. 5 days in the future)
-5	Today minus 5 days (i.e. 5 days ago)
dd/mm/[cc]yy	Day, month and 2 or 4 digit year as specified
dd/mm	Day and month as specified, current year as default
dd	Day as specified, current month and year as default
dd-mmm-[cc]yy	Day, alphabetic month (Jan, Feb, Mar, etc.) and 2 or 4 digit year as specified
dd-mmm	Day and alphabetic month as specified, current year as default
ddmm[cc]yy	Day, month and 2 or 4 digit year as specified

Object Fields

When an input field prompts for a list of objects or components, they can be selected or de-selected in a number of standard ways:

Method	Example
Select individually by name	ABC120/1.2
Select with a wildcard	ABC*
	J
Select by range	A-B
De-select individually by name	-ABC123/1.2
De-select with a wildcard	-ABC*
De-select by range	-A-B

If data has been entered, the prompt repeats when <return> is pressed. Continue to the next prompt by pressing <return> without entering anything at the prompt.

A special object selection notation enables the highest versions at the location to be selected. Instead of naming the version number (e.g. A*/1.3), use the letter H (e.g. A*/1.H). Note that this refers to the highest version which is active at that location. There may be other higher versions which are not at that location, for example, ABC/1.3 may be the highest version in LIB, although ABC/1.4 is under development in DEV.

Output Device Fields

An output device can only be used if it has previously been entered in the printer file under Printers in VC/m Set-up.

When an input field prompts for an output device, it is entered using the name which was specified in the printer file.

If a device has been set up in the printer file with ??? for the $IO value of the device, the actual device ID is entered at the time of printing.

For example, if a device called FILE is set up with a device ID of ???, at any Output Device prompt, the device name is input, followed by a single space and the actual device ID to be used:

```
Output device        [FILE MYFILE.TXT          ]
```

Similarly, if three question marks have been entered at any point in the open parameters field of the printer file, a value can be supplied at print time to substitute the three question marks.

For example, the open parameters contain the following:

```
Open parameters      [(file="???":mode="W")    ]
```

At print time, the file name can be specified at the Output Device field:

```
Output device        [FILE MYFILE.TXT          ]
```

2.3 Menu Structure

The following diagrams provide a quick reference guide to where each option in the main VC/m system and in VC/live can be found.

VC/m Programmer Menu

Many of the menu options also have two or four standard options below them: Show and Print, or Add /Modify, Delete, Show and Print.

Note: The VC/m menu structure is customizable and may therefore be different from what is shown.

```
Main Menu
└─ Programmer                  - VC/m Developer options
   ├─ Out                      - Check out an object
   ├─ In                       - Check in an object
   ├─ Cancel                   - Cancel an object checked-out in error
   ├─ Audit                    - Audit trail
   │  ├─ Object                - Audit trail activity by object or change request
   │  └─ Date                  - Audit trail activity by date
   ├─ Object                   - Object maintenance and component registration
   │  ├─ Add /Modify           - Add an object or modify an existing one
   │  ├─ Components            - Add or modify components of an object
   │  ├─ Move                  - Move a component from one object to another
   │  ├─ Show                  - Display details of an object or objects
   │  ├─ Print                 - Print details of an object or objects
   │  └─ Delete                - Delete an object
   ├─ Transfer                 - Transfer an object between locations
   ├─ Change Request           - Change request maintenance
   ├─ Merge                    - Merge two versions of an object
   ├─ Diff                     - Compare two objects
   └─ Find                     - Search for a string
```

VC/m Manager Menu

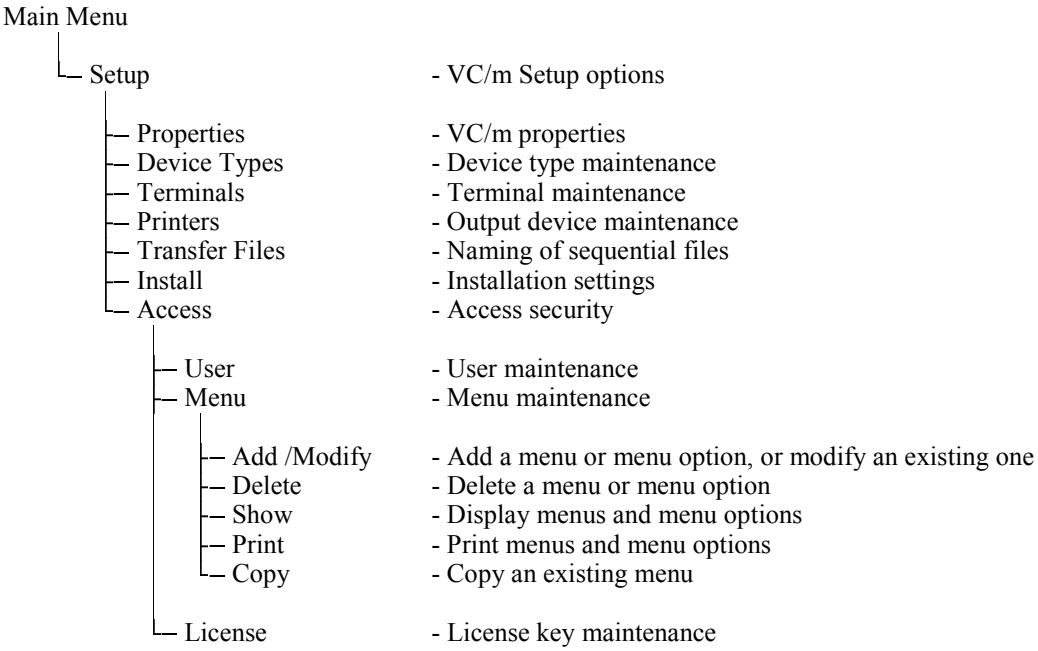
Many of the menu options also have two or four standard options below them: Show and Print, or Add /Modify, Delete, Show and Print.

Note: The VC/m menu structure is customizable and may therefore be different from what is shown.

```
Main Menu
└─ Manager                    - VC/m System and Project Manager options
   ├─ Master Files            - VC/m master files
   │  ├─ User                 - User maintenance
   │  ├─ Module               - Module maintenance
   │  ├─ Location             - Location maintenance
   │  │  ├─ Physical Location - Physical location maintenance
   │  │  ├─ Location          - Logical location maintenance
   │  │  │  ├─ Add /Modify    - Add a location or modify an existing one
   │  │  │  ├─ Delete         - Delete a location
   │  │  │  ├─ Copy           - Copy version information to another location
   │  │  │  ├─ Version        - Delete version information from a location
   │  │  │  ├─ Show           - Display details of locations
   │  │  │  └─ Print          - Print details of locations
   │  │  ├─ Class             - Location class maintenance
   │  │  └─ Routes            - Transfer route maintenance
   │  ├─ System               - System maintenance
   │  ├─ Types                - Change request type maintenance
   │  └─ Change Request       - Change request maintenance
   ├─ Reports                 - Management reports
   │  ├─ Audit                - Audit trail
   │  │  ├─ Object            - Audit trail activity by object or change request
   │  │  └─ Date              - Audit trail activity by date
   │  ├─ Release              - Release control sheet
   │  ├─ Matrix               - Matrix of versions by locations
   │  ├─ Checked-out          - Checked-out objects report
   │  ├─ Change Request       - Change request summary by period
   │  ├─ Unregistered         - Unregistered components report
   │  ├─ Summary              - Summary of transfers by period
   │  ├─ Version              - Versions across locations report
   │  └─ Concurrent           - Concurrent development analysis
   ├─ Transfer                - Transfer an object between locations
   ├─ Archive                 - Archive and purge old versions
   ├─ Bulk                    - Transfer multiple objects between locations
   ├─ Load                    - Bulk component registration
   ├─ Baseline                - Create a new baseline
   ├─ Delete                  - Remove an object completely
   └─ Integrity               - Database integrity checker
```

VC/m Set-up Menu

Note: The VC/m menu structure is customizable and may therefore be different from what is shown.



VC/live Menu

Note: The VC/m menu structure is customizable and may therefore be different from what is shown.

Main Menu

- Install - Install objects from a sequential file
- Force install - Force installation with overrides
- Rollback - Roll back objects from a backup file
- Force rollback - Force rollback with overrides
- Audit trail - Audit trail
  - Object - Audit trail activity by object or change request
  - Date - Audit trail activity by date

2.4 Set-Up Options

Device Types: Device type maintenance

A number of standard device types are pre-defined, as follows:

DEFAULT	Default device type (pre-set as for ANSI)
SYSTEM	For internal use only
IBMPC	IBM Compatible PC display (DataTree Console)
MSMCON	MSM console
VDU	Basic character terminal
ANSI	ANSI standard terminal
LP	Basic character printer
HPL	HP Laser printer (landscape)
HPP	HP Laser printer (portrait)
LN03	DEC Laser printer
DOSFILE	Sequential DOS file
WINFILE	Sequential Windows file

In order to use devices which are not in the list above, an entry needs to be created in the device type file, specifying the attributes of the device.

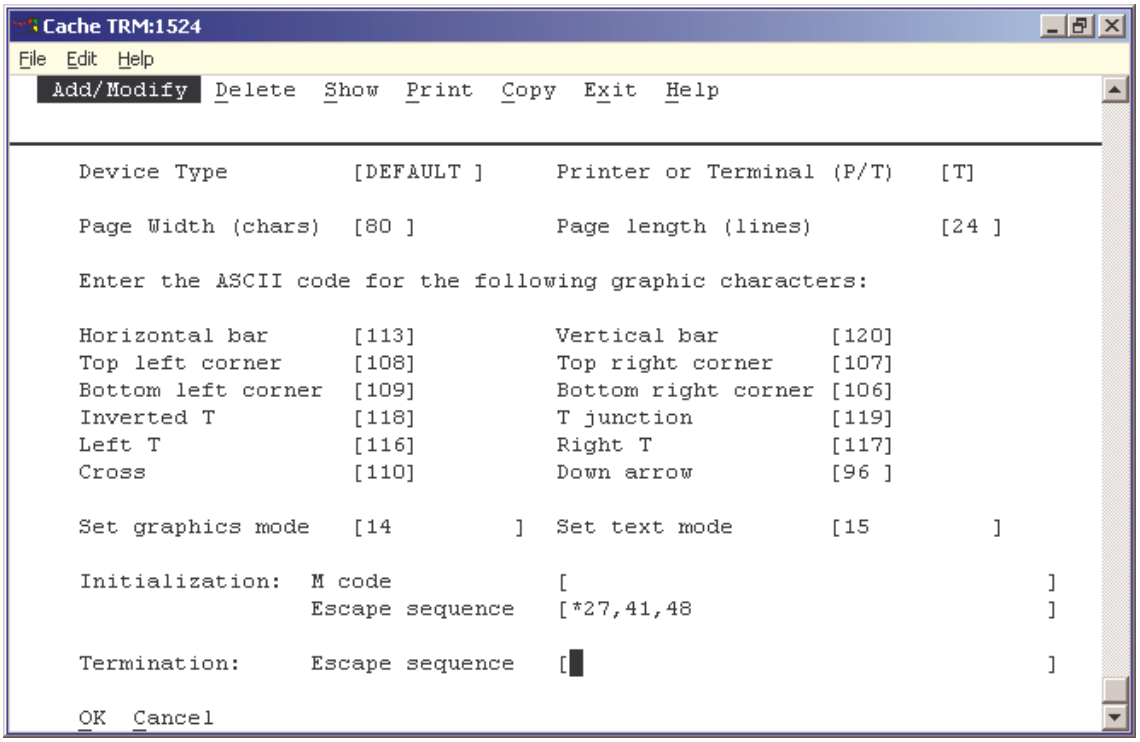


Figure 2.5 Device type maintenance

- Device Type

  Enter a code to identify the device type. Certain device types, e.g. ANSI, which are supplied as a standard part of the package, cannot be modified. You may, however, create as many new ones as you need.

- Printer or Terminal

  Enter P if the device you are defining is a printer or other output device. Enter T if it is a terminal or PC screen.

- Page Width

  Enter the width of the page (or screen) in characters. Where relevant, VC/m will format the screen or page to fit within this width. Note that most screen displays and reports in VC/m assume a minimum width of 80 characters.

- Page length

  Enter the length of the page (or screen) in lines. Where relevant, VC/m will format the screen or page to fit within this length. Note that most screen displays in VC/m assume a minimum length of 24 lines.

- ASCII code for graphic characters

  The next twelve prompts define the line and box drawing characters to be used for this device.

  You should enter the ASCII code corresponding to the character that should be displayed in each case.

Field	Character
Horizontal bar	─
Top left corner	┌
Bottom left corner	└
Inverted T	┴
Left T	├
Cross	┼

Field	Character
Vertical bar	│
Top right corner	┐
Bottom right corner	┘
T junction	┬
Right T	┤
Down Arrow	v

- Set graphics mode

  If the device uses an alternative character set for graphics characters, enter the ASCII codes required to switch the device to that character set. Enter each character in the control string as an ASCII code, delimited by commas.

- Set text mode

  If the device uses an alternative character set for graphics characters, enter the ASCII codes required to switch the device back from the graphics character set to the normal text character set. Enter each character in the control string as an ASCII code, delimited by commas.

- Initialization: M code

  Enter any string of M code which you wish to be executed on device initialization.

- Initialization: Escape sequence

  If the device requires a special initialization string (for example, to select compressed print), enter the control sequence required. This can be entered in any one of the following formats:

Format	Example
Text string	`!r!font 9;exit;`
ASCII code sequence	`*27,41,48` `*nn,nn,nn,...`
Escape sequence	`@[10l;@10c;@[1m`

Note: @ is used for escape.

Note: The formats cannot be mixed within the same field.

- Termination: Escape sequence

  If the device requires a special termination string (for example, to reset the printer margins), enter the control sequence required.

  It can be entered in any of the formats described for the initialization string.

Install: Installation settings

If the install option of VC/live is used, this screen is used to specify the locations which are used.

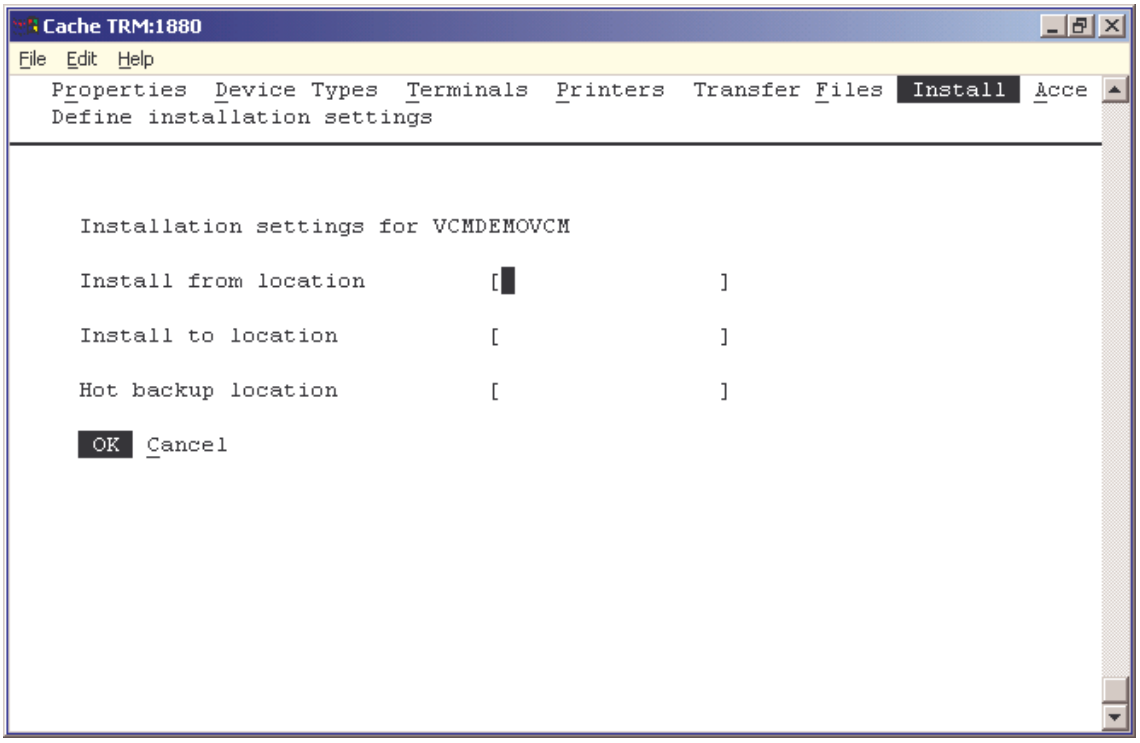


Figure 2.6 Installation settings

License: License key maintenance

This option is used to enter a new or amended license key. It controls the number of users, and the component and location drivers which are enabled. Individual features can also have an expiry date for evaluation purposes.

The license key is stored in an operating system file.

This option can be used to view the details of the key. It can also be used to manually edit the details, although this is unlikely to be necessary.

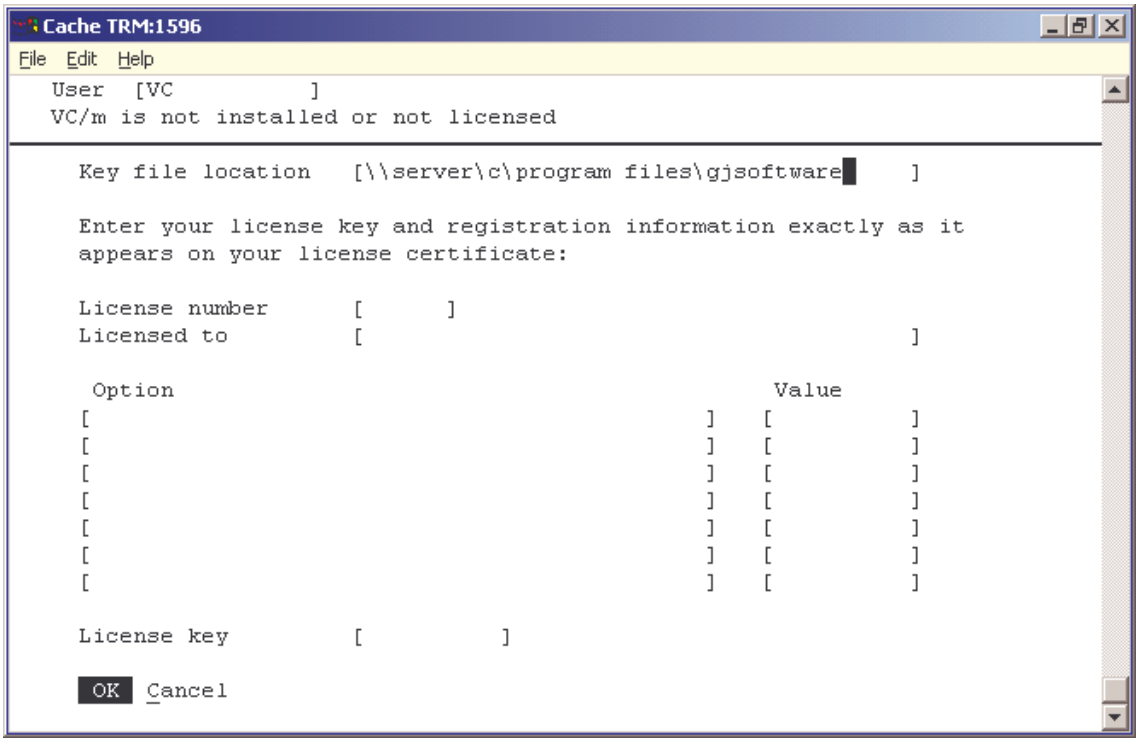


Figure 2.7 License key maintenance

- Key file location

  Enter the location of your license key. If you are running in a networked environment, it is recommended that you enter the path for the license key location using the whole network path.

  When <return> is pressed, the details of your license will be displayed.

Menu: Menu maintenance

VC/m menus can be customized to provide tailored menus and menu options for use by different groups of users. Access codes can also be assigned to control which users are able to use which options.

Menus can be defined for both the browser and the character-based interfaces. Access controls can also be set for both types of menu.

Each menu record defines either a menu or a menu option, whether part of VC/m or not. A menu comprises a list of options each of which can invoke another menu or menu option. A menu option defines the name of the menu option and enables access codes to be attached.

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

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 are standard VC/m menu options.

When a standard VC/m menu or menu option is selected, only the access code field is editable.

To customize a menu, either a completely new menu can be created or an existing menu can be copied and then modified. Once the new menu has been created, it can be used to replace one of the existing main VC/m menus or it can be invoked directly.

Add /Modify: Add a menu or menu option or modify an existing one

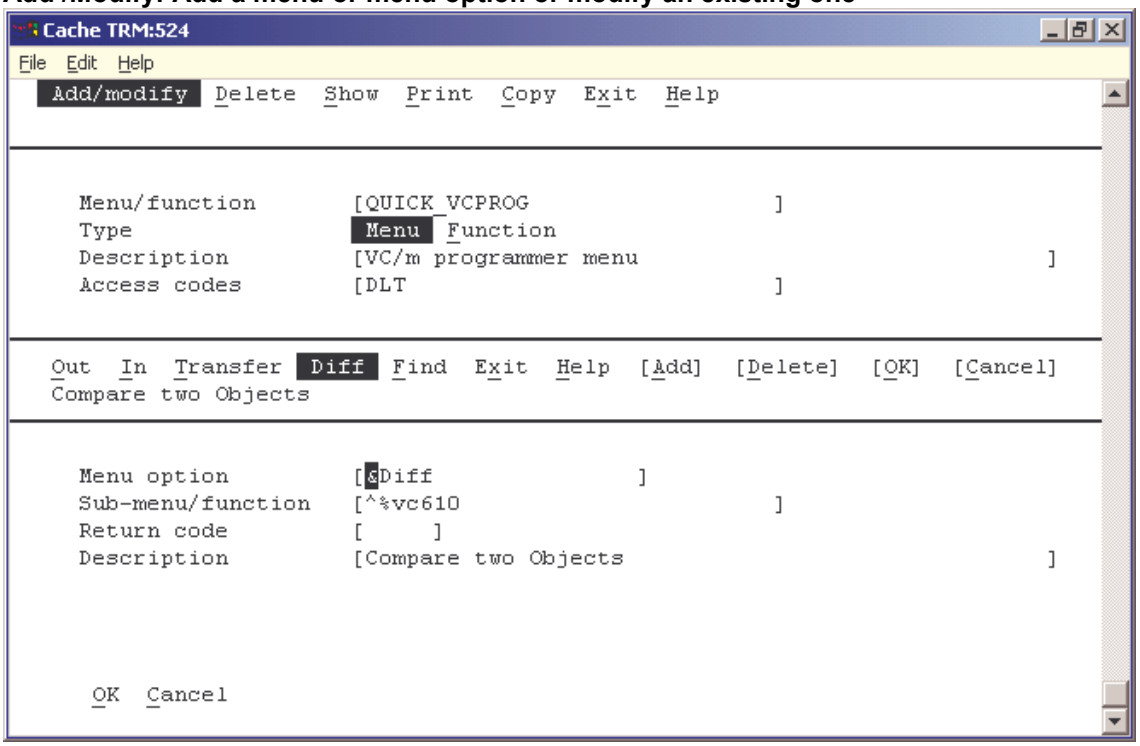


Figure 2.8 Add a menu or modify an existing one

- Menu /function

  Enter the name of the menu or menu option. A selection list of existing menus and menu options is available.

More detailed information can be found in the General VC/m Reference, p. 107.

A menu name can be any string of letters and numbers. A menu option name is the line label of the M code to be invoked (e.g. ^ABC or LABEL^ABC).

- Type

  Select Menu or Function for the type of entry. A function is a menu option.

- Description

  Enter a description of the menu or menu option. This is the text which will appear on the second line of the screen when the menu option is highlighted.

- Access codes

  Enter a list of access codes that determine which users are allowed to access this menu or menu option. The access codes can either be a simple list of codes or sets of codes enclosed in parentheses. Any user who has a matching access code can access the menu or invoke the menu option. The access codes are considered to match if any of the codes in the user access code coincide with any of the codes attached to the menu or menu option.

  More detailed information can be found in the VC/m User Guide..

- Menu Edit Bar

  The menu edit bar displays the menu as it is currently defined, followed by four edit buttons (enclosed in square brackets). Use the left and right cursor keys to move along the bar, in the same way as when choosing options from a menu at the top of the screen. If not all the items are displayed, the screen will scroll. Press <return> to select the highlighted option.

  When one of the defined menu items is selected, it can be edited. Use the edit buttons to add and delete items. Use the OK button to save the changes when they are complete.

- [Add]

  Press <return> when this button is highlighted to add an item to the menu. You will be prompted, with the message 'Insert before ...', to identify the position on the menu bar where you want to insert the new item. Move the highlighted option using the left or right cursor keys until the item highlighted is the one before which you wish to insert the new menu option, then press <return>. You will then be prompted for the attributes which define the menu item.

- [Delete]

  Press <return> when this button is highlighted to delete an item from the menu. You will be prompted, with the message 'Delete option ...', to position the highlight on the menu option which you want to delete. Once you have positioned the highlight on the desired option, press <return>. The details of the selected item are displayed and you will be prompted to confirm that you wish to delete the item.

- [OK]

  When you have finished creating or modifying the menu, select the [OK] button to save the changes you have made. No changes are permanent until you select this button.

- [Cancel]

  Select this button if you wish to abandon the changes which you have made.

- Menu option

  Enter the text that you want to appear on the menu bar for this menu item. By default, the first character of the text is a shortcut key for the menu item. If you wish to specify an alternative character as the shortcut key, prefix that character with an ampersand. For example, 'Chec&k Out' would make K the shortcut key and the menu option would appear on the menu as 'Check Out'.

- Sub-menu /function

  If this menu item is to invoke a sub-menu, and it already exists, enter the name of the menu. If the menu has not yet been created, enter `^%vcmnu("menu")` where *menu* is the name of the menu that you want to invoke. A menu name can be any string of letters and numbers.

  When <return> is pressed, VC/m converts the name of any existing menu to the form `^%vcmnu("menu")` where *menu* is the name of the menu.

  If you want to invoke a menu option, enter the line label of the M code to be invoked (e.g. ^ABC or LABEL^ABC). If the line label is already defined as a VC/m menu option, its description is used as the default description on this menu. Any access codes defined for the menu option also apply when a user attempts to invoke this option. The routine label must include the ^ character.

- Return code

  If you do not enter a sub-menu or menu option at the previous prompt, you will be prompted for a return code. The return code of E is always interpreted by the VC/m menu controller as an exit command. This field is only applicable for including an Exit option, or when calling the menu or option from within an M routine.

  If you add an Exit option to your menu, it should have the following attributes:

```
Menu option       [E&xit      ]

Sub-menu/function [        ]

Return code       [E ]

Description       [Return to previous menu   ]
```

- Description

  Enter the text that you want to appear on the second line of the screen when the menu option is highlighted. If you have referenced an existing sub-menu or menu option, the description defaults to the description of that item, but this can be overridden for context-specific uses.

Delete: Delete a menu or menu option

This option can be used to remove a menu or menu option from the VC/m database. It cannot be used for standard VC/m menus and menu options.

Copy: Copy a menu

This option can be used to copy a menu to another name. This provides an easy way to start creating a customized menu.

Printers: Output device maintenance

Each output device is identified by a logical name, which need not be the same as its $IO value. VC/m will not be able to use any printer or other output device which does not have an entry in this file.

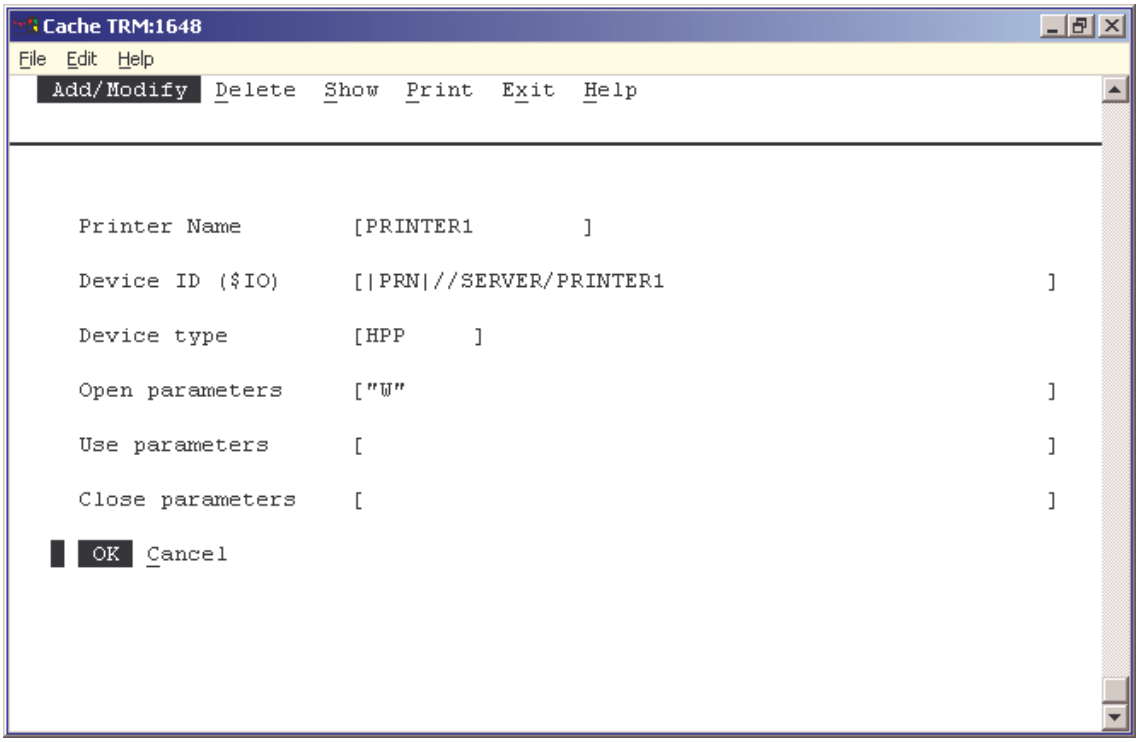


Figure 2.9 Output device maintenance (printer)

- Printer name

  Enter the name by which the printer or output file is to be known.

- Device ID ($IO)

  Enter the address of the output device. This may be a file name, a device number, or a device mnemonic, depending on the host operating system and M implementation that you are using.

  If you enter '???' in this field, the device ID is entered at print time following the device name. For example, if a device called FILE is set up with a device ID of ???, at any Output Device prompt, the actual device ID to be used is input following the device name.

```
Output Device        [FILE MYFILE.TXT          ]
```

- Device type

  Enter the device type code for the output device, as defined in the device type table.

- Open parameters

  If the device requires special parameters when it is opened, enter them here. If there are multiple parameters, they probably need to be in brackets. For example:

```
Open parameters      [(width=0:escape          ]
Open parameters      [(0::::262208)            ]
```

If three question marks are entered at any point in the open parameters field then, in a similar way to the Device ID, a value can be supplied at print time which will be substituted for the three question marks. For example, the open parameters contain the following:

```
Open parameters      [(file="???":mode="W")     ]
```

At print time, the file name can be specified at the Output Device field:

```
Output Device        [FILE MYFILE.TXT           ]
```

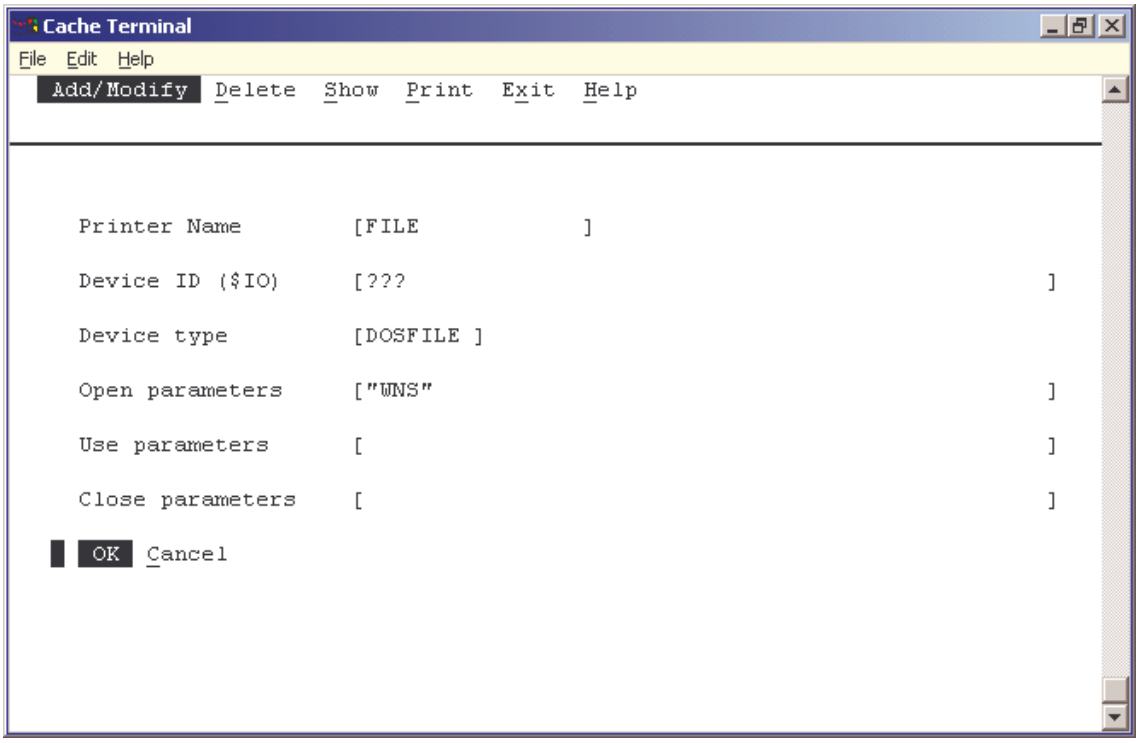


Figure 2.10 Output device maintenance (file)

- Use parameters

  If the device requires special parameters when it is used, enter them here. If there are multiple parameters, they probably need to be in brackets.

- Close parameters

  If the device requires special parameters when it is closed, enter them here. If there are multiple parameters, they probably need to be in brackets.

Properties 1: Transfer callouts and custom menus

A transfer callout is an additional, installation-specific piece of code which should be run whenever a transfer is made. This screen is also used to specify customized menus to be used. The menus are created using the Menu option in Set-up.

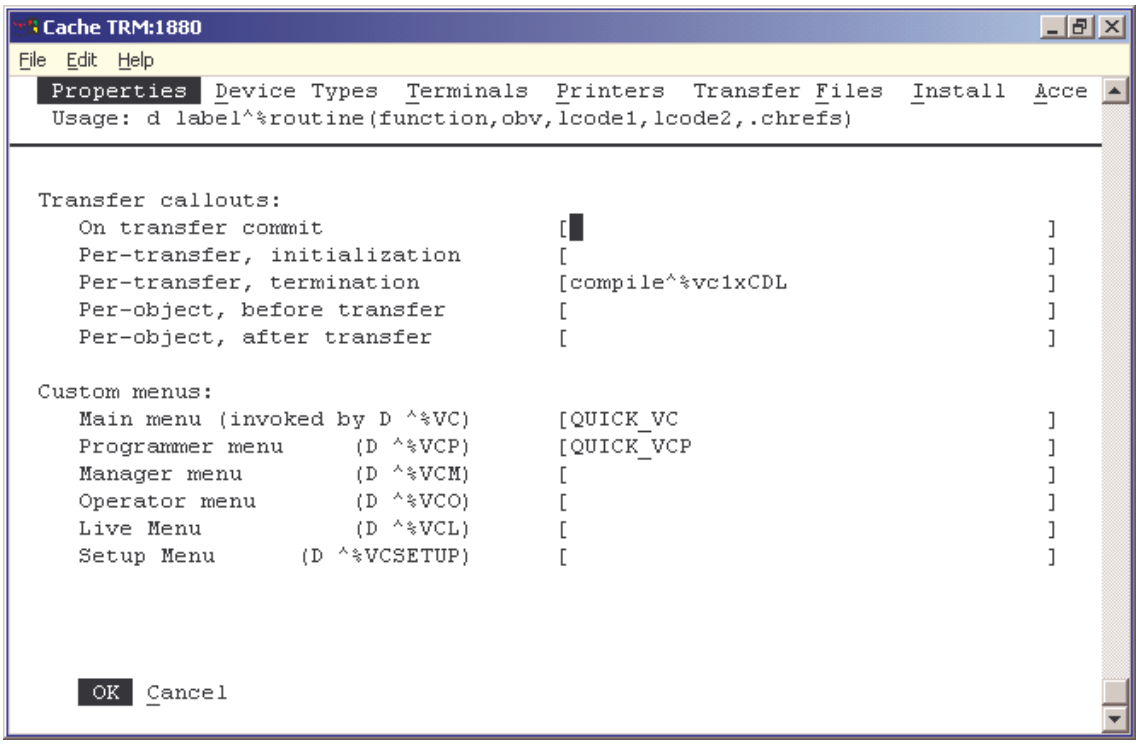


Figure 2.11 Transfer callouts and custom menus

Transfer callouts

- On transfer commit

  Enter the line label of the M code to be invoked, e.g. trancomm^%gjextra.

- Per-transfer, initialization

  Enter the line label of the M code to be invoked, e.g. traninit^%gjextra.

- Per-transfer, termination

  Enter the line label of the M code to be invoked, e.g. tranterm^%gjextra.

- Per-object, before transfer

  Enter the line label of the M code to be invoked, e.g. tranbobj^%gjextra.

- Per-object, after transfer

  Enter the line label of the M code to be invoked, e.g. tranaobj^%gjextra.

Properties 2: Component driver callouts

A component driver callout is an additional, installation-specific piece of code which should be run whenever a particular type of component is changed.

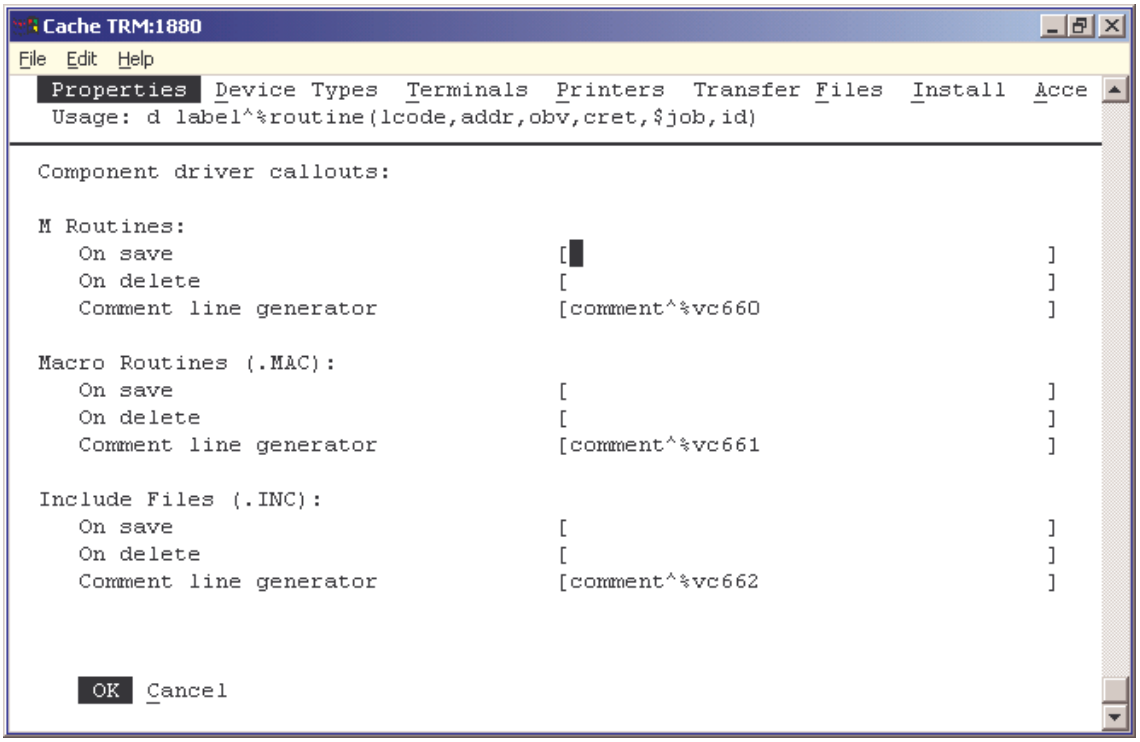


Figure 2.12 Component driver callouts

Component driver callouts: M Routines

- On save

  Enter the line label of the M code to be invoked, e.g. msave^%gjextra.

- On delete

  Enter the line label of the M code to be invoked, e.g. mdel^%gjextra.

- Comment line generator

  Enter the line label of the M code to be invoked, e.g. mcomm^%gjextra.

Component driver callouts: Macro Routines (.MAC)

- On save

  Enter the line label of the M code to be invoked, e.g. macsave^%gjextra.

- On delete

  Enter the line label of the M code to be invoked, e.g. macdel^%gjextra.

- Comment line generator

  Enter the line label of the M code to be invoked, e.g. maccomm^%gjextra.

Component driver callouts: Include files (.INC)

- On save

  Enter the line label of the M code to be invoked, e.g. incsave^%gjextra.

- On delete

  Enter the line label of the M code to be invoked, e.g. incdel^%gjextra.

- Comment line generator

  Enter the line label of the M code to be invoked, e.g. inccomm^%gjextra.

Properties 3: Change request callouts and M implementation-specific settings

A change request callout is an additional, installation-specific piece of code which is run when certain changes are made to a change request.

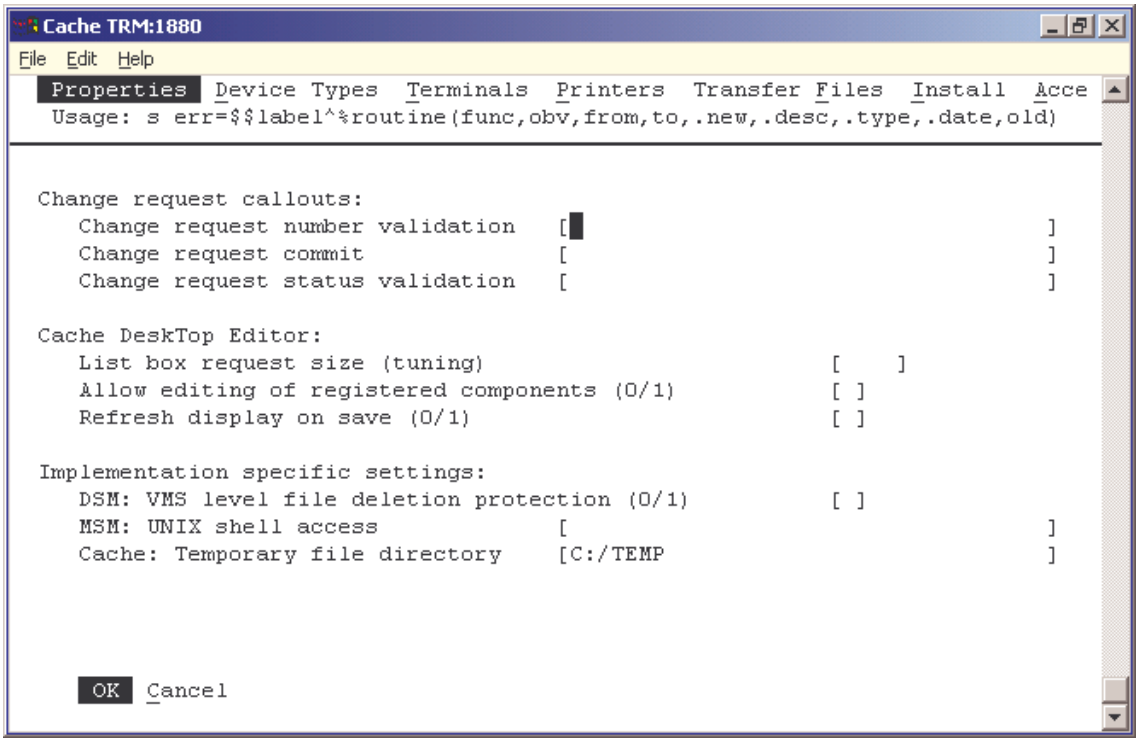


Figure 2.13 Change request callouts and M implementation-specific settings

Change request callouts

- Change request number validation

  This callout is called to validate the change request code whenever a new change request is created. Enter the line label of the M code to be invoked, e.g. crcode^%gjextra.

  More detailed information can be found in the VC/m User Guide.

- Change request commit

  Enter the line label of the M code to be invoked, e.g. crcomm^%gjextra.

- Change request status validation

  Enter the line label of the M code to be invoked, e.g. crstat^%gjextra.

Implementation Specific Settings

- Cache Temporary File Directory

  This setting can have both a directory and a file prefix. If only the directory is specified, it must have a trailing directory delimiter, e.g. on Windows, c:\temp\.

Properties 4: Default values for function screens

This screen is used to set up default values for several of the prompts which appear on the screen.

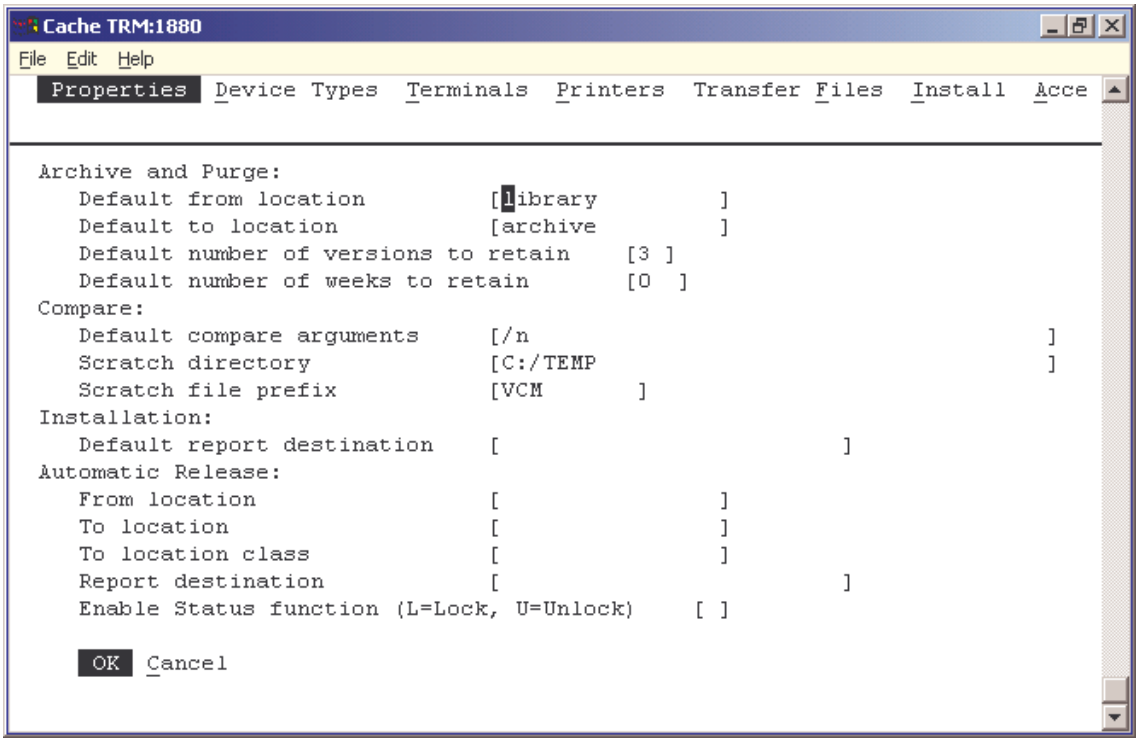


Figure 2.14 Default values for function screens

Properties 5: Audit trail settings

For the audit trail, the message level for various kinds of transfer must be set up. Without these, VC/m will fail to operate.

For the audit trail message levels, 0 is the most important, 98 is the least important and a level of 99 will result in no permanent entry being stored in the audit trail file.

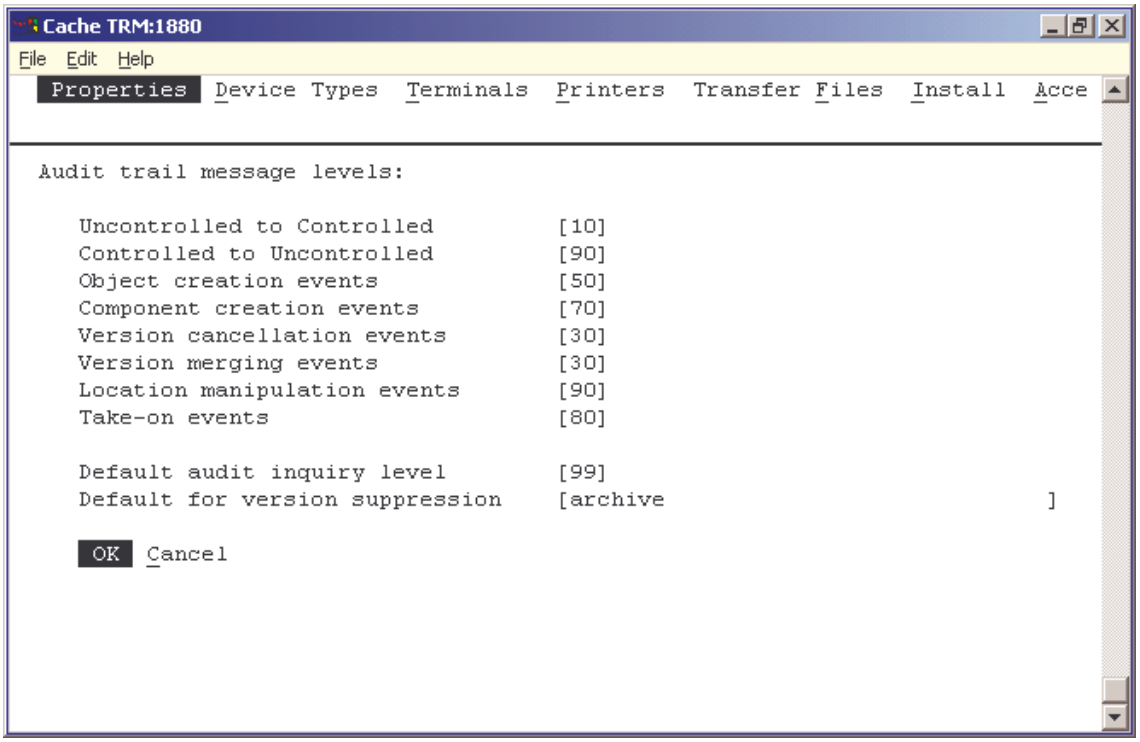


Figure 2.15

The Default for version suppression field allows you to specify locations which should not be included in a Versions across Locations report. Suggested entries for this field would normally be the OLD and ARCHIVE locations.

Properties 6: Miscellaneous properties

This screen is used to specify various miscellaneous values for VC/m.

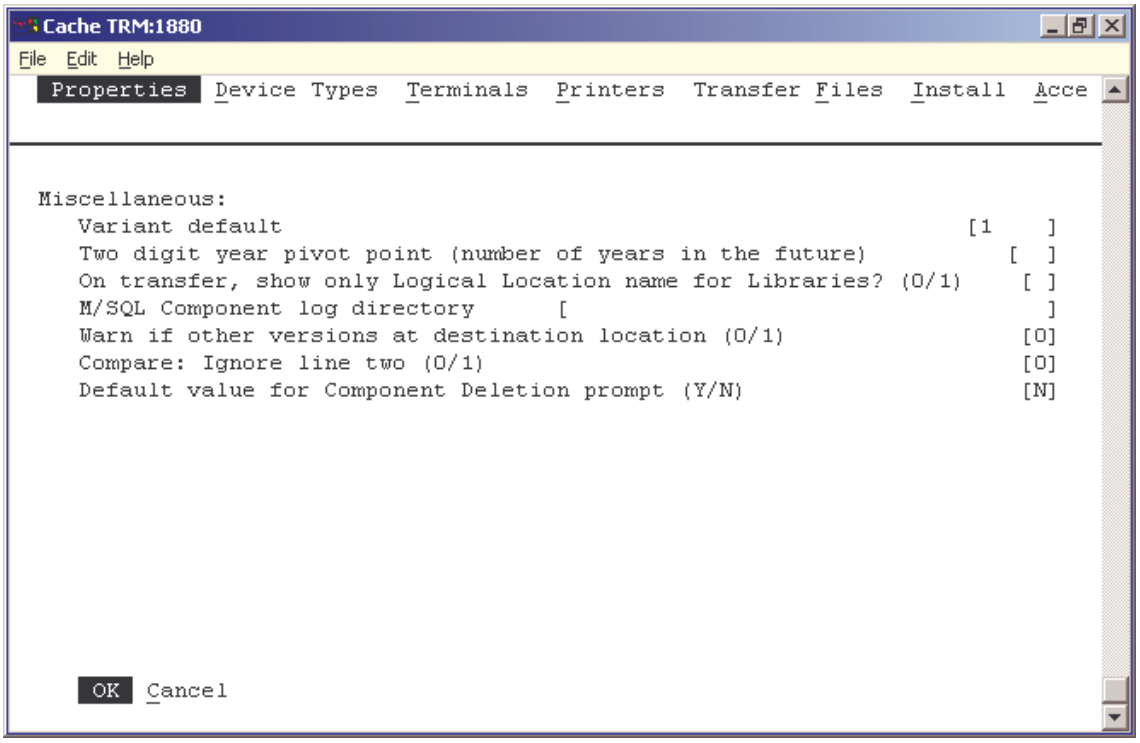


Figure 2.16 Miscellaneous properties

Terminals: Terminal maintenance

VC/m will run on any terminal that 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 terminals that will be used with VC/m over a terminal server or network.

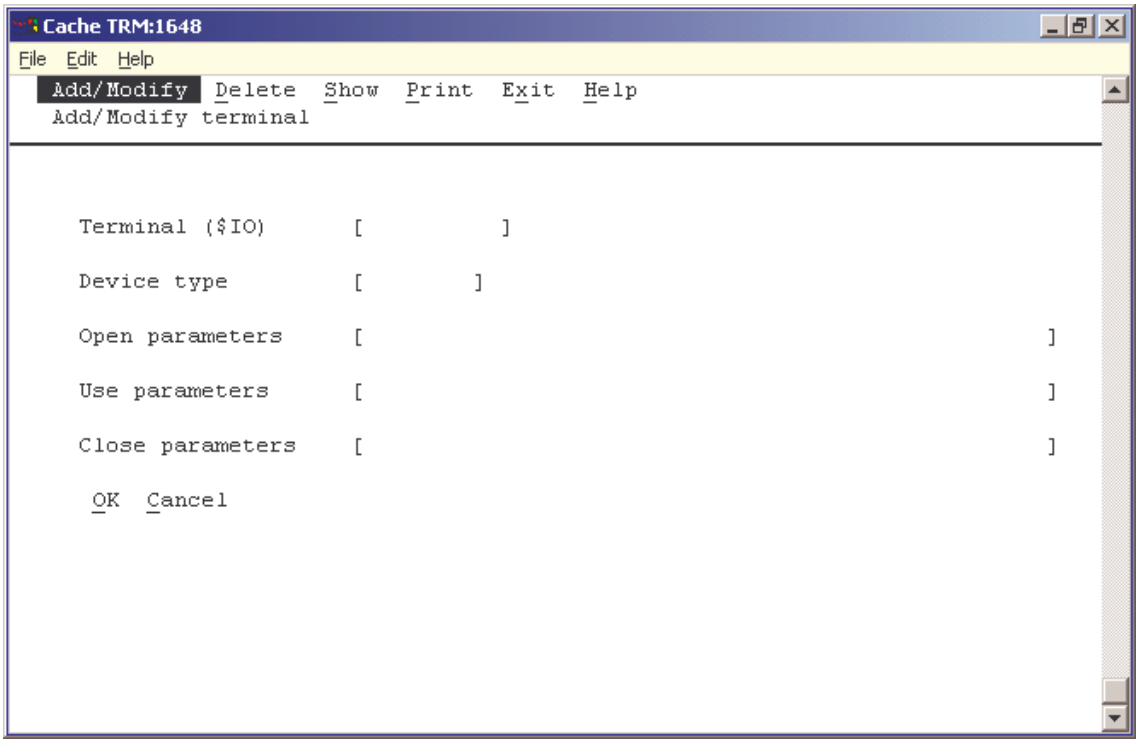


Figure 2.17 Terminal maintenance

- Terminal ($IO)

  Enter the $IO value that identifies the terminal.

- Device type

  Enter the device type code for the terminal, as defined in the device type table.

- Open parameters

  If the device requires special parameters when it opened, enter them here. If there are multiple parameters, they will probably need to be in brackets. For example:

```
Open parameters      [(width=0:escape)          ]

Open parameters      [(0::::262208)             ]
```

- Use parameters

  If the device requires special parameters when it is used, enter them here. If there are multiple parameters, they probably need to be in brackets.

- Close parameters

  If the device requires special parameters when it is closed, enter them here. If there are multiple parameters, they probably need to be in brackets.

Transfer Files: Naming of sequential files

For sequential locations, it is possible to specify the file names which will be used.

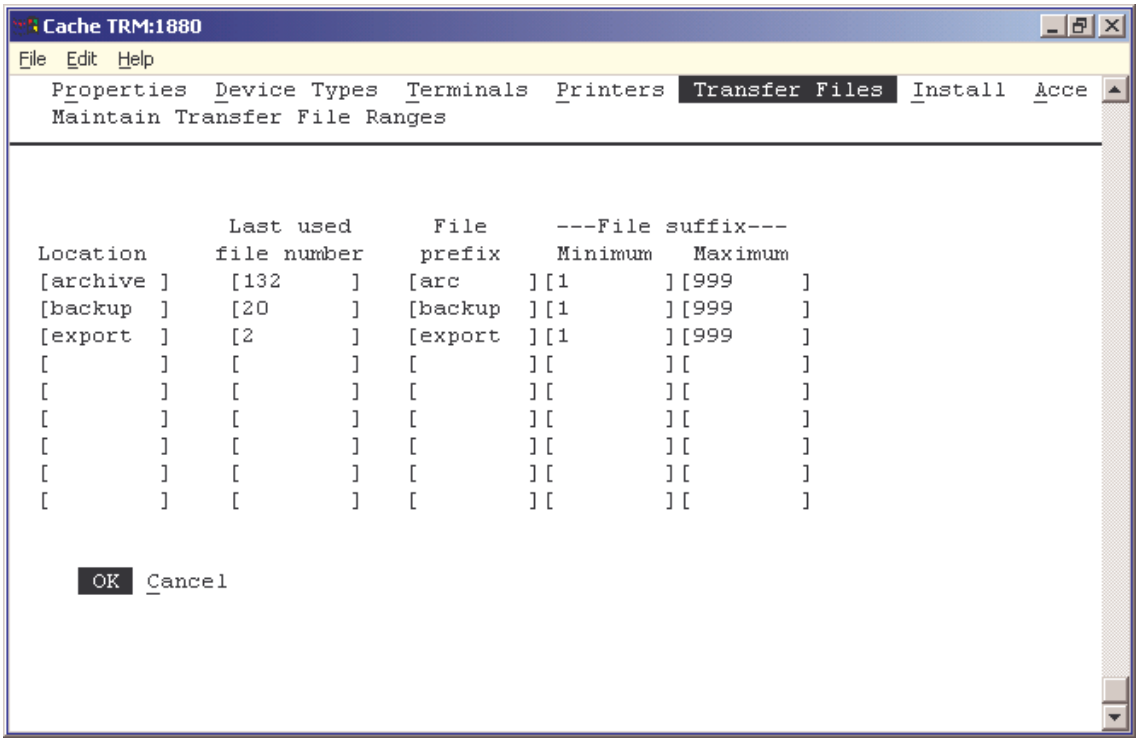


Figure 2.18 Naming of sequential files

2.5 VC/m Master Files

Change Request: Change request maintenance

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

Often, a change request is created as an object version is checked out. The change request maintenance screen can be reached by entering the code for a change request which does not exist or the code for a new change request in the change request field on the check-out screen.

Add /Modify: Add a change request or modify an existing one

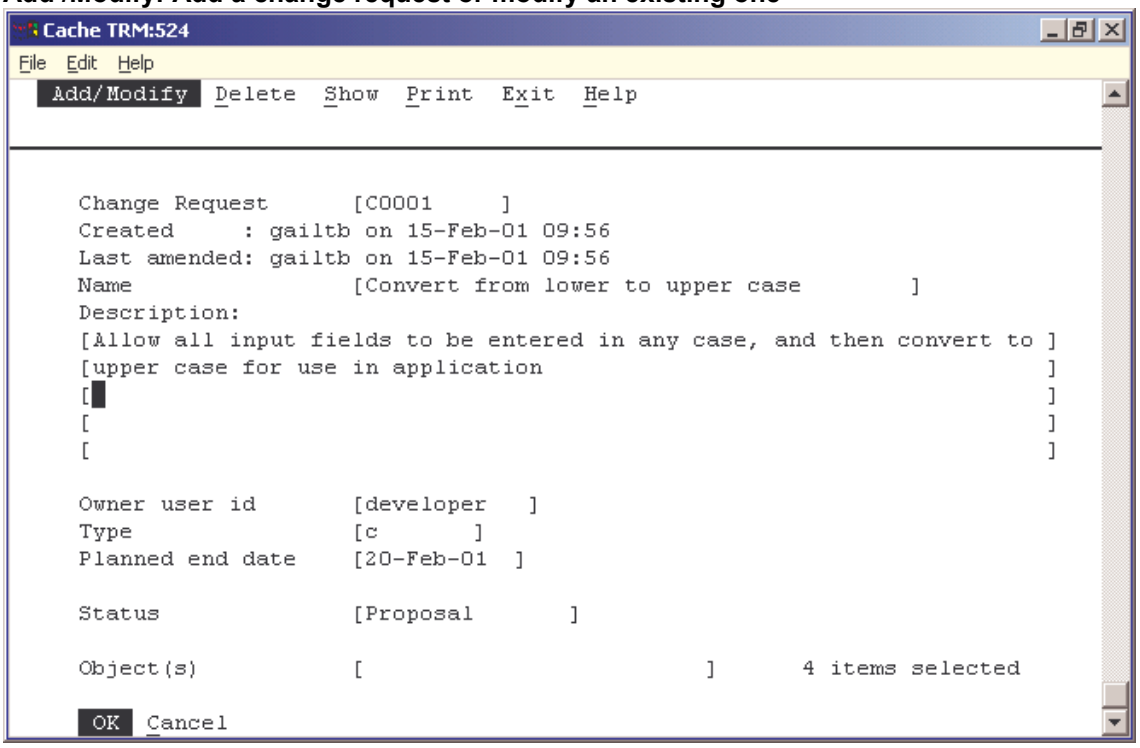


Figure 2.19 Add a change request or modify an existing one

- Change Request

  Enter the unique identifier for the project, task, error reference, etc. A selection list of change requests defined in VC/m is available. Alternatively, if the change request type is set up to allocate numbers automatically, the change request type can be entered here.

  If the change request exists, when <return> is pressed, the user ID and date are displayed for the creation of the change request, and the last modification made to it.

  Note: The wording of this prompt may vary depending on what is entered in the field for Prompt text under Change request type maintenance.

- Name

  Enter a descriptive name for the change request.

- Description

  A scrolling display of 5 long text lines is available for recording details about the change request.

- Owner user id

  Enter the user who created or is responsible for this change request. If a change request is created at the time that an object is checked out, the user ID will be that of the person who checked out the object. A selection list of users defined for VC/m is available.

- Type

  This field is used to categorize change requests by type for reporting purposes. A selection list of types defined for VC/m is available. These types are set up in the Change Request Type master file.

- Planned end date

  Enter the date by which the work on the objects is planned to be completed. This is the estimated date at which the objects can be returned to the library. It is used by the 'Overdue Checked-out Objects' report.

- Status

  This prompt is dependent on what is set up for this change request type. The types of status are set up in the change request type file. If the change request type does not have at least one status defined, the prompt does not appear. A selection list is available. If the status has a tied location, the objects in the change request will automatically be transferred to the location.

- Object(s)

  Enter the object versions to be included in the change request. They can be selected and de-selected in the standard ways.

  More detailed information can be found on p. 27.

  A selection list of the objects in VC/m is available.

  The contents of another change request can also be included at this prompt by prefixing a change request code with @.

  Alternatively, object versions can be added to a change request at the time when they are checked out.

- Comment

  A full screen of comment fields is available for entering information about the change request. This is a scrolling block of 16 long text lines.

Delete: Delete a change request

This option can be used to remove a change request from the VC/m system. Normally it is not desirable to do this, since it can remove valuable information about work which has at least been considered.

Class: Location class maintenance

A location class is a convenient way of grouping several logical locations together for transfer operations. For example, all live locations may be placed in a location class called LIVE. When a transfer function in VC/m asks for a 'to location', a location class can be entered instead, prefixed with @. Location classes can also be used instead of locations in the dependent locations and the new active locations fields when defining a transfer route. They cannot be used for the new master location field on this screen, nor can they be used in the 'from location' field for a transfer.

Add /Modify: Add a location class or modify an existing one

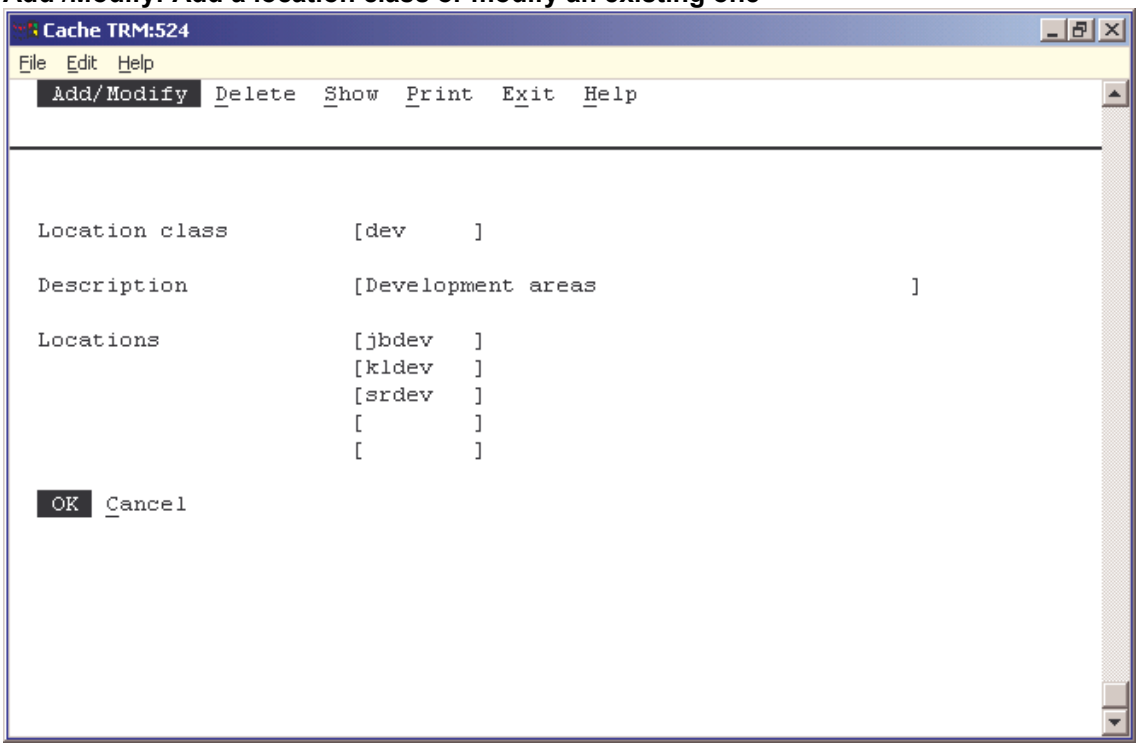


Figure 2.20 Add a location class or modify an existing one

- Location class

  Enter the name for the location class. A selection list of location classes defined in VC/m is available.

- Description

  Enter a description for the location class.

- Locations

  Enter the locations which are to be included in this location class.

Delete: Delete a location class

This option can be used to remove a location class from the VC/m system.

Delete: Remove an object completely

By default, this option is located on the Manager's menu in VC/m. It allows one or more object versions to be completely deleted from the VC/m database. All the location and component information for each selected item 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.

Normally it is not necessary or desirable to completely erase an object or object version from VC/m, as the benefits of software security and audit trail would be lost. However, there are some occasions when it is necessary to delete all trace of an object from VC/m, for example, when using a temporary object to test the set-up of VC/m transfer routes.

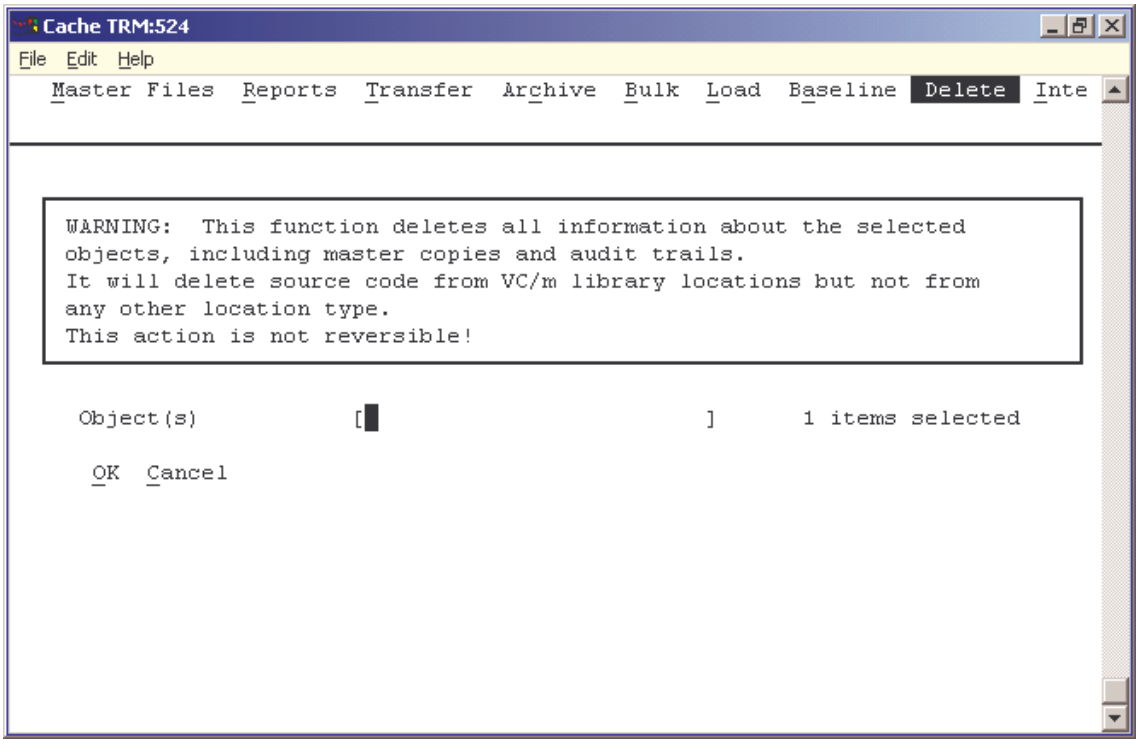


Figure 2.21 Remove an object completely

- Object(s)

  Enter the objects, variants or versions to be deleted. They can be selected and de-selected in the standard ways.

  More detailed information can be found on p. 27.

  A selection list of the objects in VC/m is available.

Location: Location maintenance

Location maintenance defines the logical locations. Each logical location maps onto one or more physical locations. Several logical locations may also map onto the same physical location.

Add /Modify: Add a location or modify an existing one

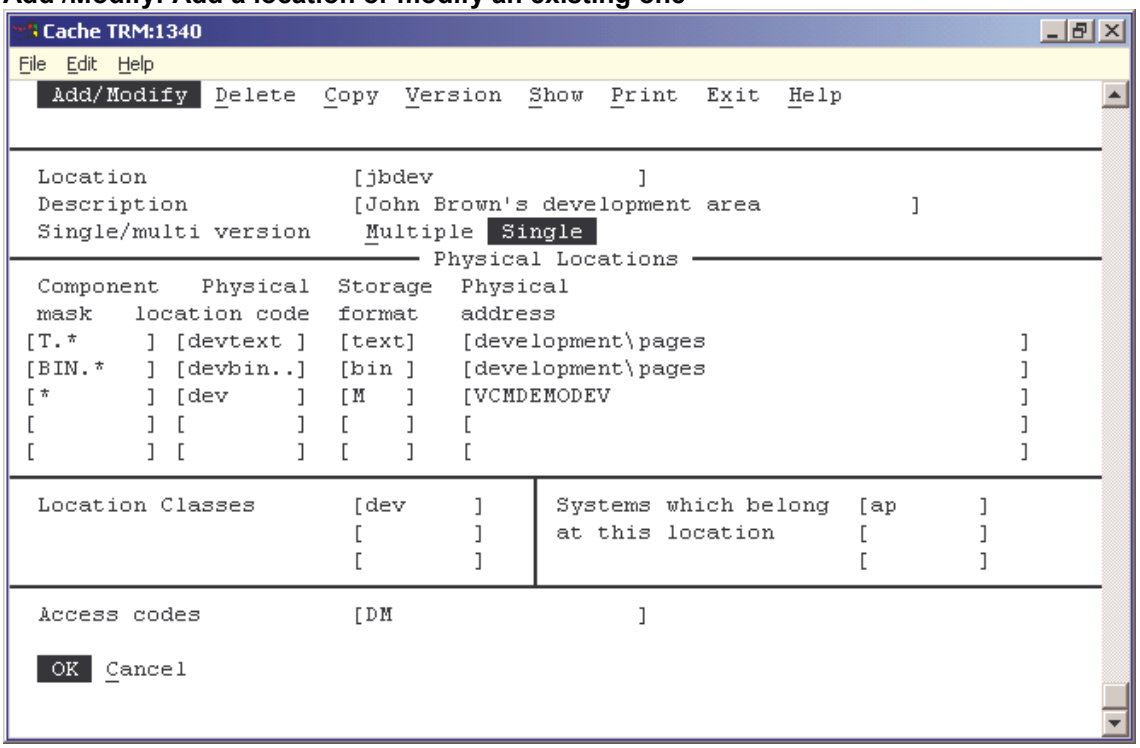


Figure 2.22 Add a location or modify an existing one

- Location

  Enter the code to be used for the location.

  Note: EMPTY should not be used as a location code because it has a meaning which is already defined to the system for dependent locations.

- Description

  Enter a description for the location.

- Single /multi-version

  Each physical location is defined as either a single or multi-version location. Any logical location which maps only onto single-version physical locations must be, by definition, a single-version location. Any logical location which maps onto a multi-version physical location may itself be defined either as a single-version or as a multi-version location.

  The main purpose of this functionality is to allow library locations which only hold the most recent version of each object. When a new version is copied to such a location, as with any other single-version location, its ancestor is displaced.

- Component mask

  This field is used to specify which components at this location map to a particular physical location. VC/m searches through this list in order until it finds the first mask which matches the component. Be careful therefore that all possibilities are covered. Component types, specific names and wildcards can all be used. Examples are as follows:

  `T.*.html` All components of type text and with extension .html

  `T.*` All components of type text

  `*` All components

- Physical location code

  Enter the code to refer to the physical location.

- Storage format

  If a physical location has already been defined, this value will default. If not, the physical location can be defined here.

  More detailed information can be found in VC/m's Character-Based Interface Reference, p. 67

- Physical address

  If a physical location has already been defined, this value will default. If not, the physical location can be defined here.

  More detailed information can be found in VC/m's Character-Based Interface Reference, p. 67

- Location Classes

  The location can be associated with one or more location classes.

- Systems which belong at this location

  All the systems that are permitted in this location should be entered. If a system is not entered, objects that belong to that system cannot be transferred to this location.

- Access codes

  These access codes are used to determine which locations are displayed on the browser interface. If a user who connects to VC/m has the required access codes, this location will be accessible via the browser interface. If no access codes are entered here, the location will be accessible to all users.

Delete: Delete a location

This option is used to remove the definition of a logical location from the VC/m system. It should normally only be used for locations which were set up erroneously.

Copy: Copy version information to another location

The Copy option copies VC/m's information about the object versions at a logical location to another logical location. VC/m populates the new location with the same version information as was at the first location.

If a physical location is copied, this option can be used to create a new logical location with the correct object version information. Note that VC/m does not actually physically copy any components when this option is used.

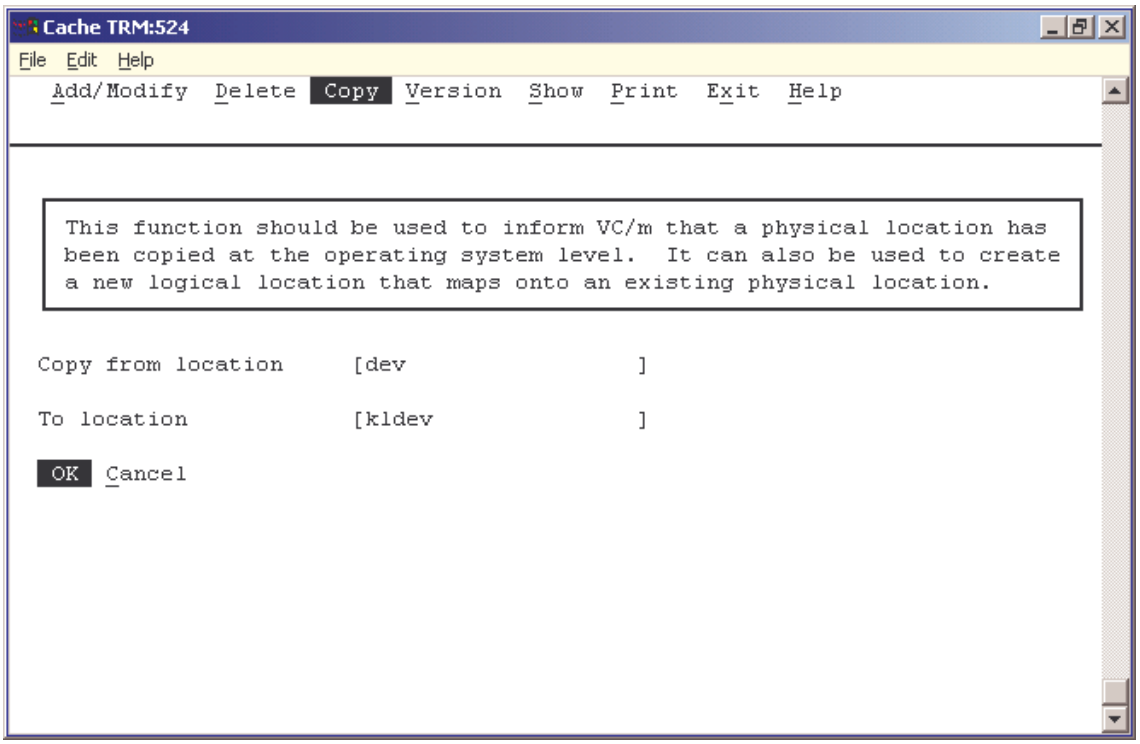


Figure 2.23 Copy version information to another location

Version: Delete version information from a location

This option deletes all of the version information which VC/m stores for a location. It can be used, for example, to inform VC/m that a location has been physically deleted. It can also be used to remove a redundant logical location which is one of several mapping onto the same physical locations.

When a location is deleted, it is de-activated. The components at the location are not physically deleted, and the audit trail is not deleted.

> ***Warning: This operation is not undo-able and can destroy valuable information about the contents of a location.***

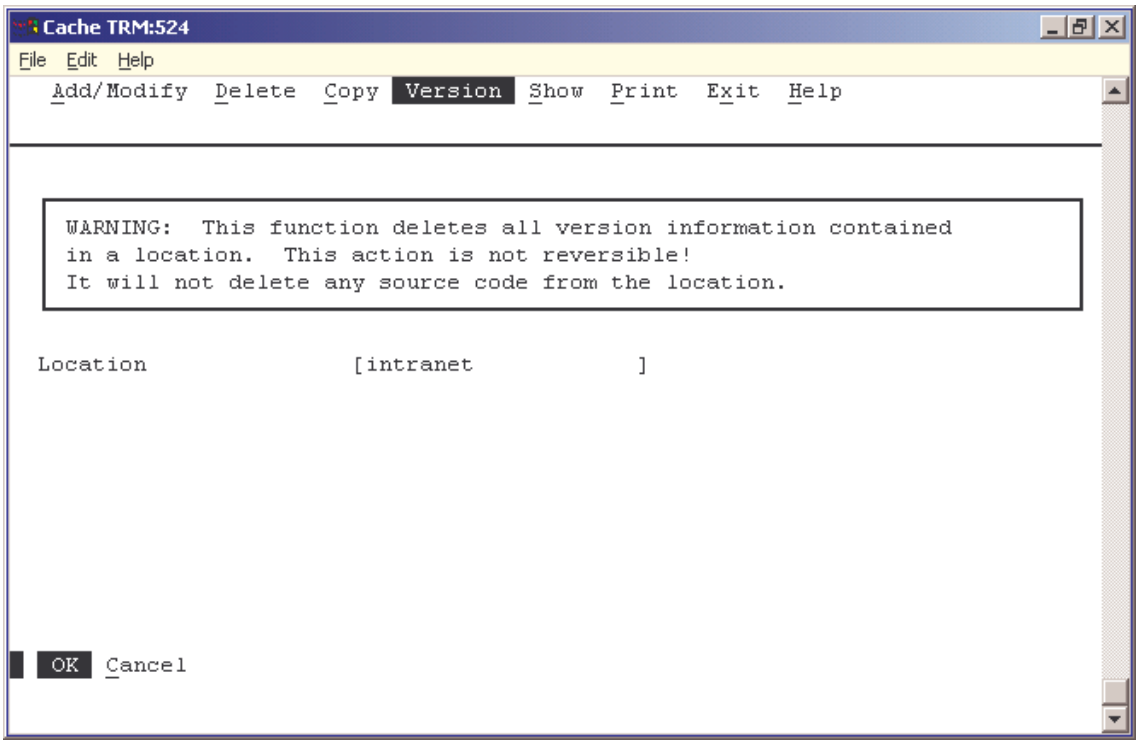


Figure 2.24 Delete version information from a location

Module: Module maintenance

This file stores information on modules. A module provides a way of grouping objects together. It can create a principled name for a new object in the group, and it can generate a skeleton M routine or routines for the object.

Add /Modify: Add a module or modify an existing one

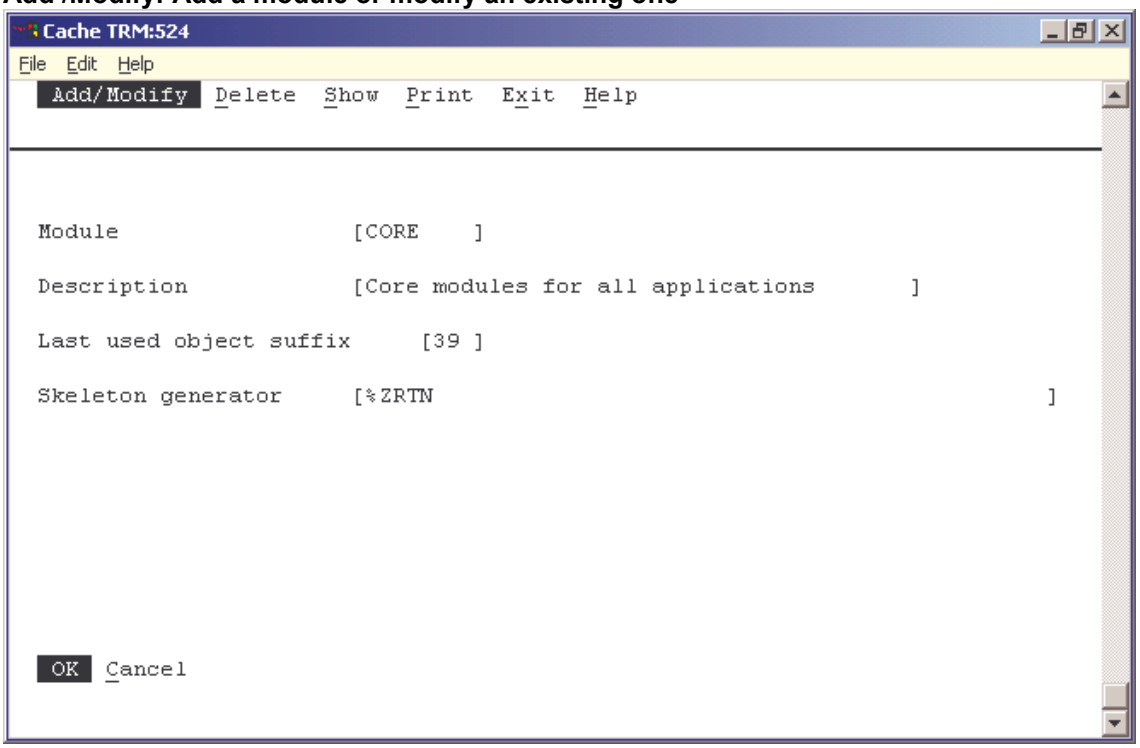


Figure 2.25 Add a module or modify an existing one

- Module

  Enter the identifier, usually a prefix, for the module.

- Description

  Enter a description for the module

- Last used object suffix

  Each time a new object is created, the next object name can be allocated automatically from the module file. The module identifier is used as a prefix. The last used object suffix is then incremented and appended to the module identifier to generate a unique object name. For example, if the module identifier is CORE and the last-used suffix was AAF, the next object name defaults to COREAAG.

- Skeleton generator

  A module can have a routine which generates a skeleton M routine whenever a new object is required. When a routine which does not already exist is added to an object, the appropriate skeleton routine generator is invoked. Routine %vc380 is supplied as an example of how to write a skeleton routine generator for your environment.

Delete: Delete a module

This option is used to remove the definition of a module from the VC/m system. It should normally only be used for modules which were set up during testing.

Object: Object maintenance and component registration

The object file stores information on each object version, including its components and which systems it belongs to. This registration and maintenance function allows the user, for example, to create new objects, or to move components between different objects.

When changes are made, they do not automatically change the physical contents of the locations. For example, making a change in the systems which an object belongs to or the components which belong to an object version does not immediately affect the components which are at the location. The object version must be transferred to the location(s) for the change in the components to take effect.

Add /Modify: Add an object or modify an existing one

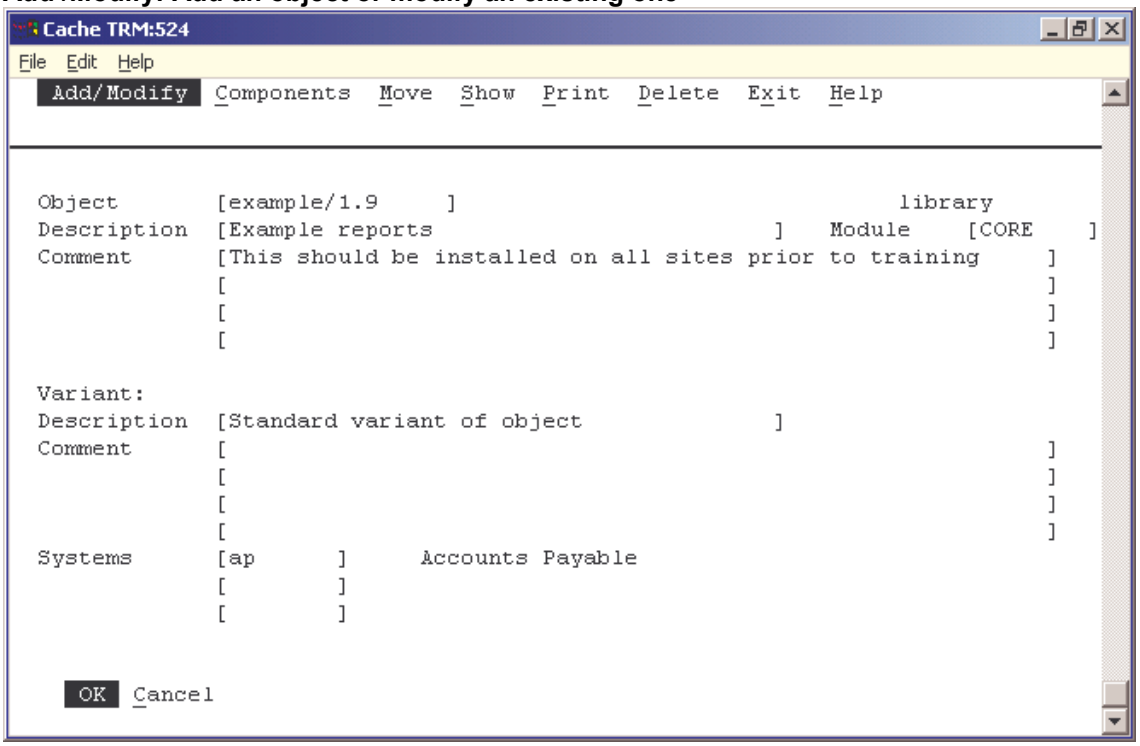


Figure 2.26 Add an object or modify an existing one

- Object

  Enter the object version.

  If the version number is not entered, it will default to the highest version of the selected variant.

  If the variant is not entered, and there is only one variant for the selected object, it will default to that variant.

  A selection list of object versions in VC/m is available.

  If a module identifier is entered, the next available object name for that module will be allocated, based on the algorithm specified in the module master file.

  When <return> is pressed, the master location of the object version is displayed.

- Description (Object base)

  Enter a description of the object base. Note that this will apply to all versions and variants of the object.

- Module

  If the object belongs to a module, enter the module code. A module can be used to auto-allocate the next object name as a suffix to the module code. It can also be used to identify the skeleton generator to invoke when a new routine is created and added to the object. This field is optional.

- Comment (Object base)

  Enter notes about the object base in this text field. This is particularly useful for notes for a user who is installing at a remote location, since any information in this field will be printed on the release control sheet (unless it is overridden at the variant level).

- Description (Object variant)

  A description of the object variant may be entered in this field. If nothing is entered, it will default to the object base description.

- Comment (Object variant)

  Enter notes about the object variant that may be of interest or importance to a user who is installing the object in a remote location.

  Any information in this field will be printed on the release control sheet when this variant is released, to provide information and guidance to the person responsible for installing it.

- System

  If systems are used, enter the system or systems to which the object belongs.

  Note: This applies to all versions of the object, not just the one which is being edited.

If the object is a new one, it is given a nominal location of NEW. When the details have been completed, the check-out screen is displayed. The object should be checked out to a development location so that it can be worked on and components can be added. The third screen which is invoked is the component maintenance screen.

If the object is not a new one, the second screen which is invoked is the component maintenance screen.

Components: Add or modify components of an object

The component maintenance screen refers to the master copy of the object version. It lists all the components of the object version, along with their component type.

The component list cannot be changed while the object version is in a library location. To make changes, the object version must first be checked out from the library.

When a component is added, it should physically exist in the master location of the object version. Alternatively, it should be created immediately after it is added, so that it is not forgotten.

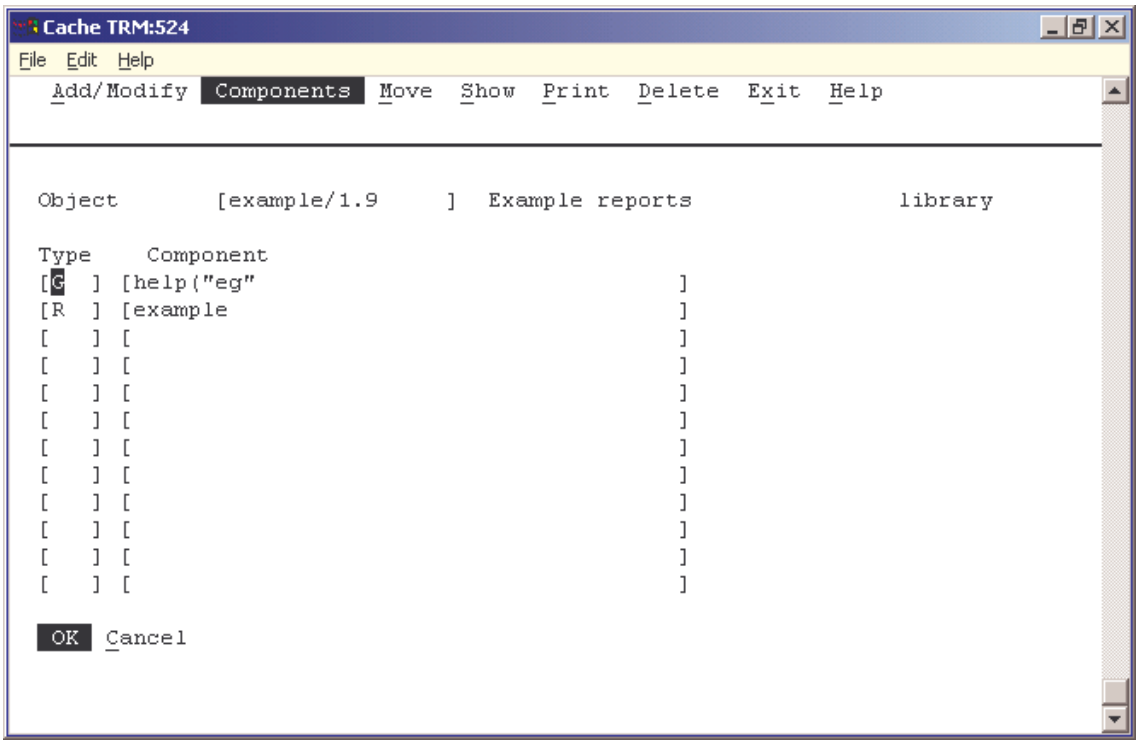


Figure 2.27 Add or modify components of an object

- Object

  Enter the object version.

  If the version number is not entered, it will default to the highest version of the selected variant.

  If the variant is not entered, and there is only one variant for the selected object, it will default to that variant.

  A selection list of object versions in VC/m is available.

  When <return> is pressed, the description of the object and the master location of the object version is displayed.

- Type

  Enter the code for the component type. A selection list of component types defined for VC/m is available.

- Component

  Enter the name of each component that belongs to the object version.

  If the component is of type T or BIN, it must include the complete path from the physical location to the actual file.

  If the component is of type R, it may be entered either with or without a preceding up-arrow. If the object belongs to a module which has a skeleton routine generator, this will be called when a new routine is added to the object, unless the routine already exists.

  If the component is of type G, it can be entered with or without a leading up-arrow. Subscripts can be entered with or without being enclosed in quotes and the final close bracket may be entered or omitted. Whatever format is used to enter the global reference, it will be transformed into a standard format which is used for display wherever it is required.

  Warning: If a component is deleted from the component list, it will be physically deleted from the master location, subject to user confirmation.

Note: The actual component versions which belong to an object version may be different at each location where it is active. They will only be the same when the version is transferred from the master location to all the other locations where it is active.

Move: Move a component from one object to another

The Move option enables all or some of the components of an object to be moved to a different object.

If components are moved from one object to another, the new object will automatically displace the old object when it is transferred.

Delete: Delete an object

This option is used to delete an object version which was created in error. The object version can only be deleted if it has not been checked out or transferred to any location. The audit trail is not deleted.

Physical Location: Physical location maintenance

The physical location file defines the attributes of the areas where the software is physically stored. Physical locations can also be defined on the logical location maintenance screen.

Add /Modify: Add a physical location or modify an existing one

Figure 2.28 Add a physical location or modify an existing one

- Physical Location

  Enter a name to identify the physical location

- Description

  Enter a description of the physical location

- Storage format

  The storage format code identifies the way in which components are stored in this location. A selection list is available.

  More detailed information can be found in the VC/m User Guide.

- Multiple versions?

  Each location is defined as storing a single version or multiple versions of each object. By their nature, some storage formats are only able to store a single version of an object, and so these should always be defined as single-version locations. Locations with a storage format of L or vss will typically be multi-version.

  More detailed information can be found in the VC/m User Guide.

- RE/m repository?

  If the location maps onto an RE/m repository location, VC/m will not update it directly, but will leave it to RE/m or some other process to maintain the location contents.

- Physical address

  Enter the physical address of the location being defined. The contents of this field depend upon the storage format, the host operating system and the M implementation on which VC/m is running.

  More detailed information can be found in the VC/m User Guide.

- Location type

  Controlled is the default and this should normally be used. Use uncontrolled if you want to transfer objects to a temporary location and do not want to formally manage, control or track the contents of that location.

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

  More detailed information can be found in the VC/m User Guide.

- Dependency check

  When there is a dependency check on a logical location which maps to this physical location, VC/m compares the versions of each component. This field determines whether the check is done by comparing VC/m's date /time stamp, or whether a complete physical compare is made.

Delete: Delete a physical location

This option can be used to remove a physical location from the VC/m system. This is only likely to be used, for example, to remove a physical location which was set up for test purposes.

Routes: Transfer route maintenance

This file stores details of the transfer routes which are defined. For each valid route, user access codes, dependencies and audit trail messages can also be defined.

Add /Modify: Add a transfer route or modify an existing one

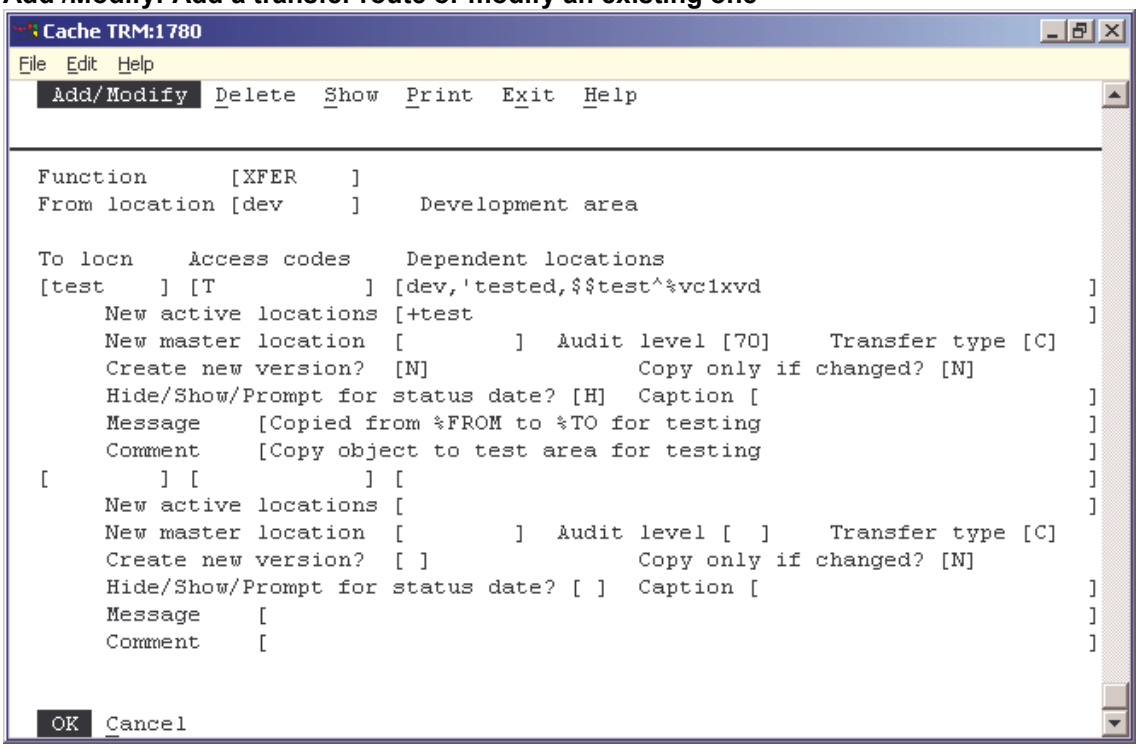


Figure 2.29 Add a transfer route or modify an existing one

- Function

  The function code identifies the type of transfer for which this transfer route is used. VC/m has a number of standard function codes, but installation-specific ones can also be used.

  More detailed information can be found in the VC/m User Guide.

- From location

  Enter the location code of the logical location from which objects will be transferred. The generic location %ANY can also be used.

  When <return> is pressed, the description of the logical location is displayed.

- To locn

  Enter the location code of the logical location to which objects will be transferred. The generic location %ANY can also be used.

  To delete the transfer route, enter a space in this field and <return>.

- Access codes

  Enter the access code or codes which a user must have in order to make the transfer.

  More detailed information can be found in the VC/m User Guide.

- Dependent locations

  Enter a list of location codes and location classes (prefixed by @) at which the object version must be active before the transfer is allowed. The generic locations %FROM and %TO can also be used. If the object version does not belong at the location (because of the system controls), the dependency will not need to be satisfied to permit the transfer.

  An installation-specific dependency can also be added in this field. This is in the form of an extrinsic function.

  More detailed information can be found in the VC/m User Guide.

- New active locations

  Enter a list of location codes and location classes (prefixed by @) at which the object version will have a status of active on completion of the transfer. The generic locations %FROM and %TO can also be used. The 'to location' should normally be included in this list.

  The list is incremental, i.e. the locations are made active in addition to any locations which were already active. Each location where the object version is to be made active is prefixed with a plus sign (+). Locations can also be de-activated by prefixing them with a minus sign (-).

- New master location

  Enter the location which will be the new master location when the transfer has been completed. The generic locations %FROM and %TO can also be used here. This field is optional. If it is not entered, the master location will remain unchanged.

  Note: Care should be taken if an absolute list of new active locations is entered, but a new master location is not entered. It is possible that the result would be an object which is not active at its master location!

- Audit level

  Each transfer route can be assigned an individual audit trail level. All audit trail entries for transfers which use this route will be given this audit trail level.

  More detailed information can be found in the VC/m User Guide.

- Transfer type

  Enter M to move or C to copy an object version.

  If an object is moved, it will be deleted from the 'from location' after successful transfer. Note that it will not be deleted from the 'from location' if that location still contains the master copy of the version.

  If copy is selected, the object will not be deleted from the 'from location' after transfer. If the 'from location' is not included in the list of new active locations, the version at the 'from location' will have a status of error (E).

- Create new version?

  Enter Y to increment the version number when this transfer is made, or N to leave the version number unchanged. The default value is N.

  Normally, the version number is incremented when an object version is checked out of the library to a development area. It is therefore recommended that this field is set to Y when the transfer has a function code of OUT.

- Copy only if changed?

  Enter Y or N. The default value is N.

If this is set to Y, 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.

More detailed information can be found in the VC/m User Guide.

- Hide/Show/Prompt for status date?

  This field controls whether the status date field is available when making a standard transfer using this route. Enter H (hide) if the status date should not be displayed at all. Enter S (show) if the status date should be displayed, but cannot be set or changed. Enter P (prompt) if the status date should be displayed and can be set or changed.

- Caption

  If S (show) or P (prompt) is entered in the previous field, this is the text which will be displayed at the prompt for status date. The default value is 'Release date'.

- Message

  This is the text which will be entered in the audit trail for transfers which use this route. Various % values can be used which will be substituted with the relevant information when the message is written to the audit trail.

  More detailed information can be found in the VC/m User Guide.

- Comment

  This field can be used to document the purpose of the transfer route. This is especially useful when complex systems of transfer routes are being used and the purpose of the route is not immediately obvious to another user.

Delete: Delete a transfer route

This option will delete all transfer routes with the specified function code and from location.

To delete only one transfer route, use the Add/Modify option, and enter a space in the field for the 'to location'.

System: System maintenance

This file defines which systems exist and which locations they are installed in. A system is a set of objects that are installed together to form a working set of software. If there are systems installed at a location, an object can only be transferred to the location if it belongs to at least one of the systems which are installed at the location. If there are no systems installed at the location, an object can only be transferred to the location if it does not belong to any systems.

Add /Modify: Add a system or modify an existing one

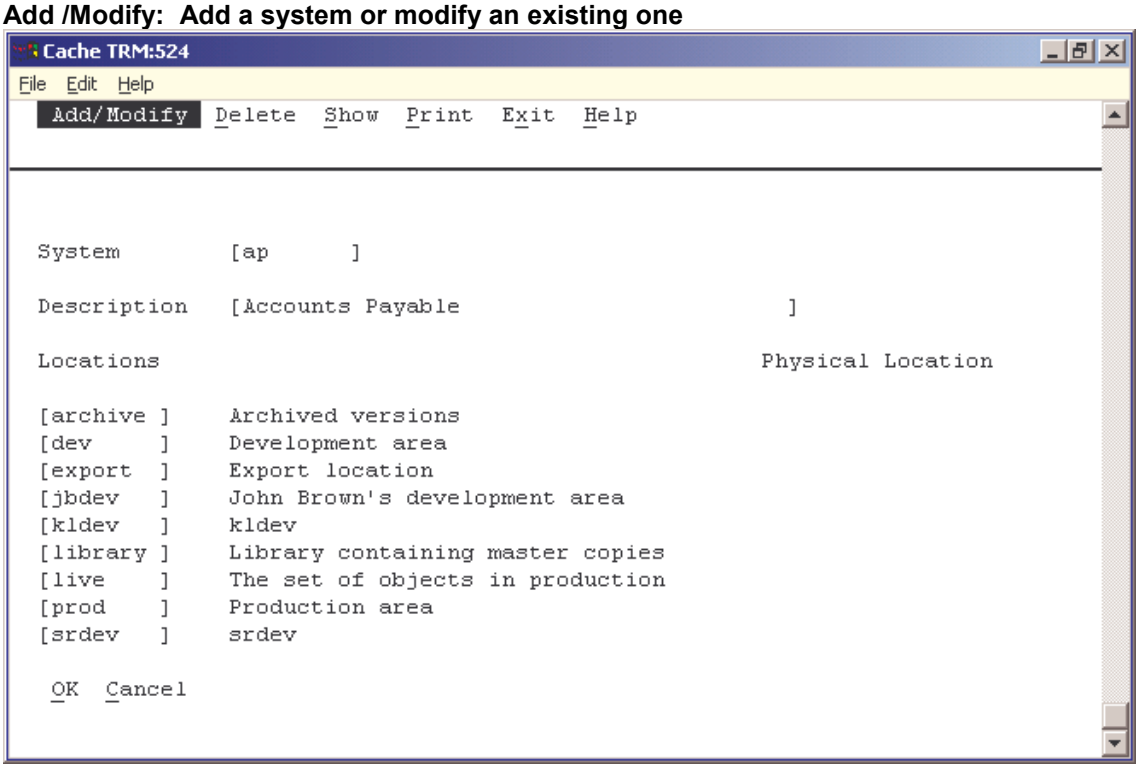


Figure 2.30 Add a system or modify an existing one

- System

  Enter the code to identify the system.

- Description

  Enter a description of the system.

- Locations

  Enter the codes for all the locations at which this system is installed.

  When <return> is pressed, the description of each location will be displayed.

Delete: Delete a system

This option can be used to remove a system from the VC/m database. Normally this would only be used for systems which were set up during testing.

Types: Change request type maintenance

This file is used to define the different types of change request which are to be used.

For each change request type, it is possible to define a list of statuses with associated sequence numbers. These can optionally have a tied location. When the status is changed, the objects in the change request are automatically transferred to the tied location. Transfer routes must therefore also be set up between the locations.

The types are also used by the transfer summary report when it groups the number of transfers by change request type.

Change requests can be categorized in many different ways. A typical scheme might be as follows:

Change Request Type	Description
D	Development projects
B	Bug fixes
C	Customization work
T	Language translation
X	Special projects

Add /Modify: Add a change request type or modify an existing one

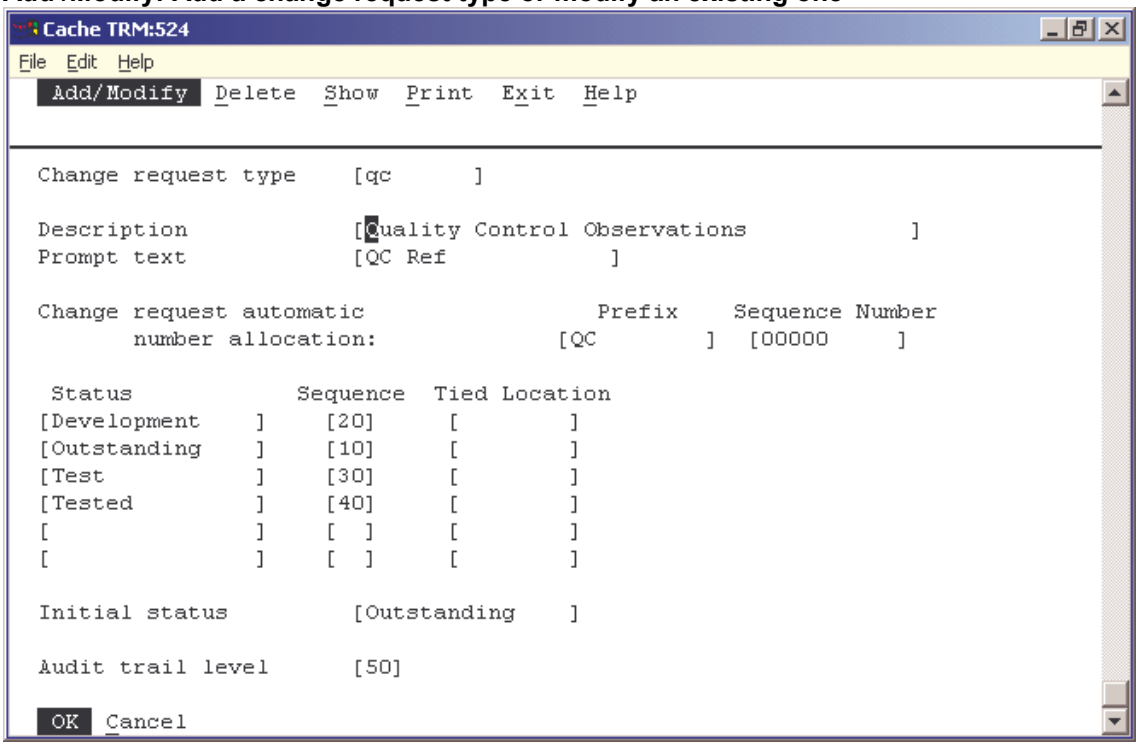


Figure 2.31 Add a change request type or modify an existing one

- Change request type

  Enter the code which is to be used to identify the change request type.

- Description

  Enter a description for the change request type.

- Prompt text

  A menu option which invokes Change request maintenance for this specific type of change request will use this text in place of the standard Change Request prompt.

- Prefix (Change request automatic number allocation)

  If you wish the change request codes to be allocated automatically in a numeric sequence, enter the prefix which is to be used, e.g. BC for all bug fixes.

- Sequence number (Change request automatic number allocation)

  If you wish the change request codes to be allocated automatically in a numeric sequence, enter the last used number of the sequence here, using the number of digits which are to be used. For example, if BC is entered in the previous field, and 0000 is entered in this field, the first change request code to be allocated will be BC0001.

- Status

  Change requests can optionally have statuses associated with them. Use these fields to define the possible statuses for this change request type.

- Sequence

  If statuses are used with this change request type, use these fields to define the order in which the statuses should be used. The values should be numbers, with the lower numbers coming earlier in the sequence.

- Tied Location

  A tied location can optionally be associated with a change request status. When the status of the change request is changed, the objects in the change request will be transferred to the tied location. Enter a location code.

- Initial status

  When change request statuses are used, this field determines the initial default status for a new change request of this type.

- Audit trail level

  Enter the audit trail level for messages for addition and removal of object versions for change requests of this type.

Delete: Delete a change request type

This option is used to remove a change request type from the VC/m system.

User: User maintenance

This file stores the list of users. Each user must have an entry with their user ID and access code(s).

Add /Modify: Add a user or modify an existing one

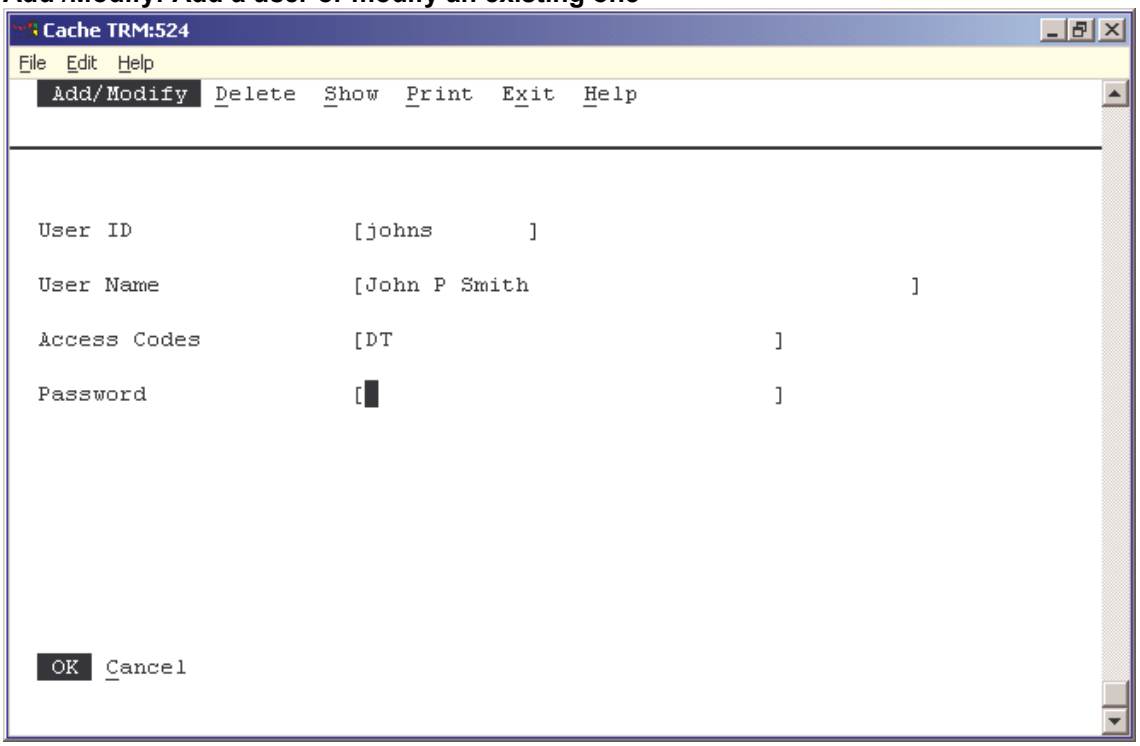


Figure 2.32 Add a user or modify an existing one

- User ID

  Enter the code (username) which the user will use to log on to the system.

  If VC/m is installed on Windows 95 or later, UNIX /Linux or OpenVMS, it is advisable to set up a user ID which corresponds to each user's operating system account name.

- User Name

  Enter the actual name of the user.

- Access Codes

  Enter one or more access codes. These determine which transfer routes the user can use. If the menu has been customized, it will also determine which menus and menu options the user has access to. If the browser interface is used, it will also determine which locations are displayed.

  More detailed information can be found in the VC/m User Guide.

- Password

  Enter the password for this user. This is primarily for the browser interface.

  Note: The password is not encrypted and will be re-displayed when this screen is called up. Do not use a password which will give access to other secure systems.

- Cancel objects checked out by other users?

  By default, only the user who checked an object version out is able to cancel the check-out. Set this to Yes to allow this user to cancel any check-outs.

Delete: Delete a user

This option is used to remove a user from the VC/m system.

2.6 Functions

Archive: Archive and purge old versions

The purge function is used to transfer object versions from a specific location to an archive location and then delete them from the former location. It selects all object versions which have been superseded by a specified number of later versions and were superseded before the specified cut-off date. The user specifies both the number of later versions and the period of time. A transfer route with function code XFER must first have been set up.

To enable the archive function to identify versions which are eligible for purging, it is recommended that, as old versions are superseded at a single-version location, e.g. LIVE, an automatic status change function is used to move them to a specific virtual location (e.g. OLD). The archive function can then be run in this location and will thus purge only those versions that have been superseded.

The archive location can be any location defined to VC/m. For example, it could be:

- A directory that can contain sequential files.
- A VC/m library, perhaps on a dismountable disk volume.

If an archived object needs to be retrieved, it can be transferred from the archive location back to any permitted location using the general purpose transfer function, providing a transfer route is first defined. If the complete contents of an archive file are required, a volume transfer function can be used.

Default values can be set up under 'Archive and Purge' on the second page of the Configure option in VC/m Set-up.

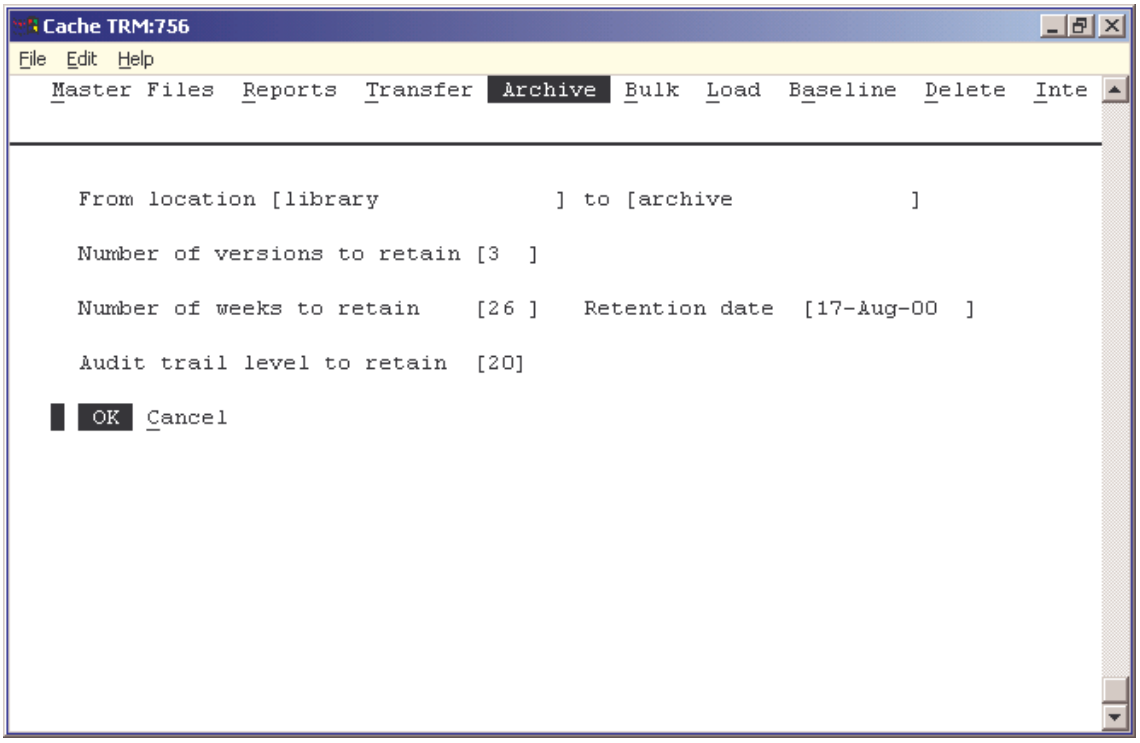


Figure 2.33 Archive and purge old versions

- From Location

  Enter the location which is to be purged. The default can be configured in VC/m Set-up.

- To Location

  Enter the location to which the versions should be archived before they are purged. The default can be configured in VC/m Set-up.

- Number of versions to retain

  Enter the number of versions of each object which should be kept by VC/m. Only versions that have this number of successors will be selected for archiving. The default can be configured in VC/m Set-up.

  If zero is entered, all versions created on or before the retention date will be purged.

- Number of weeks to retain

  This field enables the cut-off date for the archive and purge process to be specified. Any version which was superseded less than the specified number of weeks previously will not be archived and purged, even if it has the requisite number of successor versions. The default can be configured in VC/m Set-up.

- Retention date

  If the previous field is entered, the retention date is calculated and presented as a default. If the previous field is not entered, the retention date can be specified as an actual date.

- Audit trail level to retain

  When objects are archived, their audit trail details can be selectively purged so that the minimum of audit trail information is retained. Any entries with a value higher than the retention level will be transferred to the archive along with the object version they refer to. If an object is recovered from the archive, its audit trail will be recovered with it.

Baseline: Create a new baseline

This function is used to create a new variant from a selected set of object versions, rename the new objects as version zero of the new variant and copy them to a new location. This creates a baseline for development of the new variant.

For example:

LIB	Selection Set	DEV
ABC/1.3		
ABC/1.4		
ABC/1.5	ABC/1.5	ABC/5.0
DEF/2.0		
DEF/2.1	DEF/2.1	DEF/5.0
DEF/3.3	DEF/3.3	
GHI/2.6	GHI/2.6	GHI/5.0
JKL/2.3	JKL/2.3	JKL/5.0

In this example, the highest version of each object at the LIB location has been selected and used to create a baseline for variant 5 at the DEV location.

If a new variant is created in a multi-version location, it will not displace any objects which already exist there. A library location, for example, can therefore be used as both the 'from' and 'to location'.

> *Note: Because a library location does not store multiple copies of components which are identical, there is very little disk space overhead in the creation of a new baseline variant in the library.*

If a new variant is created in a single version location, it will displace any version of the object which already exists at that location. The displaced object will be removed from the location.

The following conditions must be satisfied before VC/m will create an object for a new baseline variant:

- An object with the new variant code must not already exist anywhere.
- The object must belong in a system which is installed at the new location.
- The creation of the new variant must not cause the master copy of any existing object to be displaced at the 'to location'.
- The creation of the new variant must not displace a component which exists at the 'to location' but is unknown to VC/m.

Figure 2.34 Create a new baseline

- Object(s)

  Enter the object versions to be selected. They can be selected and de-selected in the standard ways.

  More detailed information can be found on p. 27.

  A selection list of the objects in VC/m is available.

Bulk: Transfer multiple objects between locations

The bulk transfer function is used to make a volume transfer over a transfer route with function code XFER. This transfer route must first have been set up. The route defines the exact processing which will take place.

More detailed information can be found in the VC/m User Guide.

A transfer can be invoked for all or selected object versions at a location.

The bulk transfer function provides a general purpose transfer which can be used, for example, to populate a new location. It is useful both for one-offs and for transfers which are made on a regular basis.

> *Note: Before transferring to a brand new location, ensure that all the relevant systems have been added to the location.*

Figure 2.35 Transfer multiple objects between locations

- From location

  Enter the code of the location from which objects should be transferred.

- Release date

  If a date is entered, only those objects which have the same or an earlier status date will be selected. In addition, any objects at the selected location which do not have a specified status date will be selected.

  If no date is entered, all the object versions at the location will be selected for transfer.

  When <return> is pressed, the number of objects which match the selection criteria will be displayed.

- Object(s)

  Initially all the object versions which match the selection criteria are selected. The object versions from this initial list can be selected and de-selected in the standard ways.

More detailed information can be found on p. 27.

A selection list of the objects which match the selection criteria is available.

- To location class

  Enter the location class to which the selected objects are to be transferred.

  If a location class is not selected, a specific location must be entered in the next field.

- To location

  If a location class is not entered, a single location must be entered here. This enables a transfer to a single location to be performed.

Cancel: Cancel an object checked-out in error

This function deletes all copies of an object version at all locations and replaces 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.

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 normally only be canceled by the user who created it. This prevents accidental deletion of another user's work. Certain users can be give the right to cancel any object version, regardless of who created it.

If version 0 of an object is canceled, all information about that particular variant will be deleted. However, there will still be a record of its existence in the audit trail.

> *Note: The cancel function should not be used on an object version which has actually gone live, because it will delete all copies of the object. Any code which has caused a problem will then not be available for examination. In such circumstances the preceding version should be checked out as a new version and made live through the normal procedures.*

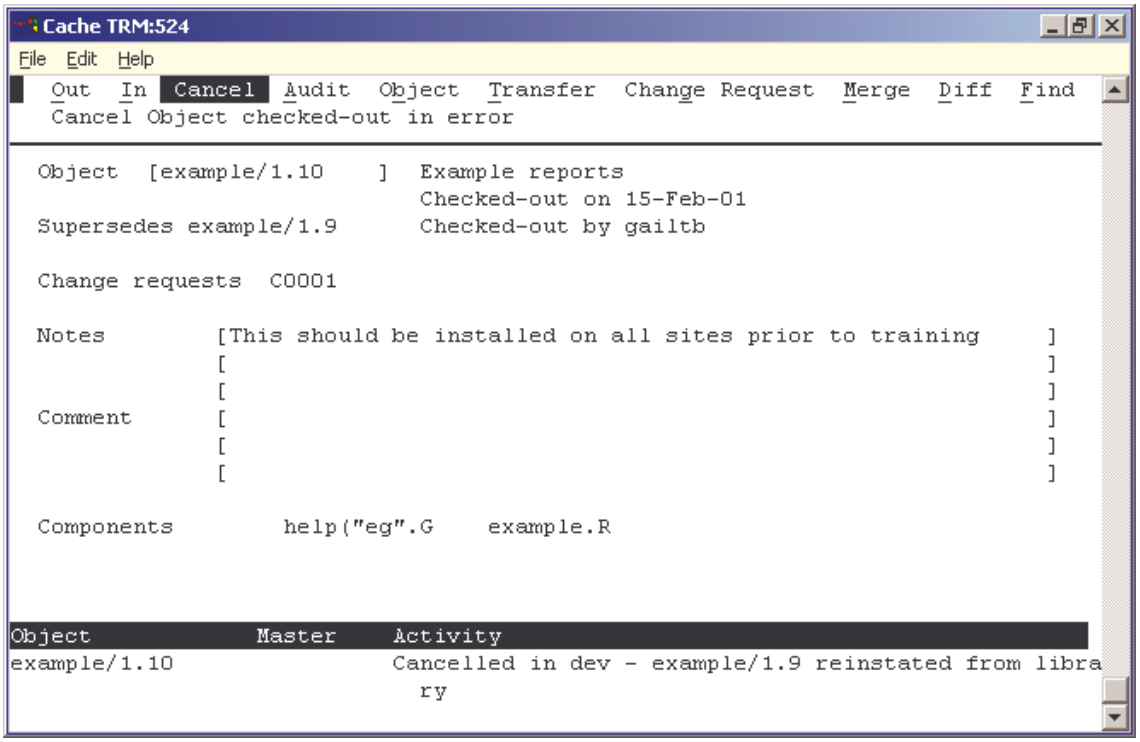


Figure 2.36 Cancel an object checked-out in error

In: Check in an object

The check-in function is used to make a standard transfer over a transfer route with function code IN. This transfer route must first have been set up. The route defines the exact processing which will take place.

More detailed information can be found in the VC/m User Guide.

A transfer can be invoked for a single object version or for a change request.

By convention, the main use of the check-in function is to transfer the master copy of an object version back to a library location after changes to it have been completed and tested. Because object versions cannot be directly edited when they are in the library, this effectively prevents any further changes.

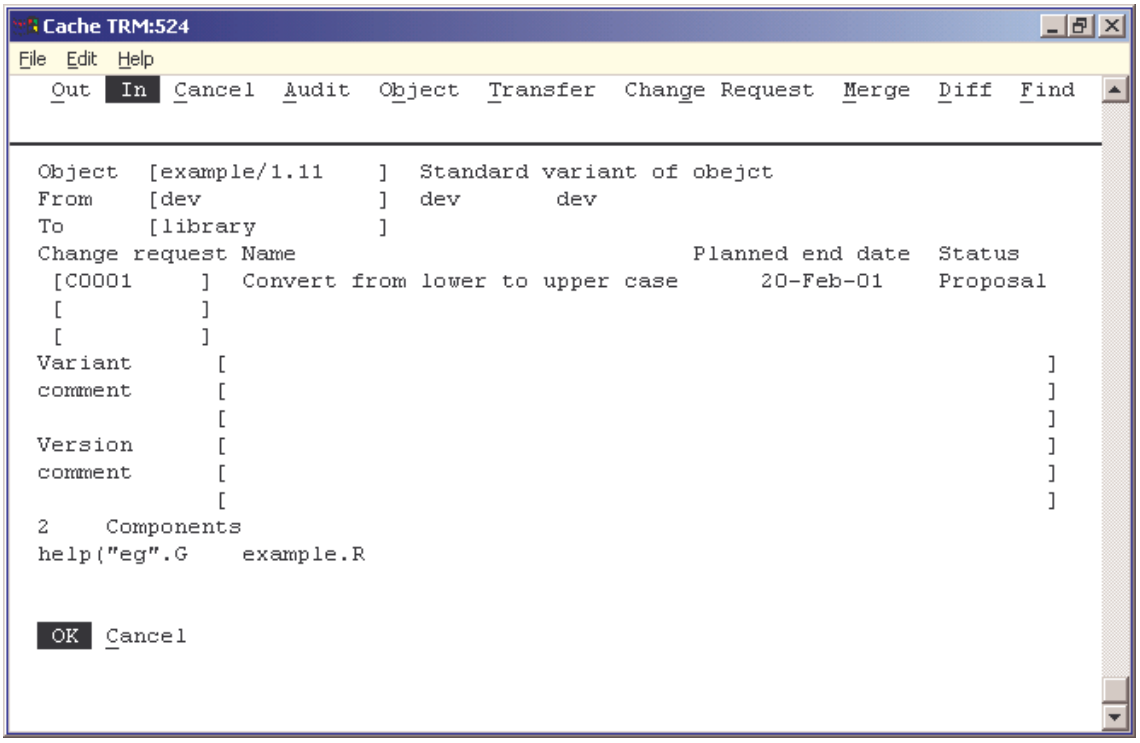


Figure 2.37 Check in an object

For details of how to answer the standard transfer prompts, see p. 94.

Install: Install objects from a sequential file

The install function is used to install objects at a remote location. It reads all the sequential files destined for that location and installs the objects which they contain.

Before using the install function in any location, the location must have been set up under the Install set-up option in VC/m Set-up. Install may only be used to read from sequential files, but it can install them into a physical location of any type.

When the install function is used, it initially searches for and lists the first ten transfer files in the holding area which are destined for the current location. In each case, brief details of each file are displayed to enable the user to decide which file or files should be installed.

Once a transfer file has been selected, the contents of the file can be reviewed and individual objects selected or de-selected for installation.

If only a sub-set of the objects are selected for installation, the transfer file is not deleted on completion of the installation. If all objects are selected, and the installation is successful, the transfer file is deleted.

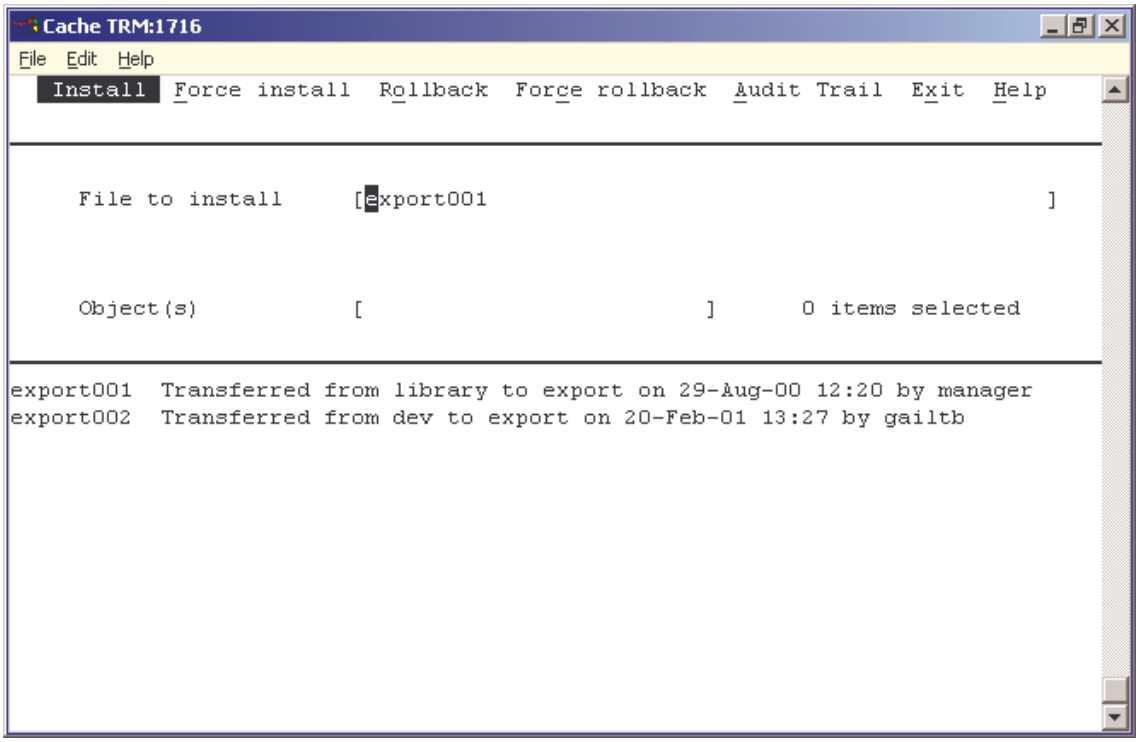


Figure 2.38 Install objects from a sequential file

- File to Install

  Enter the name of the file to be installed. The selected file does not have to be displayed on the screen in order to be selected. However, only those files which are destined for the current location are valid.

  Details of the selected file will be displayed for visual confirmation.

- Object(s)

  Once a transfer file has been selected, it can be reviewed. By default, all objects in the transfer file will be installed. They can be selected and de-selected in the standard ways.

  More detailed information can be found on p. 27.

A selection list of the objects in VC/m is available.

- Report Destination

  The installation process will print a report to positively confirm that the objects have been installed. Enter the printer to print the report on.

- Processing

  Each object in the transfer file will be installed using the standard transfer mechanism. The process checks that each object is overwriting the expected ancestor (or itself, in the case of a re-transfer).

  If the object is valid for installation, the old version of the object will be deleted and the new version installed.

  The local configuration management database is updated to record the version of the object that has been installed at this location, and a record is also written to the audit trail.

If any object cannot be installed, for example, because a higher version already exists there, none of the objects in the transfer file will be installed. If, upon inspection, it is determined that the file should be installed, despite evidence of problems, the Force Install option should be used to install the contents of the file.

Force Install: Force installation with overrides

As each invalid object in the transfer file is processed, the reason is displayed and the user is prompted. The user should then confirm whether or not the object should be installed. Whichever response is given, the audit trail is updated with a record of what happened.

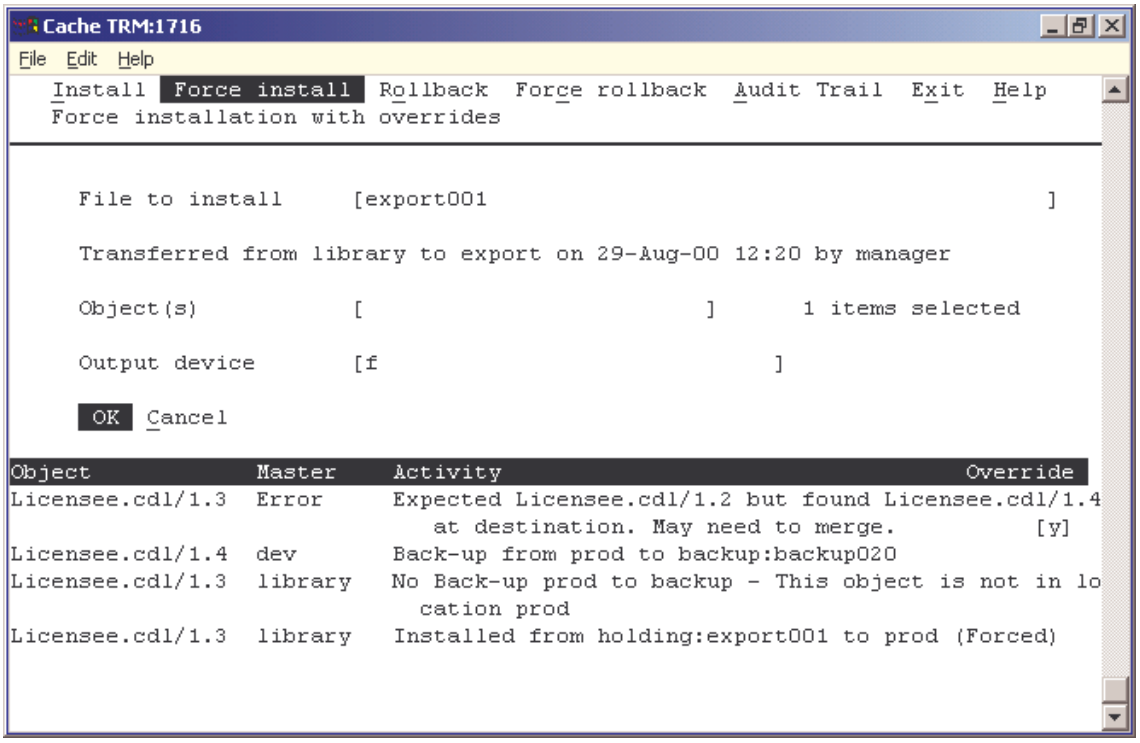


Figure 2.39 Force installation with overrides

Load: Bulk component registration

This function is used to register routines to VC/m so that they are added to its database. It is most often used to register in a library location. The components must be added to an object when they are registered.

When VC/m is installed, Load can be used as a quick way of declaring the components and creating objects. It can also be used to take on a location which was not previously under VC/m control.

The components to be registered must be in a VC/m location. VC/m compares each component with the components of each master version of the object which have already been registered to it. It does not compare with objects which have the same object base but a different variant. The objects which are compared do not need to be in the location which is being registered to.

If a version of the component which is exactly the same as the one being registered already exists in an object version at any location, that component is not re-registered into a library location.

If a component already exists in an object version at any location, but is not the same as any version of the component which is there, a new version of the object is created and the new components are copied into the 'load to location' under the new object version number. This copy is the master copy of the new version.

If the components do not already exist at any location, a new object is created using the first few characters of the component name and the default variant. The components are copied into the 'load to location' as the first version of this new object. This copy is the master copy of the new object version.

All object versions which are created can be linked to a single change request so that they can subsequently be referred to as a single unit. This change request must already have been set up.

Figure 2.40 Bulk registration of components to objects

- Load from location

  Enter the location code of the location where the components currently are.

- Routine(s)

  Enter the names of the components to be registered. This prompt allows components to be selected and de-selected in the same standard ways as objects.

  More detailed information can be found on p. 27.

- Number of characters to match

  If you want to group several components together into one object, enter the number of characters from the beginning of the component name which should be used to form the object's name. This is useful when one logical object has been split into several components for size or structural reasons. For example, if a suite of routines consists of ABC120, ABC121 and ABC122, matching on the first 5 characters would link this set of routines to an object called ABC12.

  This algorithm is only used if the components do not already exist at any location. If previous versions exist at any location, the new components will use the component to object mapping that is already established.

- Load to Location

  Enter the location code of the 'to location'. The object will be transferred to this location and made master at the location.

- Active Locations

  Enter a list of locations at which the object should be made active once it has been created.

- Default system

  This prompt allows you to define which system the new objects will belong to.

- Default variant

  This prompt defines the variant that will be used for new objects which are created.

- Change request

  All new objects or versions which are created will be linked to the change request entered at this prompt.

Merge: Merge two versions of an object

This function is used to inform VC/m that the changes to two versions of an object, which were being developed concurrently, have been merged.

VC/m will only allow a version to be transferred to a single-version location if the version which it displaces is an ancestor of the new version.

When two versions of an object have been checked-out simultaneously, changes made to the first version that is checked-in must be incorporated into the second version. The actual merging of the software changes is a manual process. The merge function is then used to inform VC/m that a merge has taken place. VC/m gives the second version a new version number which is higher than the first version, and makes that version its ancestor.

More detailed information can be found in the VC/m User Guide.

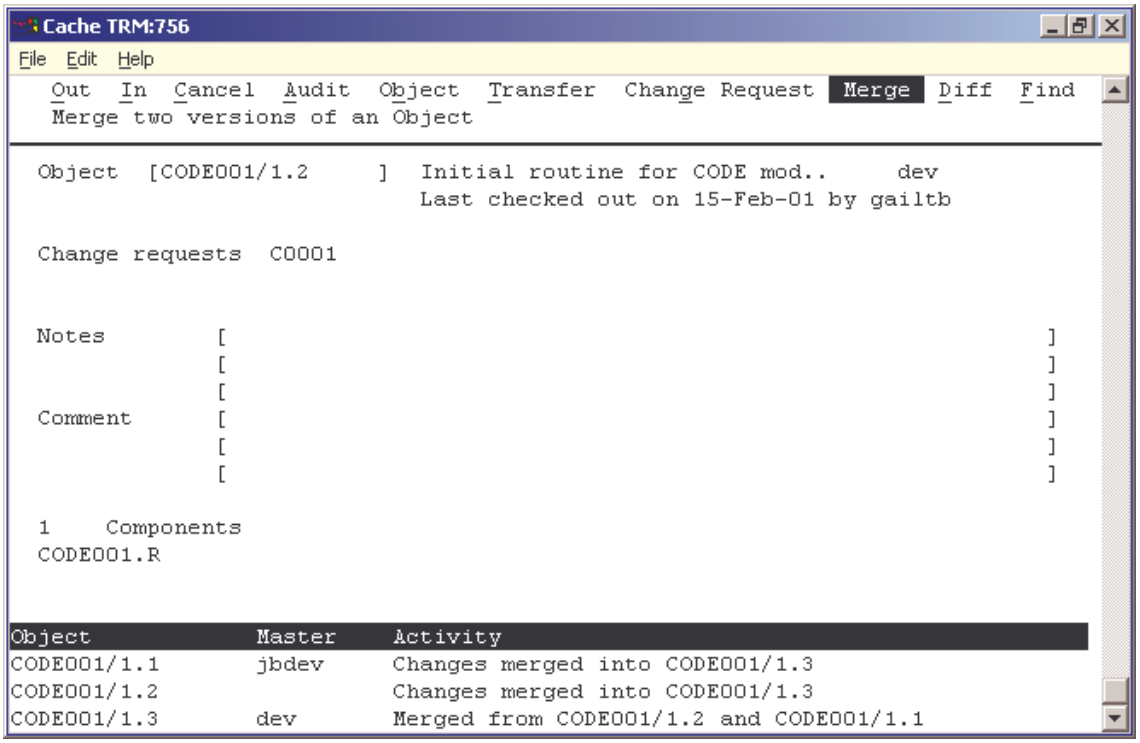


Figure 2.41 Merge two versions of an object

- Object

  Enter the object version which has the merged version of the code. This must be in an editable location. This version will be the one which is used to create the new, combined object.

- Merge in version

  Enter the version which is to be replaced with the new version.

Out: Check out an object

The check-out function is used to make a standard transfer over a transfer route with function code OUT. This transfer route must first have been set up. The route defines the exact processing which will take place.

More detailed information can be found in the VC/m User Guide.

A transfer can be invoked for a single object version or for a change request.

By convention, the check-out function is used to copy the master copy of an object out of a library for development work. An increment in the version number serves to identify the new version which is being developed.

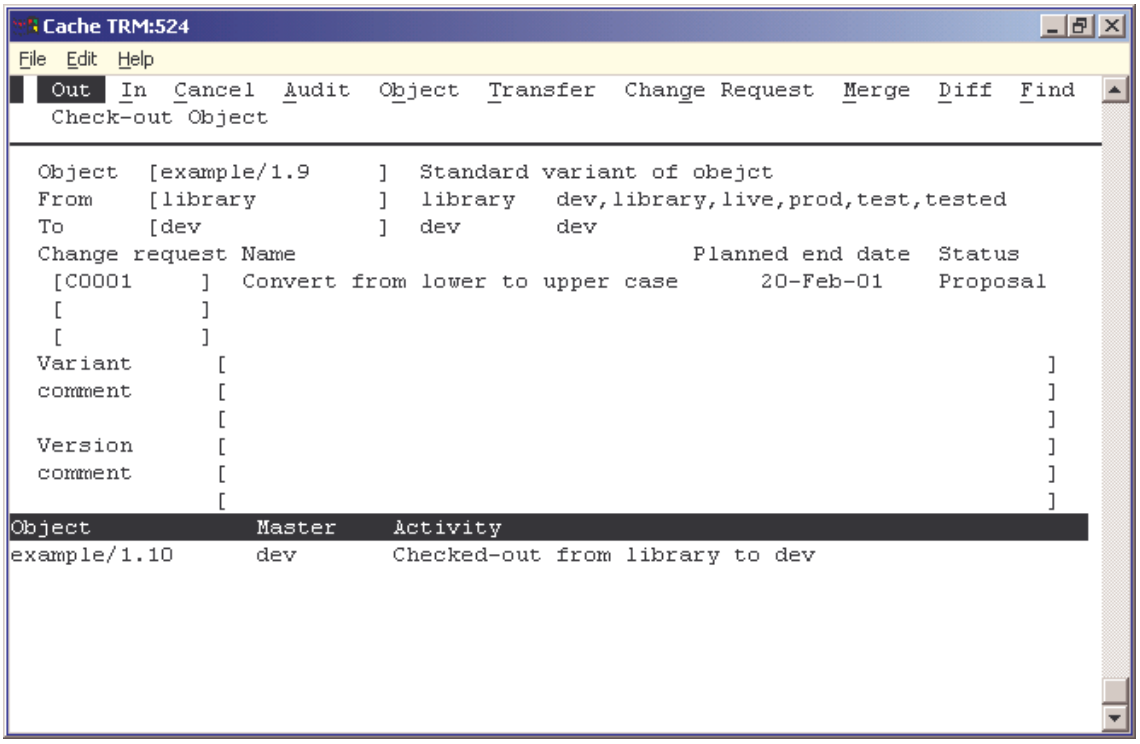


Figure 2.42 Check out an object

For details of how to answer the standard transfer prompts, see p. 94.

Release: Release objects to one or more locations

The release function is used to make a volume transfer over a transfer route with function code XFER. This transfer route must first have been set up. The route defines the exact processing which will take place.

More detailed information can be found in the VC/m User Guide.

The release function provides a semi-automated function which is used to release specified object versions to the live location or locations. It is designed to be used as part of a standard or repetitive procedure where manual intervention and operator error need to be reduced to a minimum.

The selection process selects all active object versions which are at the specified location and have a current status date.

The values for status function enabled or disabled, from location, to location or location class and report destination are defined under 'Automatic Release' on screen 4 of the Properties option in VC/m Set-up. These values cannot be changed, with the exception of the report destination.

A release control sheet is printed for each 'to location', listing each object version which is released and the object notes for each one. This report can be used by the person installing the objects, particularly at remote locations, as a checklist.

This release control sheet can also be printed only as a report, in advance of the release, so that the contents of the release can be checked before it is actually made.

For details of how to answer the volume transfer prompts, see p. 81.

Rollback: Roll back objects from a backup file

When the install function is configured with a hot backup location, any object versions that are overwritten during the installation process are copied to a backup file. The rollback function enables one or more of the object versions which have been replaced to be reinstated, for example, if the new version does not work as expected.

The rollback works by using the install function configuration parameters in reverse. It transfers the selected objects from the backup location to the original location in accordance with the rules defined for the transfer route.

A normal transfer from the backup location to the original location would be rejected because the objects at the backup location have a lower version number than those at the original location. Rollback enforces all the usual validation and protection, with the exception that the validation of the version sequence numbers is suppressed.

A complete backup file can be rolled back in one step, or specific objects can be selected from the backup file for individual rollback.

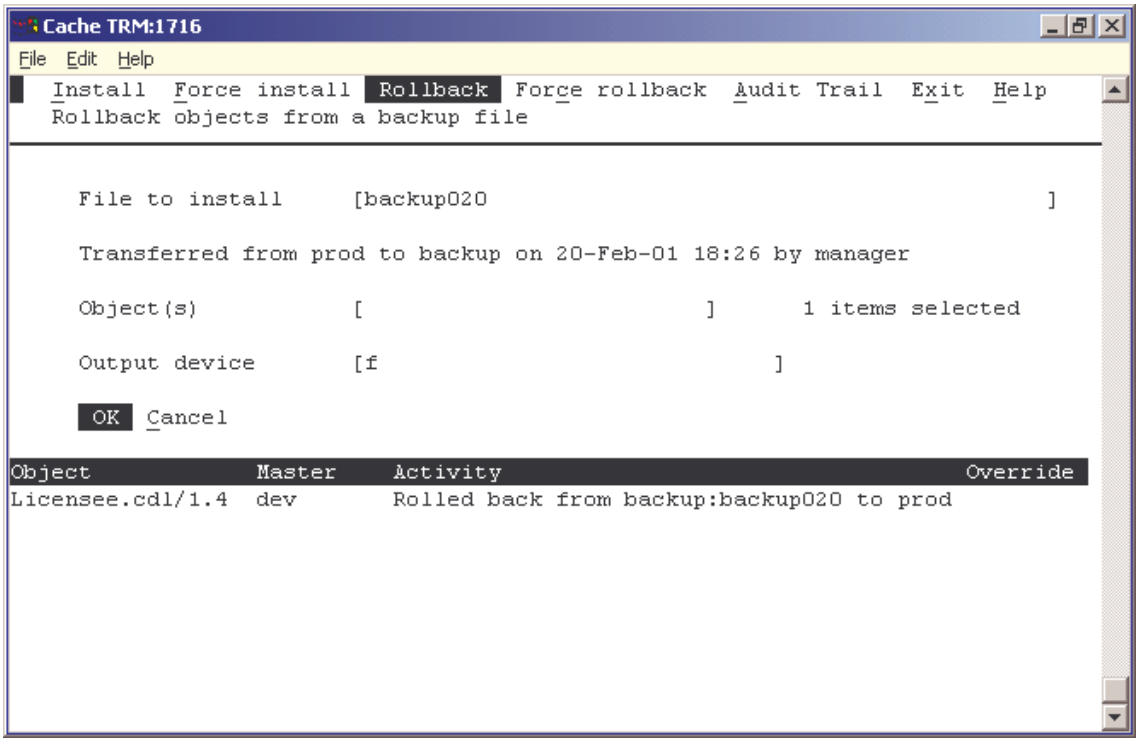


Figure 2.43 Roll back objects from a backup file

- File to install

  The rollback function displays a list of backup files in reverse date /time order to select from.

- Object(s)

  By default, all objects in the backup file will be selected for rollback. Objects can be selected and de-selected in the standard ways.

  More detailed information can be found on p. 27.

  A selection list of the objects in VC/m is available.

Force Rollback: Force rollback with overrides

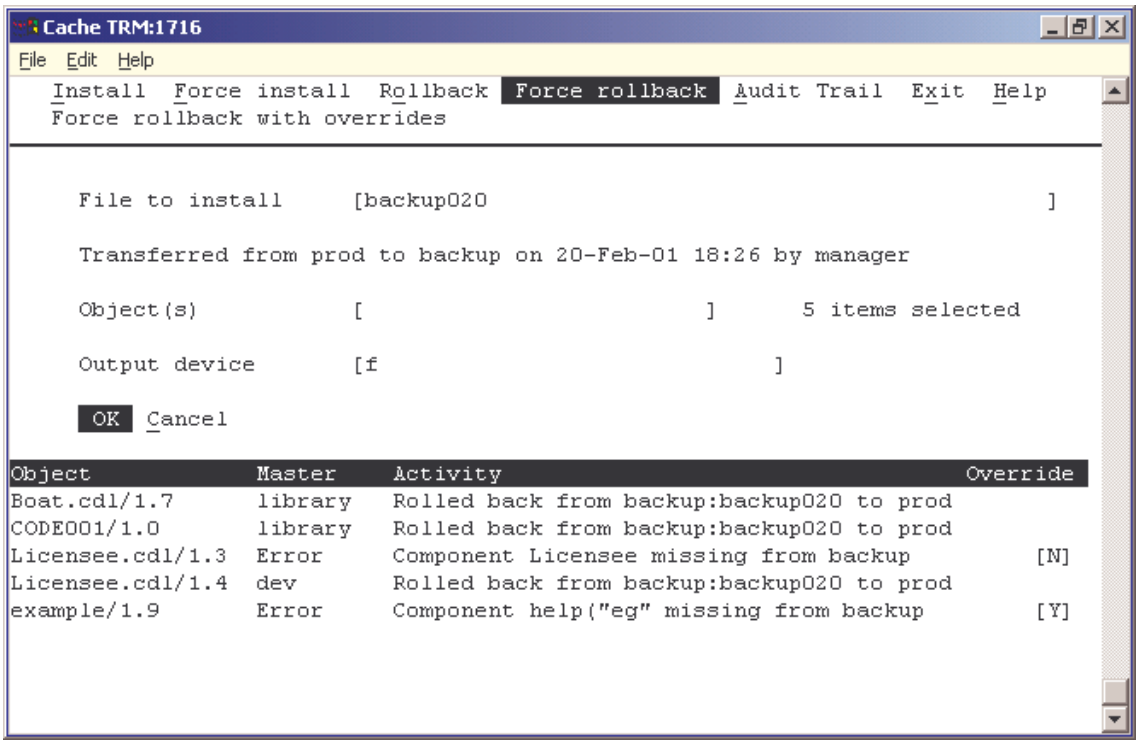


Figure 2.44 Force rollback with overrides

Transfer: Transfer an object between locations

The transfer function is used to make a standard transfer over a transfer route with function code XFER. This transfer route must first have been set up. The route defines the exact processing which will take place.

More detailed information can be found in the VC/m User Guide.

A transfer can be invoked for a single object version or for a change request.

The transfer function provides a general purpose transfer which can be used to move or copy an object version from any location to any other location.

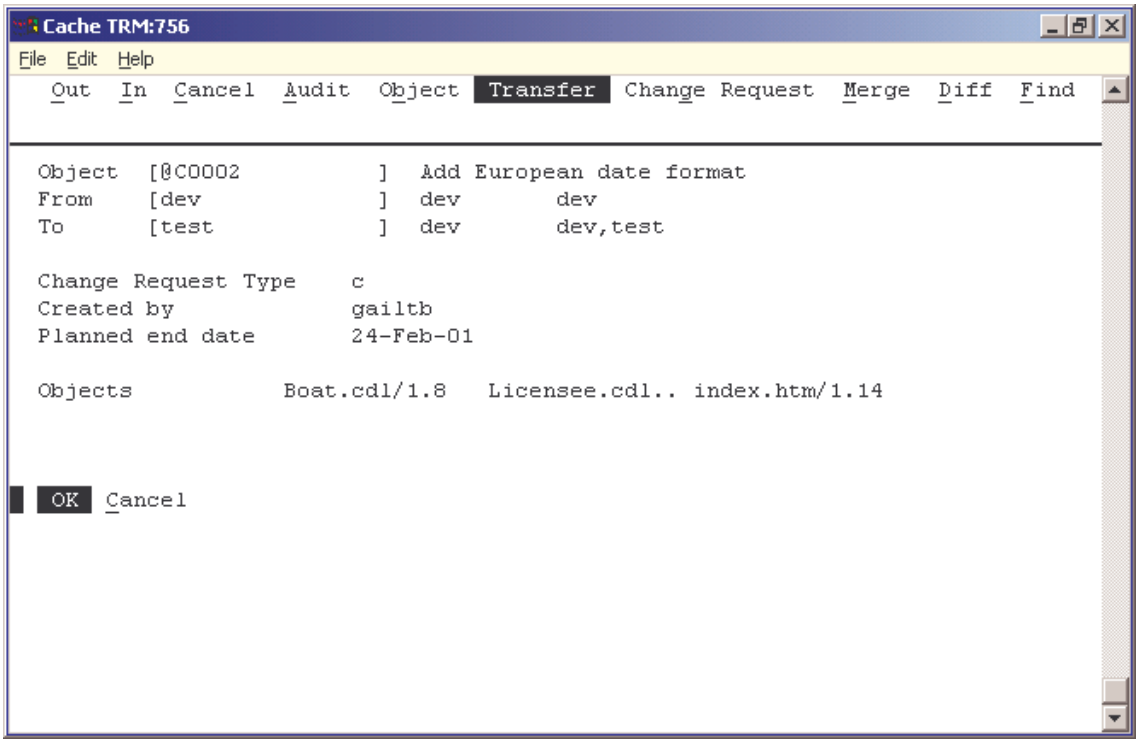


Figure 2.45 Transfer a change request between locations

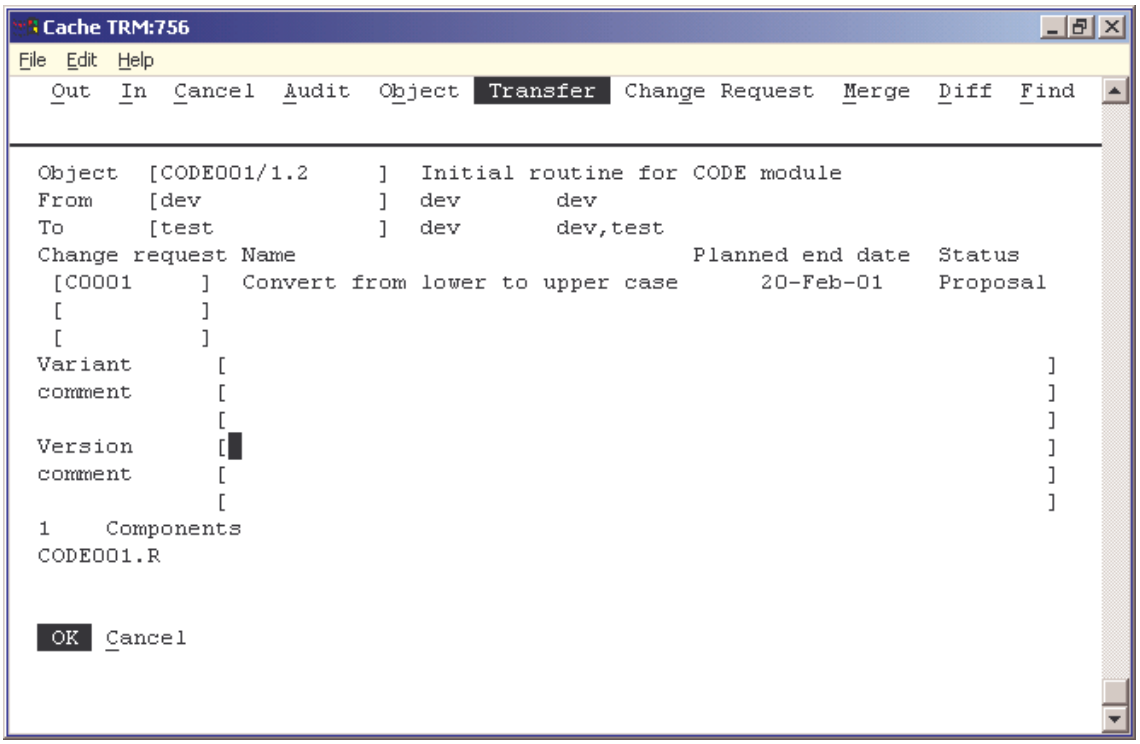


Figure 2.46 Transfer an object between locations

- Object /Change request

  To transfer by change request, enter the change request code, prefixed by @. A selection list of change requests can be obtained by entering @ and a help key.

  When <return> is pressed, the name of the change request is displayed beside the prompt and details are displayed below the input prompts. The screen is re-displayed with only the relevant prompts.

  To transfer by object version, enter the object version. A selection list of object versions is available.

  If the version number is not entered, it will default to the highest version of the selected variant. If the variant is not entered, and there is only one variant for the selected object, it will default to that variant. If the object has more than one variant, a selection list of the existing versions of each variant will be displayed.

  If a component name is entered, VC/m automatically selects the object which owns that component. In this case the component name has the following format: component type (in capitals), followed by a dot, followed by the actual component name, e.g. T.index.htm is the text component called index.htm.

  When <return> is pressed, the description of the object is displayed.

- From

  Enter the location code from which the change request or object version is to be transferred. Where possible, VC/m offers a default value. If the location of the master copy of the object version is a valid 'from location', this is the default. Otherwise, if the location in which VC/m is currently being used is a valid 'from location', this is the default. Finally, if there is only one valid 'from location', this is the default.

  If all object versions belonging to a change request are being transferred, the 'from location' can be left blank. The 'from location' will be taken as the master location of each object version. This can be used to transfer object versions that are at several different locations to one common location.

If the location maps onto a sequential file, the name of the transfer file must be entered as well as the location. For example, to transfer an object from an archive file ARC001 in location ARCHIVE, you must enter:

```
From        [ARCHIVE:ARC001            ]
```

When <return> is pressed, the current locations are displayed. The first column shows the master location and the second column shows all currently active locations. For an object version, a list of the components which belong to the object version at the 'from location' is displayed below the input prompts.

- To

  Enter the location to which the change request or object version is to be transferred. Where possible, VC/m offers a default value. If there is only one valid location to which the particular change request or object version can be transferred, this is the default. Otherwise, if the location in which VC/m is currently being used is a valid 'from location', this is the default.

  Transfers can be made to multiple locations by entering a location class code (prefixed by @). For example:

```
To          [@LIVE                     ]
```

  When <return> is pressed, the predicted locations after the transfer has been made are displayed. The first column shows the master location and the second column shows the active locations.

- Change request

  If the object version has already been linked to any change requests, they will be displayed as a list. If it has not been linked to any change requests, at least one must be entered in this field.

  A selection list of existing change requests is available.

  If a change request code which does not already exist, or a change request type, is entered, the change request maintenance screen is invoked to allow a new change request to be created.

  The change request maintenance screen can also be accessed by entering a help key in a field where there is already a value for the change request code.

  When <return> is pressed, the name, planned end date and status of the change request are displayed.

- Variant comment

  This is a free text field associated with the variant of the object. It can be used to describe the object and, for example, to highlight special installation instructions which may be required for this object. This field is printed on the release control sheet and will therefore be seen by anybody who installs this object in a remote location. If a variant comment is not entered, the object base comment will be printed on any release control sheet.

- Version comment

  This is a free text field associated with the current version of the object. It can be used to describe, for example, why the version was created and what has been changed.

- Status date

  Note: Whether or not this prompt is displayed depends on what is entered in the field for Hide /Show /Prompt for status date under Transfer route maintenance. The wording of the prompt may also vary depending on what is entered in the field for Caption under Transfer route maintenance.

  Use this field to link a date to the object version which is being transferred. This date can be used as a condition on a volume transfer.

2.7 Reports and Utilities

Audit: Audit trail

The Object option displays or prints the audit trail for any selected object. If a change request code is entered at the object prompt (preceded by @), the status and audit trail of all objects belonging to the change request will be shown.

For an object, the audit trail entries are shown in reverse chronological order. For a change request, they are shown sorted by object in the change request, in reverse chronological order for each object.

The Date option enables all audit trail entries for between two dates to be displayed in reverse chronological order.

The level of detail is selectable. Only those audit trail entries that have a level that is lower than the selected level will be displayed.

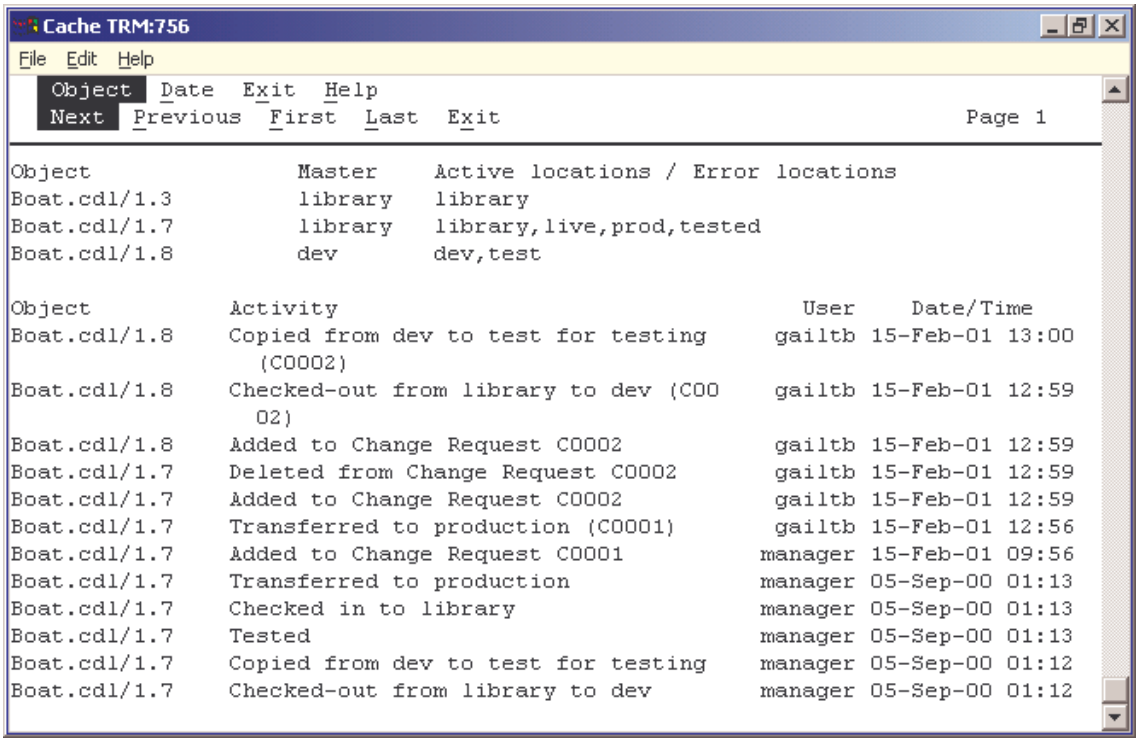


Figure 2.47 Audit trail by object

- Date /Time Stamp Inconsistencies

  If an object version in any location has a date /time stamp that is different from that of the master copy, an asterisk will be displayed against the location in the summary section of the audit trail inquiry. This usually occurs because one of the versions has been edited.

Concurrent: Concurrent development analysis

This report lists all objects which, between any two user-specified dates, have or had multiple versions checked-out for development concurrently. In each case, the associated change request(s) will be printed to enable the cause and urgency of the concurrent development to be assessed.

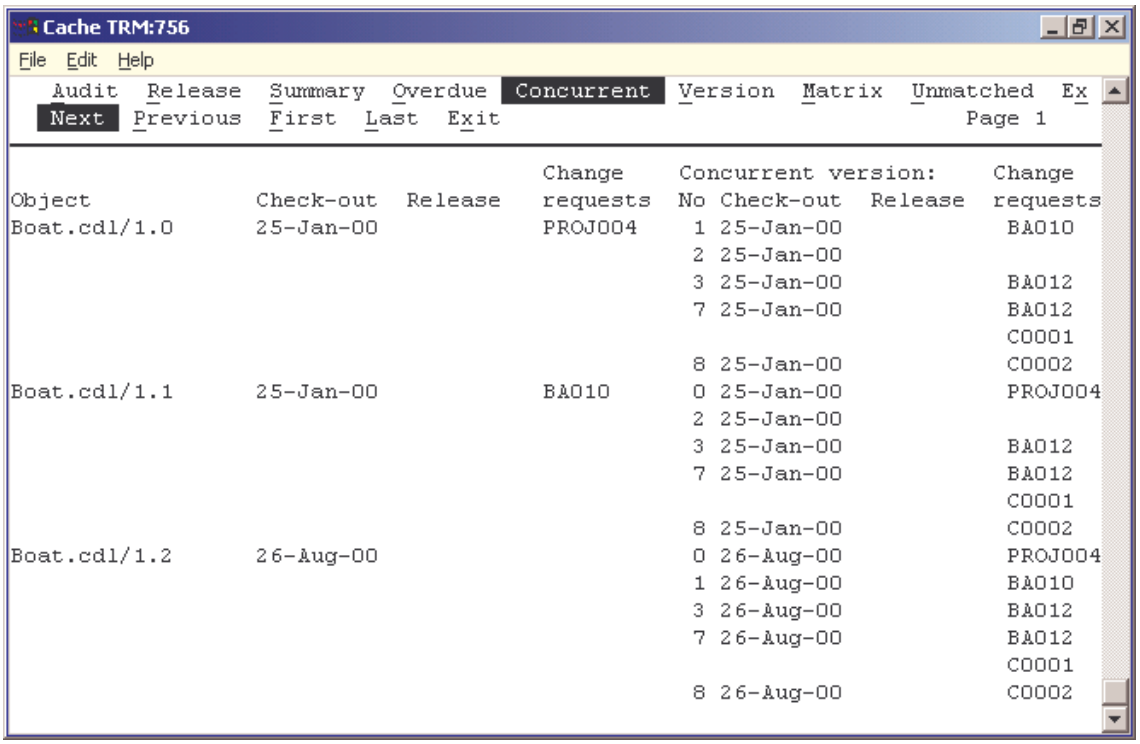


Figure 2.48 Concurrent development analysis

Diff: Compare two objects

This function can be used to compare any two object versions or change requests in any two locations.

The function creates two sets of sequential files containing copies of the components to be compared. The host operating system compare utility is then invoked to compare the two sets of files. The output produced is dependent upon the operating system compare utility and the parameters used for the comparison.

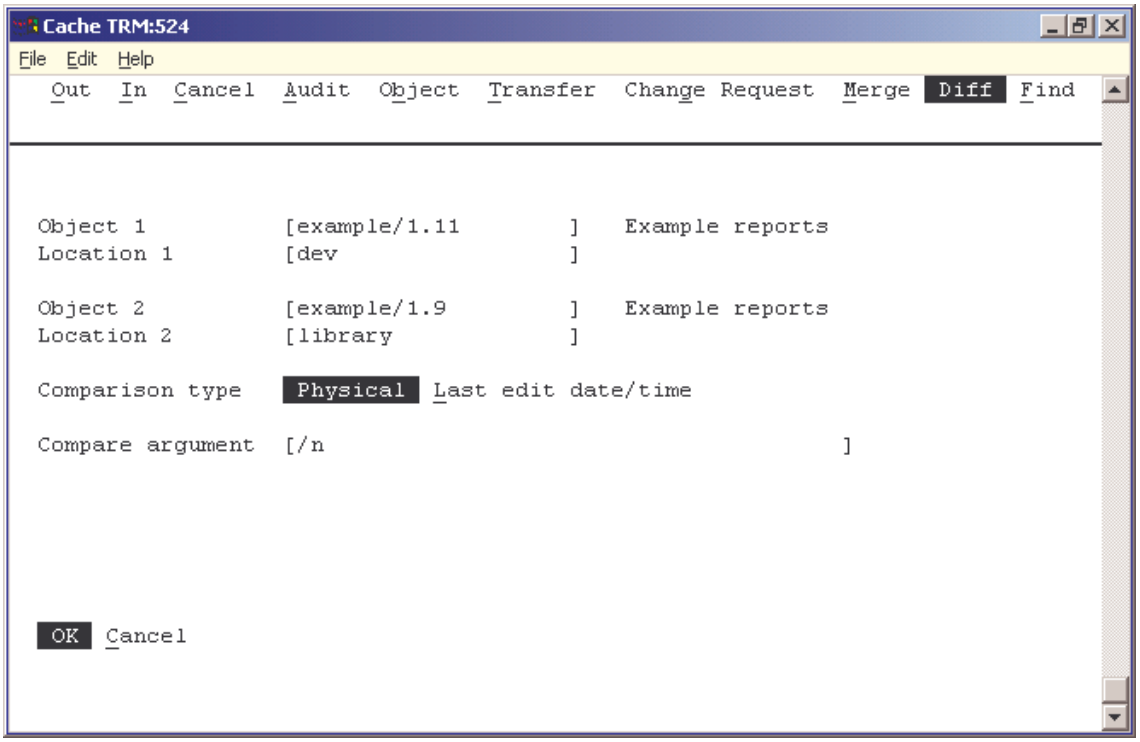


Figure 2.49 Compare two objects

VC/m uses values set up under 'Compare' on the second page of the Configure option in VC/m Set-up. This determines the scratch directory where the files will be created and the file name prefix.

The two sets of sequential files are named as follows:

```
xxxA.$   xxxA.G   xxxA.R
xxxB.$   xxxB.G   xxxB.R
```

where *xxx* is the file name prefix specified in the Configure option of VC/m Set-up

The set suffixed with A contain the components of the first object and the set suffixed with B contain the components of the second object. The file extension indicates the type of component, as follows:

$	Header file
G	Global sub-trees
R	Routines

If possible, the host operating system compare utility is invoked to compare the contents of each set of files. If the host operating system compare utility cannot be invoked directly (e.g. under DOS), VC/m will create a batch command file which can be run independently to perform the comparison.

The objects contained in a change request can be compared with their ancestors by entering the change request code (prefixed by @) at the first object prompt. This provides a one-step comparison report of all changes made for a single project, enhancement or bug fix.

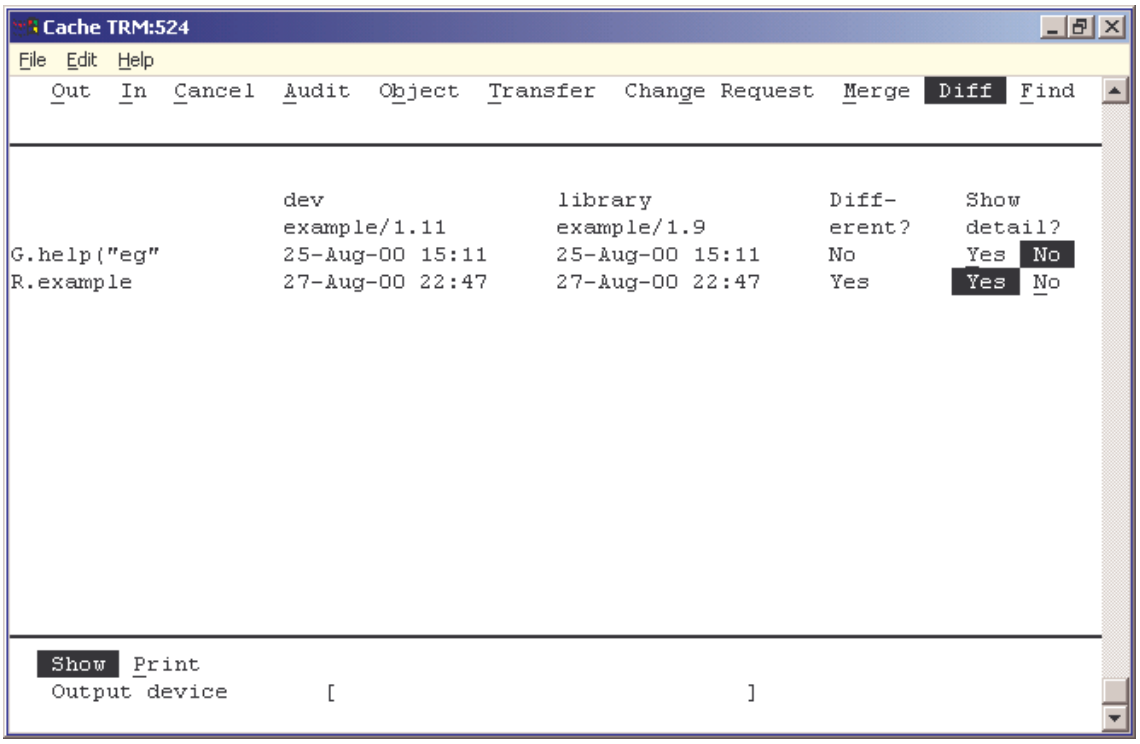


Figure 2.50 Compare two objects (results)

Find: Search for a string

The search function is primarily intended for use in searching VC/m library locations, which cannot be searched using standard M search facilities. However, it can be used to search any VC/m location type except for sequential files.

Figure 2.51 Search for a string

- Location to search

  Enter the code of the location that you wish to search. This can be any type of location except one which maps onto a sequential file.

- Object(s)

  Enter the object versions to be selected. They can be selected and de-selected in the standard ways.

  More detailed information can be found on p. 27.

  A selection list of the objects in VC/m is available.

- Search for

  Enter the string(s) that you are searching for.

- Run in Background?

  If you elect to Print the results of the search, you have the option to run the search as a background task. This can be useful if you are performing a large search.

Matrix: Matrix of version by location

This report shows the status of each object version across one or more locations in matrix form.

Each selected location is shown in a column of the matrix. Each selected object version is shown in a row of the matrix. If any selected object is not recorded in any of the selected locations, it will not appear on the report.

If an object version is recorded at a location, its status is shown at the intersection of the location and the object version. One of the following status codes will be displayed:

Status Code	Meaning
M	Master copy
A	Active at location
E	Error at location

If one or more systems are specified, only the objects which belong to the selected systems will be included in the matrix. For each selected system, a row will be included at the end of the matrix. The location in which the system is installed will be indicated with an X in the appropriate columns.

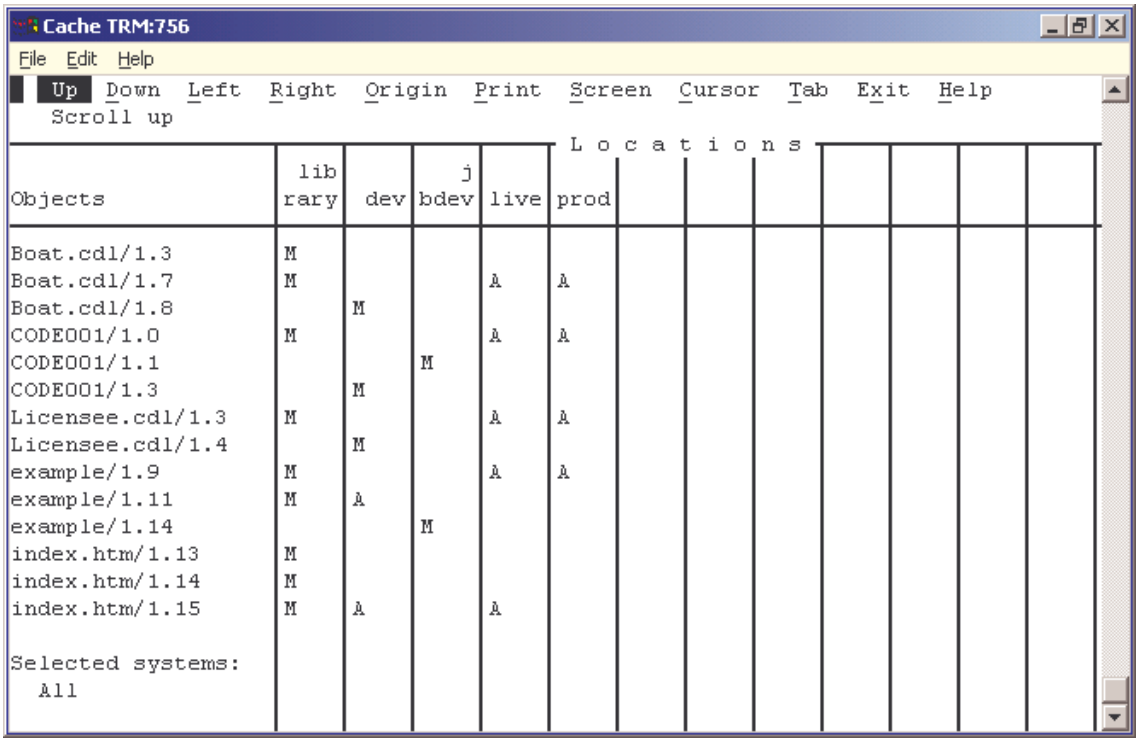


Figure 2.52 Matrix of version by location

- Systems

  Enter a list of systems for which the matrix is required. If no systems are entered, objects belonging to all systems will be selected.

  If one or more systems are entered, only those objects which belong to the selected set of systems will be included in the matrix.

- Object(s)

  Enter the objects for which the matrix is required. They can be selected de-selected in the standard ways.

  More detailed information can be found on p. 27.

  Only objects that are members of the selected set of systems can be selected.

  If no objects are selected, all qualifying objects will be included in the matrix.

- Location /class

  Enter a list of locations for which the matrix is required. Each selected location will be shown as a column in the matrix.

  If no locations are selected, all locations will be included in the matrix.

  A set of locations may be selected by referencing a location class, prefixed with @.

Overdue: Overdue checked-out objects report

This report lists all objects that have been checked-out for development but have not been returned to the library by their planned release date.

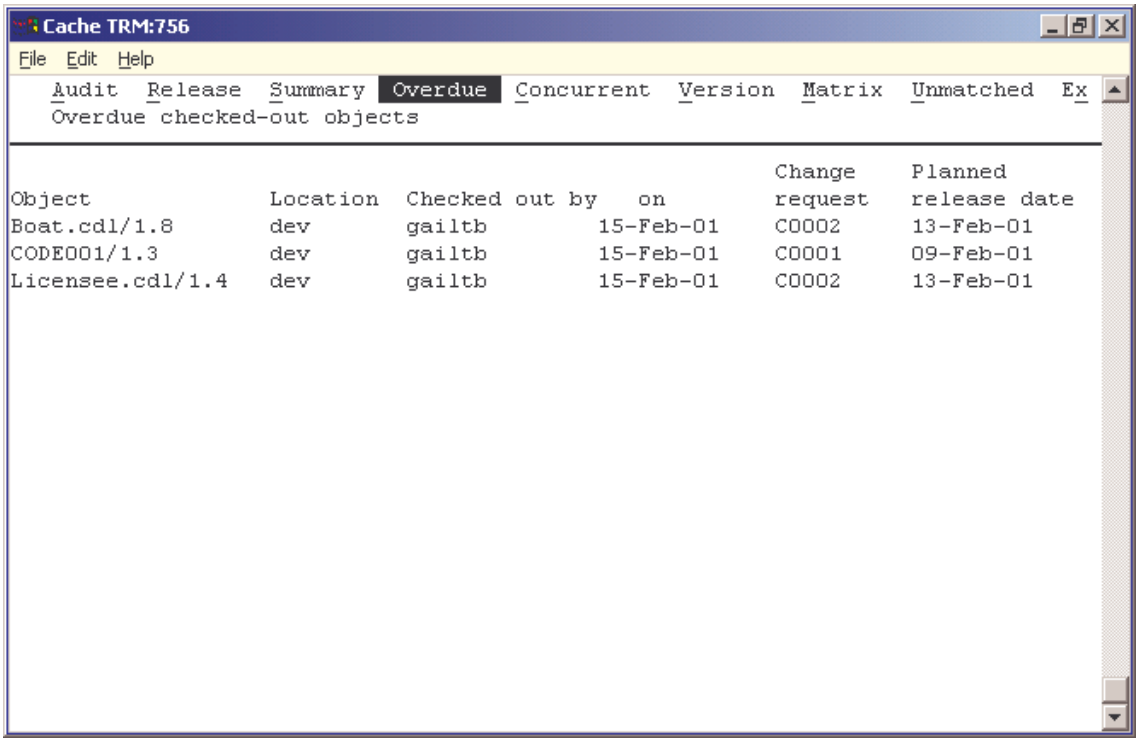


Figure 2.53 Overdue checked-out objects report

Release: Release control sheet

When a release is to be produced for installation on a LIVE system, a release control sheet can be produced. This enables the user to determine what will be released when the release function is invoked, without actually performing the release. It uses the same object selection mechanism as the release process ('from location' and release date). For each object version to be released, it lists the object version, the associated change requests which caused the change and the developers' and operators' notes.

A release control sheet is printed for each 'to location'. It can be used as a checklist by operators to ensure that they take any appropriate precautions before installing the software release (for example, shutting down processes, stopping users from running certain objects, etc.).

The report can also be used as a way of informing users which change requests will be addressed by a release.

After the release control sheet has been printed, the status function can be enabled or disabled. For example, it could be disabled when a release control sheet is first created, preventing further objects from being authorized. This effectively freezes the release area for a period to enable the release control sheet to be checked. The status function could be enabled as the release is actually performed.

- Status function

  The status function can be enabled or disabled after the release control sheet has been printed.

- From location

  Enter the code for the location from which objects should be transferred.

- Release date

  If a release date is entered, only those objects which have the same or an earlier release date will be selected.

  In addition, any objects at the selected location which do not have a specified release date will be selected.

- Object(s)

  The number of objects which match the selection criteria will be displayed, to give an indication of the size of the transfer. The selection cannot be altered.

- To location class

  Enter the location class to which the selected objects are to be transferred.

  If a location class is not selected, a specific location must be entered in the next field.

- To location

  If a location class is not entered, a single location must be entered here.

Summary: Summary of transfers by period

The report summarizes all object movements in a given period. It shows the number of objects transferred between any two locations. Each total will be broken down by type of transfer (new object or amendment) and change request type.

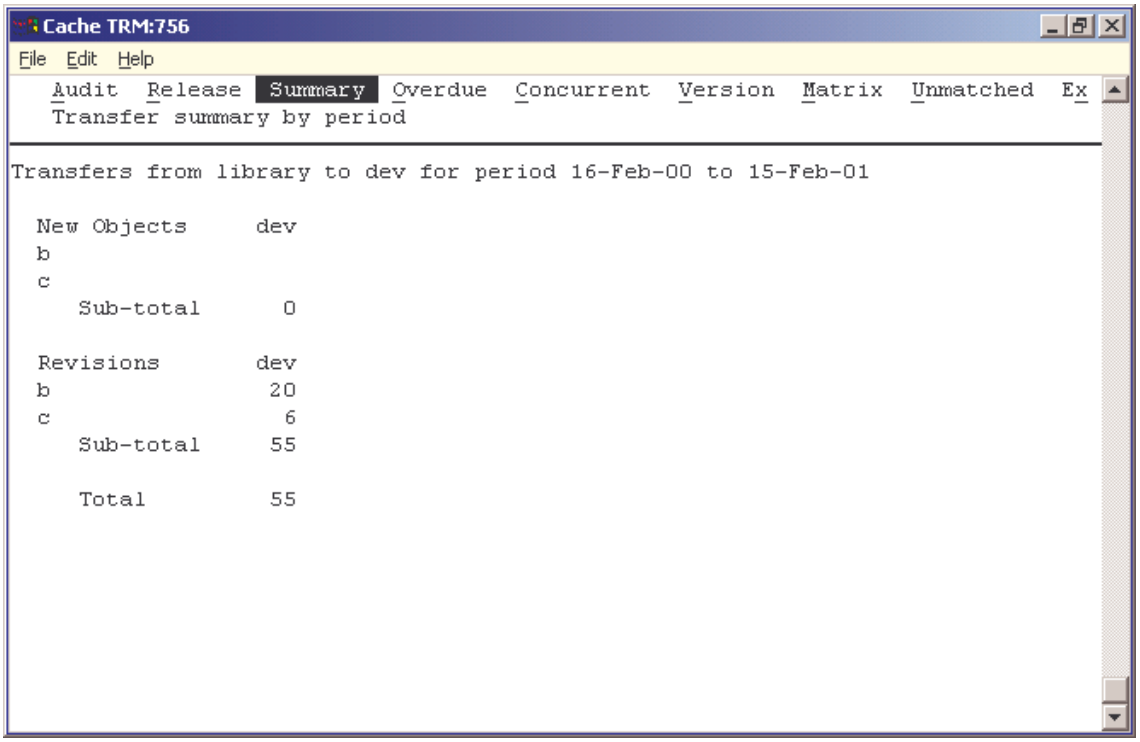


Figure 2.54 Summary of transfers by period

Unmatched: Unregistered components report

This report examines the actual physical contents of one or more locations and reports the existence of any components in that location which have not been registered to VC/m (i.e. do not belong to an object version). This can occur, for example, if a developer creates a new component but forgets to add it to the component list of any object version.

The report can also be used during installation to establish which components exist in each location and, for routines only, whether they are the same or different.

If this report is run for routines at several locations, and the same unmatched component is found in each location, the versions are compared and an asterisk printed against each component version which is different.

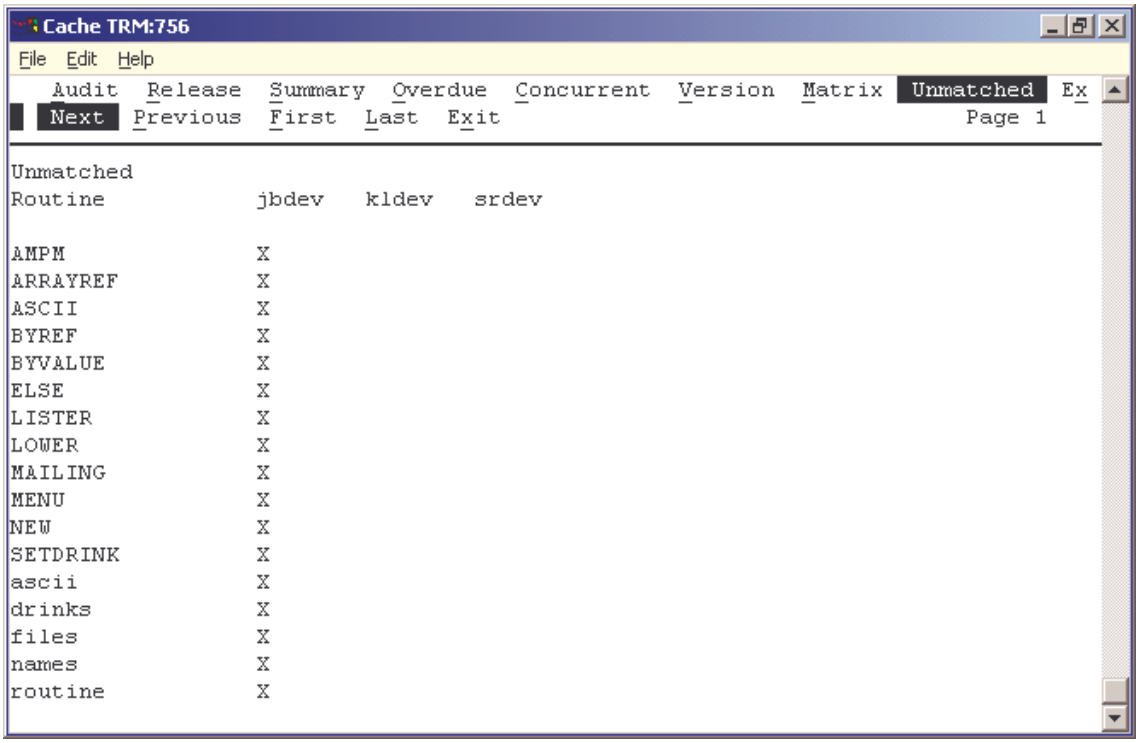


Figure 2.55 Unregistered components report

3 VC/m Callouts

A callout is an additional, installation-specific piece of code which is run when a certain operation is performed by the VC/m system. Callouts are written in the M programming language.

Callouts 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 (%).

3.1 Change Request Callouts

Change Request Validation

This callout is called when a change request is updated. It is called when an object is added to a change request (e.g. on a transfer screen) and when a change request code is entered or amended in the change request maintenance screen. It must be written as an extrinsic function.

Usage:

```
set err=$$label^routine
(function,obv,lcode1,lcode2,.chref,.chdesc,.chtype,.chdatex,oldchref)
```

Inputs:

function	=	Function code for the transfer (or CHMAINT)
obv	=	Object version being transferred (or null)
lcode1	=	From location code (or null)
lcode2	=	To location code (or null)
chref	=	Change request code
chdesc	=	Change request description
chtype	=	Change request type
chdatex	=	Change request date in a valid input format
oldchref	=	(deprecated) – may be null, or equal to chref, or contain the change request code overtyped in transfer screen of character-mode interface

Note: When this callout is called from the change request maintenance function, 'function' is CHMAINT and the values for 'obv', 'lcode1' and 'lcode2' are null.

If the change request code already exists in VC/m's database, the fields crdesc, crtype and crdatex will have a value. If the change request code does not exist, these fields will be null.

Outputs:

$$	=	1\error message = Change request is not valid 0 = Change request is valid
chref	=	Change request code
chdesc	=	Change request description
chtype	=	Change request type
chdatex	=	Change request date in DD-MMM-[CC]YY format

Change Request Status Validation

This callout is called when a change is made to the status of a change request. It must be written as an extrinsic function.

Usage:

```
set ok=$$label^routine(chtype,chref,chostat,chstat,.err)
```

Inputs:

chtype	=	Change request type
chref	=	Change request code
chostat	=	Old change request status
chstat	=	New change request status

Outputs:

$$	=	1 = Change request status change is valid 0 = Change request status change is not valid
err	=	Error message if status change is not valid

Change Request Commit

This callout is called when details of a change request are saved. It must be written as a line label or routine call.

Usage:

```
do label^routine("CHMAINT",null,null,null,chref)
```

Inputs:

chref	=	Change request code

Output:

None

3.2 Transfer Callouts

Transfer Initialization and Termination

The Transfer Initialization callout is called when VC/m has verified that a transfer request is valid and before it starts to transfer the objects.

The Transfer Termination callout is called after VC/m has transferred all of the objects in a transfer.

Both of these callouts are called once for a volume transfer, rather than for each object. Both must be written as a line label or routine call.

Usage:

```
do label^routine
```

Before and After Object Transfer

The Before Object Transfer callout is called immediately before VC/m transfers each object in a transfer request.

The After Object Transfer callout is called immediately after VC/m transfers each object in a transfer request.

Both of these callouts are called repeatedly during a volume transfer. Both must be written as a line label or routine call.

Usage:

```
do label^routine
```

Single-Object Transfer Confirm

This callout is called when the user confirms a single-object transfer request on the character-based interface. It must be written as a line label or routine call.

Note: This callout provides very limited functionality and should be used with caution.

Usage:

```
do label^routine(fcode,obv,lcode1,lcode2,.chrefs)
```

Inputs:

fcode	=	Function code for the transfer
obv	=	Object version being transferred
lcode1	=	From location code
lcode2	=	To location code
chrefs	=	Array of change requests for the object version

Output:

None

3.3 Component Driver Callouts

Component Save

These callouts are called when a component is written to a physical location with a storage format of M. The callout must be written as a line label or routine call.

R, INT, P and WLD Components

Usage:

```
do label^routine(lcode,addr,obv,cret,function,%usr,$job,id)
```

Inputs:

lcode	=	Location code
addr	=	Physical address
obv	=	Object version
cret	=	Component reference
fcode	=	Function code for the transfer
%usr	=	User ID
$job	=	M job number
id	=	Unique ID of temporary global

Output:

None

CDL Components

Usage:

```
do label^routine(context)
```

Input:

context	=	Array containing context information

Output:

None

INC and MAC Components

Usage:

```
do label^routine
```

Input:

None

Output:

None

Component Delete

These callouts are called when a component is deleted from a physical location with a storage format of M. The callout must be written as a line label or routine call.

R, INT, P and WLD Components

Usage:

```
do label^routine(lcode,addr,obv,cret,function,%usr)
```

Inputs:

lcode	=	Location code
addr	=	Physical address
obv	=	Object version
cret	=	Component reference
fcode	=	Function code for the transfer
%usr	=	User ID

Output:

None

INC and MAC Components

Usage:

```
do label^routine
```

Input:

None

Output:

None

Component Comment Lines

These callouts add comment lines to a component when it is written to a physical location with a storage format of M. The callout must be written as a line label or routine call.

R, INT, WLD, INC and MAC Components

Usage:

```
do label^routine(lcode,addr,obv,cret,id)
```

Inputs:

lcode	=	Location code
addr	=	Physical address
obv	=	Object version
cret	=	Component reference
id	=	Unique ID of temporary global

Output:

None

3.4 Change Request Selection Callouts

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

Default Values for Selection Filters

This callout is used to set default values for the selection filters. It must be written as a line label or routine call.

defaults^%vc029 is provided as an example.

Usage:

```
do label^routine(user,.context,.filter)
```

Inputs:

user	=	Current user ID
context("function")	=	Selected function code from transfer screen
context("objectVersion")	=	Selected object version from transfer screen
context("fromLocation")	=	Selected 'from location' from transfer screen
context("toLocation")	=	Selected 'to' location from transfer screen
context("systems")	=	Backslash delimited list of currently selected systems

Outputs:

filter("changeRequest")	=	Default change request filter value
filter("ownerUserId")	=	Default user ID filter value
filter("changeRequestType")	=	Default change request type filter value
filter("status")	=	Default change request status filter value

Filtering Rules

This callout is used to filter the values which are available in the change request selection field in the browser interface. It is called for each change request individually. It must be written as an extrinsic function.

filter^%vc029 is the standard functionality which will be replaced.

Usage:

```
set ok=$$label^routine(user,chref,.filter)
```

Inputs:

user	=	Current user ID
chref	=	Change request code
filter("changeRequest")	=	Change request filter value as entered on the selection field
filter("ownerUserId")	=	User ID filter value as entered on the selection screen
filter("changeRequestType")	=	Change request type filter value as entered on the selection screen
filter("status")	=	Change request status filter value as entered on the selection screen

Outputs:

$$	=	1 = Change request to be included in result list 0 = Change request not to be included in result list

4 VC/m API

4.1 Common VC/m Functions

connect - Establish a Connection with VC/m

This call establishes a log-in connection to VC/m. It performs two basic functions:

- Validation of the user ID
- Reservation of a license slot

While connected, a user license slot is in use by the calling program.

It is not mandatory for a connection to be made before making a call to any VC/m API function. However, if a call is made but no free license slots are available, the call will fail with an error message. It is therefore recommended that a connection is established before any other API calls are made.

If a VC/m API call is made from within VC/m (i.e. through a custom menu function), it is not necessary to establish a connection since logging into VC/m will have done this.

The userid argument which is required by all API calls is available in the %usr variable. This variable can be used to avoid the overhead of repeatedly deriving the user ID from the host operating system.

Usage:

```
set ok=$$connect^%vcapi(.userid,.license,.err)
```

Inputs:

userid	=	VC/m user ID (derived from host operating system if null)

Outputs:

$$connect	=	1 = Operation succeeded 0 = Operation failed
userid	=	User ID as supplied or derived from host operating system
license	=	License token (used to disconnect)
err	=	Error message if operation failed

Error messages:

vc0001	Invalid User	The user ID is not registered with VC/m.
vc0002	Number of Licensed users is nn. License limit exceeded	The number of users connected to VC/m at the time of the call has exceeded the license limit for this instance of VC/m.
vc0003	Invalid User	The user ID cannot be derived from the host operating system and must be supplied by the calling function.
vc0004	VC/m is not installed or not licensed	The license key file was not found, could not be opened or is invalid.
vc0005	The license for VC/m expired on dd-mmm-ccyy	The license key is time limited and has now expired.

discon - Disconnect from VC/m

This call releases a license slot which was allocated by the connect API call. This function should be called at the end of a VC/m session if connect was called at the start of the session.

Note: If the session is closed without calling this function, the license slot will normally be deallocated automatically.

Usage:

```
do discon^%vcapi(license)
```

Inputs:

license	=	License token returned by $$connect function

Output:

None

selnew - Create a New Work File

If the list of objects belonging to a change request needs to be stored in a global rather than a local array, a unique key to the ^%vcsel work file is required. This key can be any non-numeric value. It is the responsibility of the calling program to ensure that the key is not already in use by another process. This function can be used to allocate a guaranteed unique key which can be used once and discarded.

It is the responsibility of the calling program to delete the work file contents after use.

Usage:

```
set ok=$$selnew^%vcapi(userid,.key,.count,type,.err)
```

Inputs:

userid	=	VC/m user ID (derived from host operating system if null)
type	=	Arbitrary string identifying type of contents of work file (by convention "O" is used for objects)

Outputs:

$$vcsel	=	1 = Operation succeeded 0 = Operation failed
key	=	Guaranteed unique key to ^%vcsel work file global
count	=	Number of items in work file (always 0 for a new work file)
err	=	Error message if operation failed

4.2 Change Requests

crput - Add /Modify a Change Request

This API call adds a new change request to the VC/m database or modifies details of an existing change request.

All the fields are validated. The update is not performed if any validation errors are detected.

All auditable changes (e.g. addition /removal of objects from a change request) are logged in the VC/m audit trail.

If user-defined callouts are set up, these will be called as appropriate. The following calls are invoked during validation:

- Change request validation
- Change request status validation

And the following call is invoked after the update is complete:

- Change request commit

Usage:

```
set
ok=$$crput^%vcapi(userid,.chref,chtype,chname,chouser,chodate,chstatus,.ch
desc,.obvs,.chcomm,.err,[msg])
```

Inputs:

userid	=	VC/m user ID (derived from host operating system if null)
chref	=	Change request code (or null for auto number allocation)
chtype	=	Change request type. Must exist in change request type file.
chname	=	Change request name
chouser	=	Owner User ID (defaults to User ID)
chodate	=	Planned completion date (ccyymmdd format)
chstatus	=	Change request status (must be valid for change request type)
chdesc(1..n)	=	Change request description lines
obvs	=	If obvs not null, key to work file ^%vcsel(obvs,obv) containing list of objects attached to change request
obvs(obv)	=	If obvs is null, array of objects (object/variant.version) attached to change request
chcomm(1..n)	=	Change request comment lines.
msg	=	0 = Do not display audit trail messages on screen (default) 1 = Display audit trail messages

Outputs:

$$cradd	=	1 = Operation succeeded 0 = Operation failed
err	=	Error message if operation failed
chref	=	Allocated change request code if auto-allocation

Error messages:

vc1001	Invalid change request type	Null change request type is not permitted
vc1002	Invalid change request type	Change request type is not on file
vc1003	Invalid change request type	Change request type is flagged as deleted
vc1004	Invalid Owner user id	The owner user ID is not a valid VC/m user
vc1005	Invalid Owner user id	The user ID is flagged as deleted
vc1006	Invalid change request status	The change request status does not exist for the specified change request type
vc1007	Invalid change request number	A change request code must be from 1 to 10 alphanumeric characters
vc1008	Change request name is mandatory	
vc1009	Invalid planned completion date	
vc1010	Invalid Object xxx/xxx.nnn	Object name error
vc1011	Invalid Object xxx/xxx.nnn	Variant name error
vc1012	Invalid Object xxx/xxx.nnn	Version name error
vc1013	Object xxx/xxx.nnn does not exist	
vc1014	Cannot auto-allocate change request of this type	Change request type does not allow auto-allocation of change request code. The change request code was specified as null or the change request type table may be locked.
vc1015	Change request is locked	Another user or process is currently updating the same change request.

crget - Get an Existing Change Request

This API call is used to retrieve data about a change request, either for inquiry purposes or as a precursor to updating it.

Usage:

```
set
ok=$$crget^%vcapi(userid,chref,.chtype,.chname,.chouser,.chodate,.chcuser,
.chcdate,.chauser,.chadate,.chstatus,.chdesc,.obvs,.chcomm,.err)
```

Inputs:

userid	=	VC/m user ID (derived from host operating system if null)
chref	=	Change request code
obvs	=	Flag indicating whether to return list of objects in a work file or a local array. Null = Return list in local array Not null = Return list in global array using obvs as key

Outputs:

$$crget	=	1 = Operation succeeded 0 = Operation failed
chtype	=	Change request type. Must exist in change request type file
chname	=	Change request name
chouser	=	Owner user ID
chodate	=	Planned release date (ccyymmdd format)
chcdate	=	Creation user ID
chcuser	=	Creation date and time (ccyymmddhhmmss format)
chauser	=	Last amendment user ID
chadate	=	Last amendment date and time (ccyymmddhhmmss format)
chstatus	=	Change request status
chdesc(1..n)	=	Change request description lines
obvs	=	If obvs not null, key to work file ^%vcsel(obvs,obv) containing list of objects attached to change request
obvs(obv)	=	If obvs is null, array of objects (object/variant.version) attached to change request
chcomm(1..n)	=	Change request comment lines
err	=	Error message if operation failed

Error messages:

vc1016	Invalid change request number	A null change request code is not permitted
vc1017	Change request does not exist	The change request code supplied does not exist

crobvput - Add an Object to a Change Request

This API call adds a single object to an existing change request. An entry is written to the audit trail for both the object and the change request.

Usage:

```
set ok=$$crobvput^%vcapi(userid,chref,obv,.err)
```

Inputs:

userid	=	VC/m user ID (derived from host operating system if null)
chref	=	Change request code (must already exist)
obv	=	Object/variant.version of object to be added. If object already belongs to change request, no action will be taken.

Outputs:

$$craddobv	=	1 = Operation succeeded 0 = Operation failed
err	=	Error message if operation failed

Error messages:

vc1010	Invalid Object xxx/xxx.nnn	Object name error
vc1011	Invalid Object xxx/xxx.nnn	Variant name error
vc1012	Invalid Object xxx/xxx.nnn	Version name error
vc1013	Object xxx/xxx.nnn does not exist	
vc1016	Invalid change request number	A null change request code is not permitted
vc1017	Change request does not exist	The change request code supplied does not exist

crobvdel - Delete an Object from a Change Request

This API call deletes an object from a change request. An audit trail entry is written for both the object and the change request.

Usage:

```
set ok=$$crobvdel^%vcapi(userid,chref,obv,.err)
```

Inputs:

userid	=	VC/m user ID (derived from host operating system if null)
chref	=	Change request code (must already exist)
obv	=	Object/variant.version of object to be deleted. If object does not belong to change request, no action will be taken.

Outputs:

$$crobvdel	=	1 = Operation succeeded 0 = Operation failed
err	=	Error message if operation failed

Error messages:

vc1010	Invalid Object xxx/xxx.nnn	Object name error
vc1011	Invalid Object xxx/xxx.nnn	Variant name error
vc1012	Invalid Object xxx/xxx.nnn	Version name error
vc1013	Object xxx/xxx.nnn does not exist	
vc1016	Invalid change request number	A null change request code is not permitted
vc1017	Change request does not exist	The change request code supplied does not exist

crdel - Delete a Change Request

This API call deletes a change request.

Usage:

```
set ok=$$crdel^%vcapi(userid,chref,.err,[msg])
```

Inputs:

userid	=	VC/m user ID (derived from host operating system if null)
chref	=	Change request code
msg	=	0 = Do not display audit trail messages on screen (default) 1 = Display audit trail messages

Outputs:

$$crdel	=	1 = Operation succeeded 0 = Operation failed
err	=	Error message if operation failed

Error messages:

vc1015	Change request is locked	Another user or process is currently updating the same change request.
vc1016	Invalid change request number	A null change request code is not permitted
vc1017	Change request does not exist	The change request code supplied does not exist

Note: If audit trail messages are to be displayed on the screen, the msg argument must be set to 1. If there is not one already open, this will open a message window at the bottom of the screen. After all API calls have been completed, the message window should be closed by making a call to mterm^%vc1aud as follows:

```
s %msg=1

d mterm^%vc1aud
```

crexist - Does a Change Request Exist?

This API call can be used to test whether a change request exists.

Usage:

```
set ok=$$crexist^%vcapi(userid,chref,.exists,.err)
```

Inputs:

userid	=	VC/m user ID (derived from host operating system if null)
chref	=	Change request code

Outputs:

$$crexist	=	1 = Operation succeeded 0 = Operation failed
exists	=	1 = Change request exists 0 = Change request does not exist
err	=	Error message if operation failed

Error messages:

vc1016	Invalid change request number	A null change request code is not permitted

crstval - Validate a Change Request Status Change

This function enables a status change within a change request to be validated.

Note: lcode1,addr1,lcode2 and addr2 are returned solely for the purpose of passing to the change request status change function: crstmod.

If the crstmod function is used to perform the status change, the return values lcode1, addr1, lcode2 and addr2 must be passed into that function. If the crput function is used to update the change request with a new change request status, lcode1, addr1, lcode2 and addr2 can be discarded.

Usage:

```
set
ok=$$crstval^%vcapi(userid,chref,function,chstatus,.lcode1,.addr1,.lcode2,
.addr2,.err)
```

Inputs:

userid	=	VC/m user ID (derived from host operating system if null)
chref	=	Change request code
function	=	Transfer function code, e.g. XFER, IN, OUT
chstatus	=	New status

Outputs:

$$crstval	=	1 = status change valid 0 = status change not valid
lcode1	=	From location
addr1(psa)	=	Array of from physical locations
lcode2	=	To location
addr2(psa)	=	Array of to physical locations
err	=	Error message if status change not valid

Error messages:

vc1018	Invalid function code	Function code must exist
vc1019	Invalid change request status code	Status code cannot be null
vc1020	Invalid change request status code	Status code does not exist for this change request type

crstmod - Invoke a Change Request Status Change

This API call invokes a change request status change. If objects are linked to a change request, this function may trigger object transfers, depending on how VC/m has been configured.

Note: lcode1,addr1,lcode2 and addr2 should be obtained by first making a call to crstval.

Usage:

```
set
ok=$$crstmod^%vcapi(userid,chref,function,chstatus,lcode1,.addr1,lcode2,.a
ddr2,.err,[msg])
```

Inputs:

userid	=	VC/m user ID (derived from host operating system if null)
chref	=	Change request code
function	=	Transfer function code, e.g. XFER, IN or OUT
chstatus	=	New status
lcode1	=	From location
addr1(psa)	=	Array of from physical locations
lcode2	=	To location
addr2(psa)	=	Array of to physical locations
msg	=	0 = Do not display audit trail messages on screen (default) 1 = Display audit trail messages

Outputs:

$$crstmod	=	1 = Operation succeeded 0 = Operation failed
err	=	Error message if operation failed

Error messages:

vc1018	Invalid function code	Function code must exist
vc1019	Invalid change request status code	Status code cannot be null
vc1020	Invalid change request status code	Status code does not exist for this change request type

5 General VC/m Reference

5.1 Callable Standard VC/m Menu Options

Archive: Archive and purge old versions	^%vc0as
Audit: Audit trail	^%vc340
Baseline: Create a new baseline	^%vc440
Bulk: Transfer multiple objects between locations	a20^%vc0re
Cancel: Cancel an object checked-out in error	^%vc0ca
Change Request: Change request maintenance	^%vc330("")
Class: Location class maintenance	^%vc290
Concurrent: Concurrent development analysis	^%vc370
Delete: Remove an object completely	^%vc520
Device Types: Device type maintenance	^%vcset30
Diff: Compare two objects	^%vc610
Find: Search for a string	^%vc540
In: Check in an object	^%vc220("IN")
Install: Install objects from a sequential file	^%vc460
Install: Installation settings	^%vcset60
Integrity: Database integrity checker	^%vc430
License: License key maintenance	^%vc025
Load: Bulk registration of components to objects	^%vc410
Location: Location maintenance	^%vc280
Location: Location maintenance menu	^%vcmenu("vcm1a")
Matrix: Matrix of version by location	^%vc560
Menu: Menu maintenance	^%vc570
Merge: Merge two versions of an object	^%vc250
Module: Module maintenance	^%vc310
Object: Object registration	^%vcmenu("vc0mo")
Add /Modify: Add an object or modify an existing one	^%vc0mo
Components: Add or modify components of an object	^%vc0mo2
Delete: Delete an object	^%vc0mo1

Move: Move a component from one object to another	^%vc0mo5
Print: Print details of an object or objects	print^%vc0mo3
Show: Display details of an object	show^%vc0mo3
Out: Check out an object	^%vc220("OUT")
Overdue: Overdue checked-out objects report	^%vc360
Manager: VC/m System and Project Manager options	^%vcmenu("vcm")
Physical Location: Physical location maintenance	^%vc270
Printers: Output device maintenance	^%vcset20
Programmer: VC/m Developer options	^%vcmenu("vcp")
Properties: VC/m Properties	^%vcset50
Release: Release control sheet	arpt^%vc0re
Release: Release objects to one or more locations	auto^%vc0re
Rollback: Roll back objects from a backup file	^%vc500
Routes: Transfer route maintenance	^%vc300
Setup: VC/m Setup options	^%vcmenu("vcsetup")
Summary: Summary of transfers by period	^%vc350
System: System maintenance	^%vc260
Terminals: Terminal maintenance	^%vcset10
Transfer: Transfer an object between locations	^%vc220("XFER")
Transfer Files: Naming of sequential files	^%vcset40
Types: Change request type maintenance	^%vc420
Unmatched: Unregistered components report	^%vc400
User: User maintenance	^%vc320
Version: Versions across locations report	^%vc390

5.2 VC/m Globals

^%vcal = VC/m - Version at Location
^%vcal2 = VC/m - Version at Location Indexes
^%vcat = VC/m - Audit Trail
^%vcat2 = VC/m - Audit Trail Indexes
^%vcau = VC/m - Master File Audit Trail
^%vccd = VC/m - Caché CDL Component Driver Log File
^%vcch = VC/m - Change Requests
^%vcch2 = VC/m - Change Request Indexes
^%vcco = VC/m - Systems
^%vcco2 = VC/m - System Indexes
^%vcct = VC/m - Installed Component Drivers
^%vcdev = VC/m - Device Types
^%vcfile = VC/m - Open File Handle Index
^%vcfiles = VC/m - Transfer File Lock Table
^%vchelp = VC/m - Help File
^%vcid = VC/m - Unique Token Allocation File
^%vcio = VC/m - Printer Settings
^%vclo = VC/m - Location Master File
^%vclo2 = VC/m - Location Indexes
^%vclp = VC/m - Physical Location Master File
^%vclt = VC/m - Installed Location Drivers
^%vcmf = VC/m - Master Files
^%vcmf2 = VC/m - Master File Indexes
^%vcmnu = VC/m - Menu File
^%vcna = VC/m - Name Allocation
^%vcob = VC/m - Object Definitions
^%vcob2 = VC/m - Object Indexes
^%vcoc = VC/m - Object Comments
^%vcopt = VC/m - Print Spooler Options
^%vcprint = VC/m - Print Spool File
^%vcsccs = VC/m - SCCS Location Driver Index
^%vcsel = VC/m - Selection Lists
^%vcst = VC/m - Transfer Statistics
^%vctmp = VC/m - Temporary File
^%vctouch = VC/m - Transfer work list - Temporary
^%vctr = VC/m - Transfer Routing
^%vcvc = VC/m - Configuration Parameters
^%vcxct = VC/m - Transfer Component Termination
^%vcyf = VC/m - Transfer Files - Temporary
^%vcyfd = VC/m - Transfer File Directory - Temporary
^%vcyo = VC/m - Object Workfile - Temporary
^%vcyr = VC/m - Report Workfile - Temporary
^%vcys = VC/m - General Workfile - Temporary
^%vcyt = VC/m - Routines Transfer Workfile – Temporary
^%vcyz = No longer used
^%vczn = No longer used

^%gjtask = VC/m - Task Server

^vcli = Objects stored in library location
^vcvp = Installation parameters for remote installation

5.3 Routine Naming Conventions

Installation-specific routines which are maintained by George James Software are named using '%vcy' followed by a number, e.g. %vcy1116.

Installation-specific routines which are maintained by the customer should be named using '%vcy' followed by at least one alphabetic character, e.g. %vcyman1.

5.4 Valid Names

The following table shows the criteria for valid names of the different items in the VC/m system:

Item	Valid Characters	Maximum Length	Case Sensitive?
Object base	Any letters, numbers and punctuation characters, except / and *	60 in combination with variant code and version number	Yes
Variant code	Any letters, numbers and punctuation characters, except . and / and *	see above	Yes
Version number	Numeric only	see above	Yes
Component name	Any letters, numbers and punctuation characters	200	Yes
Change request code	Any letters, numbers and punctuation characters	30	Yes
Change request type	Any letters, numbers and punctuation characters	16	Yes
Change request status	Any letters, numbers and punctuation characters	15	Yes
Location code	Any letters, numbers and punctuation characters	40	Yes
Physical location code	Any letters, numbers and punctuation characters	45	Yes
Location class	Any letters, numbers and punctuation characters	30	Yes
System	Any letters, numbers and punctuation characters	30	Yes
Function code	Any letters, numbers and punctuation characters	30	Yes
Module name	Any letters, numbers and punctuation characters	8	Yes
User ID	Any letters, numbers and punctuation characters	30	Partially
Access code	Uppercase letters and numbers	200 per user in combination with other access codes	No

5.5 Common File Extensions

World Wide Web

Extension	Component Type	Description
.asp	T	Active server page
.csp	T	Caché server page
.css	T	Cascading stylesheet
.hta	T	Hypertext application
.htc	T	HTML component
.htm	T	HTML document
.html	T	HTML document
.htt	T	Hypertext template
.gif	BIN	GIF image
.jpg	BIN	JPEG image
.jpeg	BIN	JPEG image
.js	T	JavaScript file
.jsp	T	Java server page
.shtml	T	SHTML document
.xml	T	XML document
.xsl	T	XSL stylesheet

Cross operating system

Extension	Component Type	Description
.zip	BIN	Zip file
.pdf	BIN	Portable document format document

Microsoft Office

Extension	Component Type	Description
.doc	BIN	Microsoft Word document
.dot	BIN	Microsoft Word template
.mdb	BIN	Microsoft Access database
.pps	BIN	Microsoft PowerPoint slide show
.ppt	BIN	Microsoft PowerPoint presentation
.xls	BIN	Microsoft Excel workbook
.xlt	BIN	Microsoft Excel template

Windows

Extension	Component Type	Description
.bat	T	Batch file
.bmp	BIN	Bitmap image
.com	BIN	Program file
.dll	BIN	Application extension
.exe	BIN	Program file
.log	T	Log file
.txt	T	Plain text file
.wav	BIN	Sound file

Index

G

H

I

L

M

N

O

P

R

S

T

U

V

X