GEORGEJAMES SOFTWARE

With thanks to:

NHS
Information Authority

Using Serenji

Serenji 2.1

Covering:

Starting Serenji
Using Serenji
Editing Routines
Printing

Contents

Revision	File Reference	Date	Author	Notes
0	\\Wiz\D\Lib\Manuals\Serenji\UsingSerenji.0.doc	6 December 2001	GJS Support Desk	

Introduction

Serenji is a debugging and editing tool for Caché, DSM, GT.M and MSM.

Serenji is essentially split into two main parts, the server-side components and the 32-bit Windows application.

Once installed, the Windows application runs a component called the Serenji Sentry which listens for a connection request from the server component. If a connection request is received, then a Serenji session is established and the connection to the server is made.

From that point the user can use the facilities of the Windows-based screen editor and the advanced debugging tools of Serenji. These tools include the ability to step through the code, examining information and setting breakpoints and break conditions. There are full routine-editing capabilities and rich source color-coding makes it easier to read and identify components.

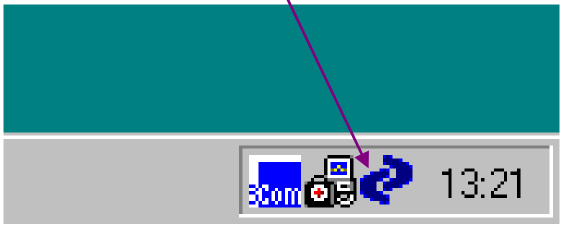
This document is intended to show you how to use the facilities offered by Serenji to aid you in both the development and support of M and /or Caché applications.

Starting Serenji

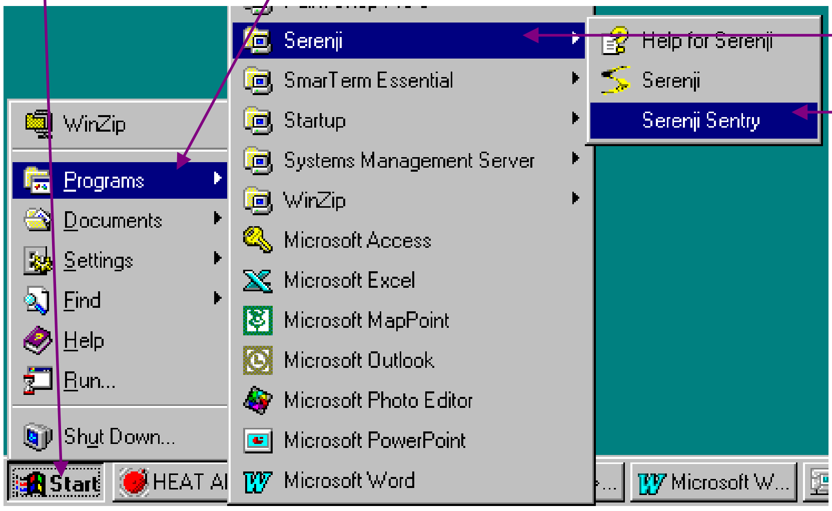
Once the application is installed on your PC and you have TCP/IP access to an M or Caché system with the server-side components installed, you can connect and use Serenji.

The Serenji Sentry

The installation of Serenji will by default add a shortcut to SerenjiSentry.exe into the Windows start-up folder on your PC. This is the component of Serenji that listens for a connection request from the M or Caché server. It means that each time you start up your PC this component will run and wait for a connection. If the component is running you will see it in the System Tray, **at the bottom right of your desktop screen**.



If it is not running or you have stopped it for some reason, you can start it manually by going to the **Start** button, choose **Programs**



Then **Serenji**

and **Serenji Sentry**.

Connecting to the M or Caché Server

Once the Sentry is running, you can connect to the server running the server-side components.

You should first connect to the M or Caché environment in the normal way.

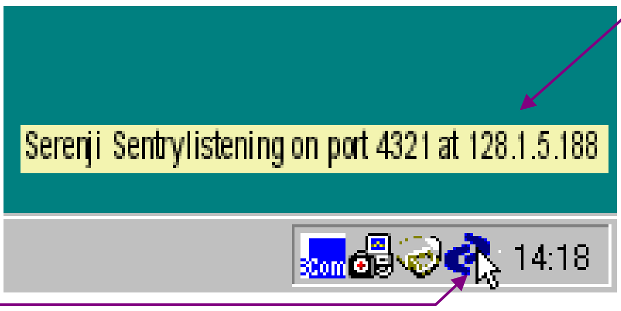
***NOTE**

Once you initiate a Serenji session, all M or Caché commands you type at the command line prompt are directed through Serenji and so there is a significant decrease in system performance. (Although as we shall see, there are ways to use the tool which will minimize speed decreases.) This is not a problem when stepping through code line by line or when writing a new routine but it is something to be aware of if a large amount of processing is expected to occur between breakpoints.

Finding your IP Address

When a Serenji session is started, you must specify the IP address of your PC, though on all server platforms except DSM you can use your PC's hostname as an alternative.

Finding the IP address of your PC is easily done when the Serenji Sentry is running. Simply **move the mouse pointer over the Sentry icon** in the System tray and Sentry will tell you what the **IP address** is.



Serenji Sentrylistening on port 4321 at 128.1.5.188

Bear in mind that some PCs are multi-homed (have more than on IP address), or are located behind firewalls, so the value reported here might not be the one which allows the server to reach it.

In many cases Serenji can successfully identify the address of the PC from which your terminal session connected. If no IP address or hostname is passed, it will default to use this. Try the following and see if this works for you:

D SHELL^%Serenji()

Starting a Session

Within the terminal session on the client machine, type the following at the command prompt :

D SHELL^%Serenji("*IP Address*")

Where ***IP Address*** is the number shown by Serenji Sentry as in the example on the previous page. The IP Address will be a set of four numbers.

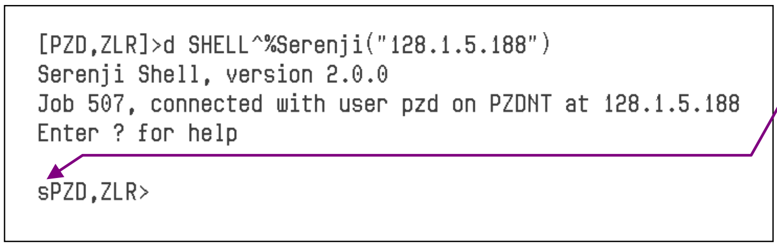
On some networks your IP Address may change from time to time so each time you start a Serenji Session you should check it using the Sentry, as illustrated previously.

***NOTE**

When typing in the command to start a Serenji session the routine name **is case sensitive**. It must have a capital **S** and the rest of the routine name is in lower case. Also the IP Address must be enclosed within quotes and there should be no spaces in the IP Address.

If you have typed the command correctly, have entered the correct IP Address and Serenji Sentry is running on your PC, a connected session will begin on your PC.

The Serenji screen will appear and then control returns to the M or Caché terminal session. Your screen will show something like this example.



```
[PZD,ZLR]>d SHELL^%Serenji("128.1.5.188")
Serenji Shell, version 2.0.0
Job 507, connected with user pzd on PZDNT at 128.1.5.188
Enter ? for help

sPZD,ZLR>
```

Notice the prompt is prefixed by a **s** to denote this is a connected Serenji session running within the Serenji Shell.

Using the Shell

From this point on, unless you specify otherwise, all commands typed will run through the Serenji debugger and will run much slower than normal.

For brief help on using the shell, type **?** and press enter.

If you want to edit a routine, type:

ED MyRtn

where **MyRtn** is the name of the routine you want to edit.

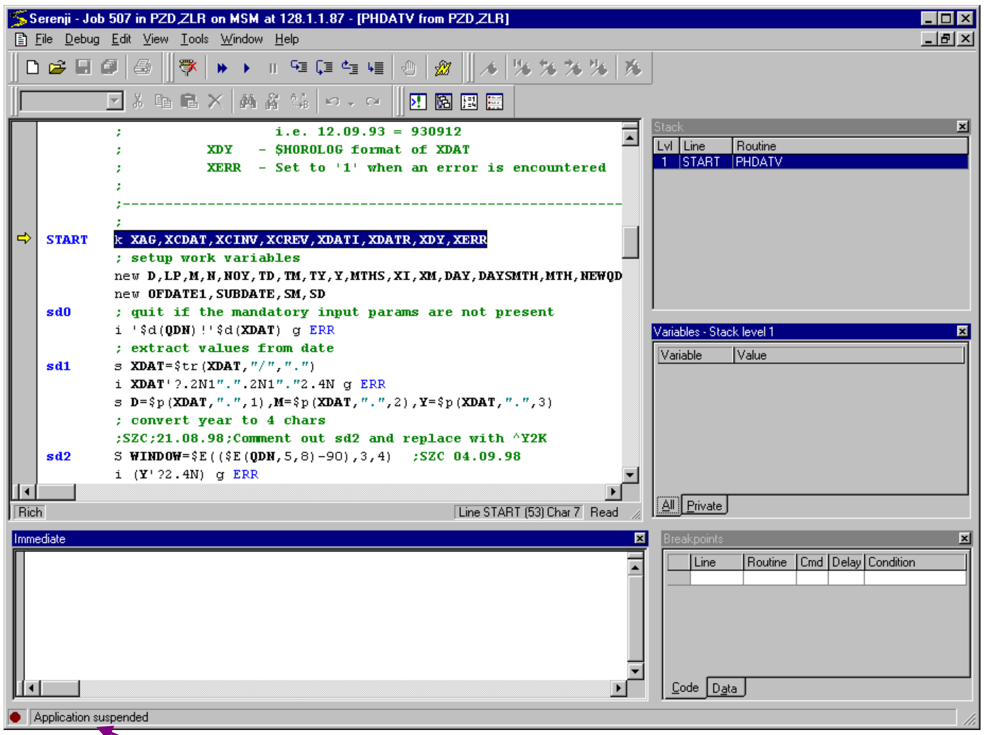
If you want to run a routine, with the debugger breaking at the first command, type:

DB ^MyRtn

where **MyRtn** is the name of the routine you want to debug.

DB stands for DeBug or Do and Break.

Serenji will begin to run the routine specified and will break at the first command of the routine. At that point the routine is loaded and displayed in the Serenji window, as in the example below.


Serenji - Job 507 in PZD,ZLR on MSM at 128.1.1.87 - [PHDATV from PZD,ZLR]
File Debug Edit View Tools Window Help
Stack
Lvl Line Routine
1 START PHDATV
Variables - Stack level 1
Variable Value
All Private
Rich
Line START (53) Char 7
Read
Immediate
Breakpoints
Line Routine Cmd Delay Condition
Code Data
Application suspended

```
             ;              i.e. 12.09.93 = 930912
             ;         XDY   - $HOROLOG format of XDAT
             ;         XERR  - Set to '1' when an error is encountered
             ;
             ;------------------------------------------------------------
             ;
START        k XAG,XCDAT,XCINV,XCREV,XDATI,XDATR,XDY,XERR
             ; setup work variables
             new D,LP,M,N,NOY,TD,TM,TY,Y,MTHS,XI,XM,DAY,DAYSMTH,MTH,NEWQD
             new OFDATE1,SUBDATE,SM,SD
sd0          ; quit if the mandatory input params are not present
             i '$d(QDN)!'$d(XDAT) g ERR
             ; extract values from date
sd1          s XDAT=$tr(XDAT,"/",".")
             i XDAT'?.2N1".".2N1"."2.4N g ERR
             s D=$p(XDAT,".",1),M=$p(XDAT,".",2),Y=$p(XDAT,".",3)
             ; convert year to 4 chars
             ;SZC;21.08.98;Comment out sd2 and replace with ^Y2K
sd2          S WINDOW=$E(($E(QDN,5,8)-90),3,4)   ;SZC 04.09.98
             i (Y'?2.4N) g ERR
```

Notice the **main status line** of the Serenji window which gives useful information about what Serenji is doing.

Alternatively, you can start the debug process from a particular line label by typing (for example):

DB Tag^MyRtn

This will load the routine called MyRtn into the debugger, begin running the routine at line label Tag and pause execution at that point. Labels that take parameters can also be called:

DB DoIt^MyRtn($H)

The Serenji Shell supports command line recall.

To recall the **Previous** command line, press the **cursor up** key or use **Ctrl+P**.

To move to the next line in the command history buffer, use the **cursor down** key or press **Ctrl+N**.

By default the history buffer holds the last fifty entries. It operates in a circular fashion, so if you move down to the last entry and move down once more it will go back to the first entry in the list.

To delete the entire command line, press **Ctrl+X**.

To move along the command line, use the **cursor left** and **cursor right** keys.

To move to the start of the line, use the **Home** key.

To move to the end of the line, use the **End** key.

To toggle between insert and overtype mode, use the **Insert** key.

To undo your edits, use **Ctrl+Z**.

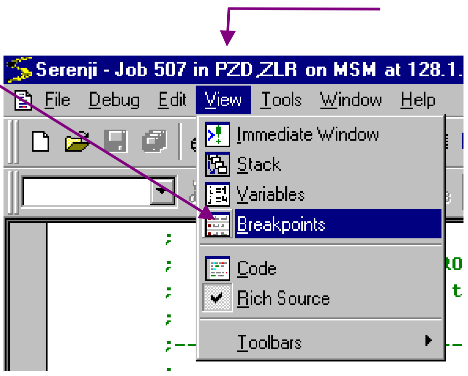
Quitting the Shell

To Quit from the Shell on the M or Caché system, type **Q** or **H**. This will return you to the normal M or Caché command prompt. You do not need to close the routines from within Windows first.

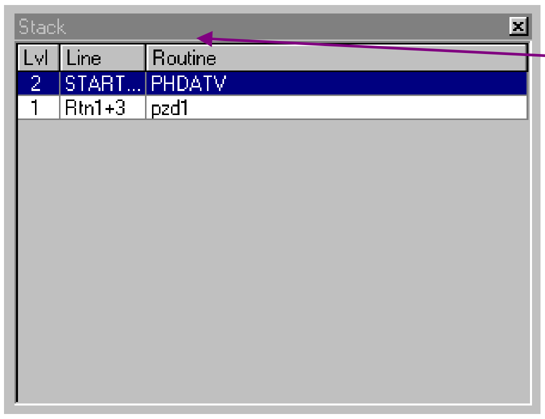
Using Serenji

The Serenji Screen

By default the Serenji screen displays all the possible information windows. To close a window, use its title bar 'Close' button or press Ctrl+F4. Windows can be displayed or switched to from the **View** menu, by clicking on the **appropriate window in the list**.



The Stack Window



The **Stack** window shows the calls made to subroutines and other routines.
In this example a call to PHDATV was made from ^pzd1, from line Rtn1+3.

Lvl	Line	Routine
2	START...	PHDATV
1	Rtn1+3	pzd1

If you click on a level in the window, the code display changes to that line of code in that routine.

You can change the order that the stack window displays the levels in by clicking on the **Lvl** column heading.

You can also adjust the width of the columns by dragging the divisions between them.

The routine name is omitted from a level if it is the same as the preceding level. A level entry without a Routine or Line entry means a call to untraceable code. This could be p-coded M routines or Caché routines with source code removed.

The Variables Window

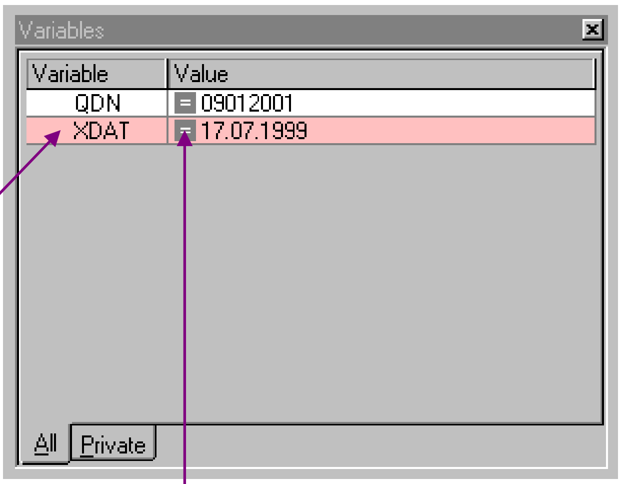
This window, as its name suggests, shows local variable names and their values.

The variables are listed in the window in alphabetical order.

Variables which were set or changed in the latest debugging step are highlighted in **Pink**.

Changes which affect the $DATA status of a variable are also highlighted in pink.

The values of the variables can be displayed in different formats. To change the format of the Value, click on the **Symbol** prefixing the value.

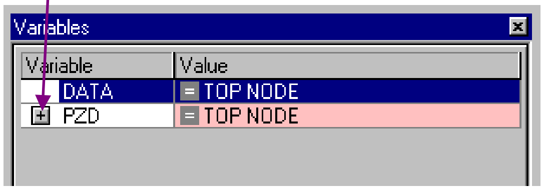


Variable	Value
QDN	09012001
XDAT	17.07.1999

Values can be displayed up to a maximum length of 2047 characters after which it is truncated. Truncation is indicated by a different symbol and ellipsis **(…)** at the end of the value.

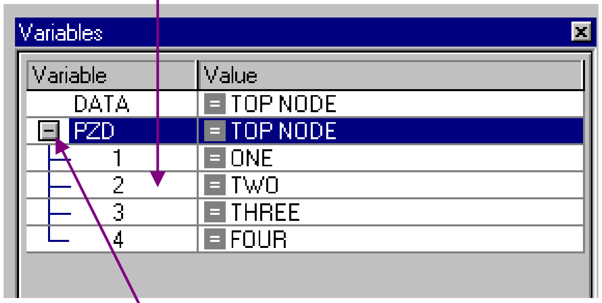
If you are interested in watching the value of a particular variable while stepping through the code of a routine, select that variable in the Variables window and it will remain in view while stepping through the code, provided the variable still exists.

Local variables which have descendants (an array) are displayed in the variables window with a **plus sign** next to them.



Variable	Value
DATA	TOP NODE
⊞ PZD	TOP NODE

Clicking on the plus sign will expand the display to show the **descendants** and their values.



Variable	Value
DATA	TOP NODE
⊟ PZD	TOP NODE
1	ONE
2	TWO
3	THREE
4	FOUR

Click on the **minus sign** to collapse the display again.

The **Private** tab can only be used when debugging Caché applications. It will show variables which are private to the current subroutine or function. They are private if their scope is limited by the use of a New statement or of formal parameters. Formal parameters are listed in bold type.

The column widths in the Variables window can be adjusted by dragging the divisions between them.

When stepping through code in the code window, operation will be faster if Serenji does not have to keep refreshing the Variables window after each step. So if you don't need it, close it. You can open it at any point if you need to see any of the variables. Moving the mouse pointer over a variable name in the Code window will display that variable name and its value. In many cases this is an adequate substitute for the Variables window.

The Breakpoints Window

This window allows you to see, set and edit breakpoints set in the code. Execution of the code is paused **just before** the specified command is performed.

A breakpoint is a point at which execution of the code will be suspended, allowing interrogation of the variables etc. Serenji will allow you to set breakpoints in a routine.

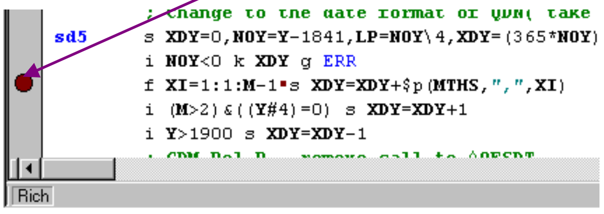
On some platforms, Serenji will also recognize the M Break command **B**. On other systems, a hard-coded breakpoint will be ignored by Serenji, or worse, may invoke the standard command-line debugger facilities. It is therefore best to remove these commands from your code.

Setting Breakpoints

You can set a breakpoint in Serenji in one of **three** ways:

1. By clicking in the gray column alongside the line of code, in the code window. A **red dot** will appear in the column, indicating an enabled breakpoint on that line.

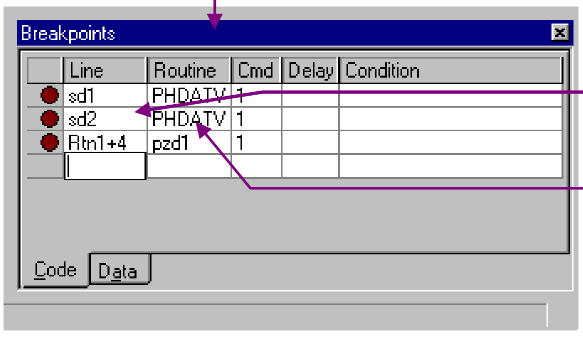
A smaller red dot in the code shows the command to which the breakpoint applies.



```
         ; change to the date format of QDN( take
sd5      s XDY=0,NOY=Y-1841,LP=NOY\4,XDY=(365*NOY)
         i NOY<0 k XDY g ERR
         f XI=1:1:M-1▪s XDY=XDY+$p(MTHS,",",XI)
         i (M>2)&((Y#4)=0) s XDY=XDY+1
         i Y>1900 s XDY=XDY-1
```

2. By right-clicking on a line of code, and choosing "Set Breakpoint" from the context menu - this will set the breakpoint for a particular command within the line.

3. By typing the reference to the line directly into the **Breakpoint window**.



Breakpoints

	Line	Routine	Cmd	Delay	Condition
●	sd1	PHDATV	1		
●	sd2	PHDATV	1		
●	Rtn1+4	pzd1	1		

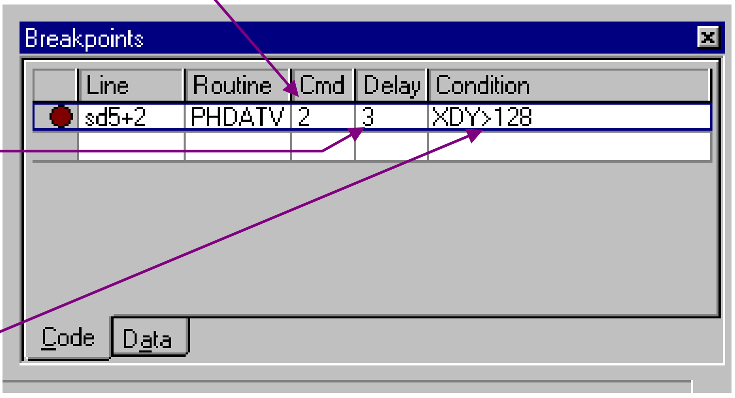
Select a blank line in the Breakpoints window with the mouse and press Enter on the keyboard.

Type the **line reference**, press Tab and then type the **routine name**. Press tab again and type the number of the command on that line at which you wish the code execution to stop.

Method one is by far the simplest, but method three (manually typing in the reference of the breakpoint) allows you to set breakpoints in a routine before the routine is loaded by Serenji. So you can set one or more breakpoints in a routine that you know will be called, before you begin the debugging process. This allows you to let the code run until the required point is reached, rather than having to step through lots of code until the required routine is called, then set the breakpoint. You can also do the same thing by opening any routine in Serenji before it is called and setting a breakpoint in it.

Breakpoints can be quite simple or they can be quite intelligent. For instance you can define a breakpoint for a particular routine, on a particular line, which will take effect only when a particular command (**Cmd**) on that line is reached.

Not only that, but you can also define a number of times that the specified point is reached and the break should be ignored. This is called the **Delay**. This function is particularly useful when debugging inside a FOR loop. Rather than having to step through the loop each time, specify the number of times you want the loop to execute before breaking.



	Line	Routine	Cmd	Delay	Condition
●	sd5+2	PHDATV	2	3	XDY>128

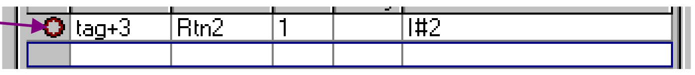
The **Condition** field allows an M or Caché expression to be evaluated whenever the breakpoint is reached. If the expression evaluates to false (zero), the breakpoint is ignored.

Conditions are applied to the breakpoint before the **Delay** is evaluated. If you use both a Delay and a Condition in the same breakpoint, be careful as it may not work as you would expect. When executing the code, Serenji decrements the value of Delay and when it becomes zero the breakpoint will take effect, otherwise it is ignored. However if you also have a condition, **which is always evaluated first**, the value of Delay will only be decremented on the occasions when the condition evaluates to true.

In the example above, the execution of the code will be suspended the third time that the second command on the line is encountered whilst the value of XDY is greater than 128.

Each time you re-run code with a breakpoint which uses a delay, you have to re-enter the value of the delay. This is because the value is decremented each time it the debugger runs through it.

Breakpoints can be either Enabled or Disabled. To keep a breakpoint but temporarily disable it, click once on the red dot in the Breakpoint window.
It will then change to a **red circle**.
This denotes a disabled breakpoint.



○	tag+3	Rtn2	1		I#2

Whilst disabled the breakpoint is ignored by the debugger and code execution will not be stopped by it.

Double-clicking on the header of the first column in the Breakpoint window disables or enables all breakpoints.

The column sizes of the columns within the Breakpoint window can be adjusted by dragging on the divisions between the column headers. Columns can also be re-ordered by dragging their headers.

***NOTES**

The Breakpoint window does not validate entries which are manually entered into it. Therefore you could enter a breakpoint for a line in a routine which does not even exist.

Entries in the Breakpoint window are automatically sorted by routine name, then by order of the line within the routine (provided the routine is loaded).

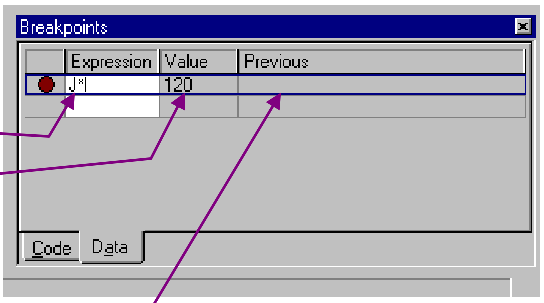
Data Breakpoints

Data breakpoints are sometimes called Watchpoints. An expression is specified and whenever the value of that expression changes, execution of the code will be suspended. This can be anywhere in the code rather than at a particular line or command.

An **Expression** is entered.

The current **Value** of the expression is shown.

Whenever that value changes code execution is suspended and the new value is displayed alongside the **Previous** value



Data breakpoints consume a lot of system resources as each expression is evaluated after each command is executed. If you can use another method to achieve the required result then you may want to do so as the performance of the system will suffer with the use of Data breakpoints.

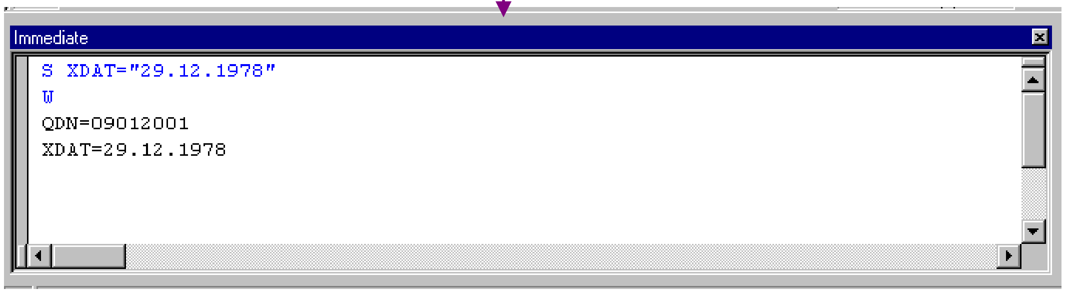
The Immediate Window

The immediate window allows interaction with the session when execution of the code is paused, such as when a breakpoint is reached.

Using M or Caché commands you can set / re-set the value of variables being used in the code.

You can run utilities from the Immediate window provided they do not issue any READ commands.

Below is an example of the use of the **Immediate window**.

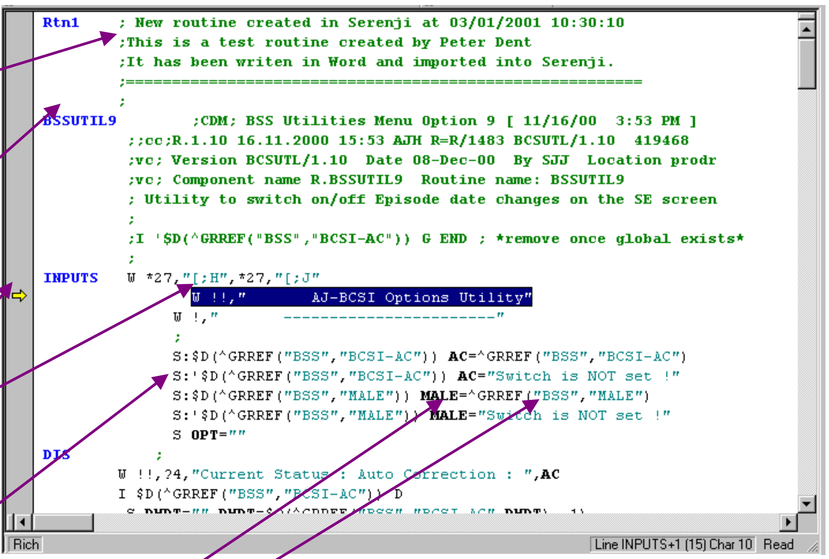


```
S XDAT="29.12.1978"
W
QDN=09012001
XDAT=29.12.1978
```

The Code Window

By default the Code window is the largest of the windows displayed. This is where the code of the routine being written /debugged appears.

By default code is displayed in rich colors. This helps to distinguish the various parts of a routine.



```
Rtn1      ; New routine created in Serenji at 03/01/2001 10:30:10
          ;This is a test routine created by Peter Dent
          ;It has been writen in Word and imported into Serenji.
          ;=======================================================
          ;
BSSUTIL9          ;CDM; BSS Utilities Menu Option 9 [ 11/16/00   3:53 PM ]
           ;;cc;R.1.10 16.11.2000 15:53 AJH R=R/1483 BCSUTL/1.10  419468
           ;vc; Version BCSUTL/1.10  Date 08-Dec-00  By SJJ  Location prodr
           ;vc; Component name R.BSSUTIL9  Routine name: BSSUTIL9
           ; Utility to switch on/off Episode date changes on the SE screen
           ;
           ;I '$D(^GRREF("BSS","BCSI-AC")) G END ; *remove once global exists*
           ;
INPUTS     W *27,"[;H",*27,"[;J"
                W !!,"         AJ-BCSI Options Utility"
                W !,"         -----------------------"
                ;
                S:$D(^GRREF("BSS","BCSI-AC")) AC=^GRREF("BSS","BCSI-AC")
                S:'$D(^GRREF("BSS","BCSI-AC")) AC="Switch is NOT set !"
                S:$D(^GRREF("BSS","MALE")) MALE=^GRREF("BSS","MALE")
                S:'$D(^GRREF("BSS","MALE")) MALE="Switch is NOT set !"
                S OPT=""
DIS                ;
          W !!,?4,"Current Status : Auto Correction : ",AC
          I $D(^GRREF("BSS","BCSI-AC")) D
```

Rich | Line INPUTS+1 (15) Char 10 | Read

Comments appear in bold green.

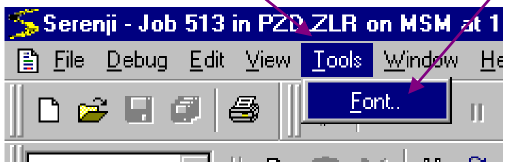
Line labels are shown in bold blue.

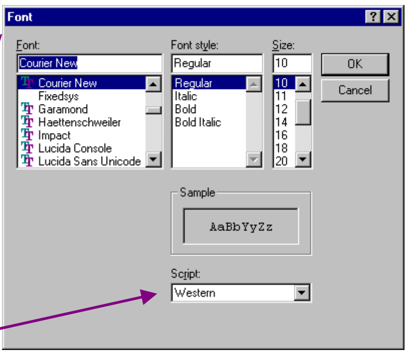
When debugging, the line of code to be **executed next** is highlighted and has an arrow pointing to it.

M or Caché Commands, Global References etc. are in plain black.

Variables are shown in bold black. **Text Values** are shown in Teal.

The size and face of the text used in the code window and the Immediate window can be changed by clicking on the **Tools** menu and choosing **Font…**





Font

Font: Courier New

Font style: Regular

Size: 10

OK | Cancel

Courier New, Fixedsys, Garamond, Haettenschweiler, Impact, Lucida Console, Lucida Sans Unicode

Regular, Italic, Bold, Bold Italic

10, 11, 12, 14, 16, 18, 20

Sample: AaBbYyZz

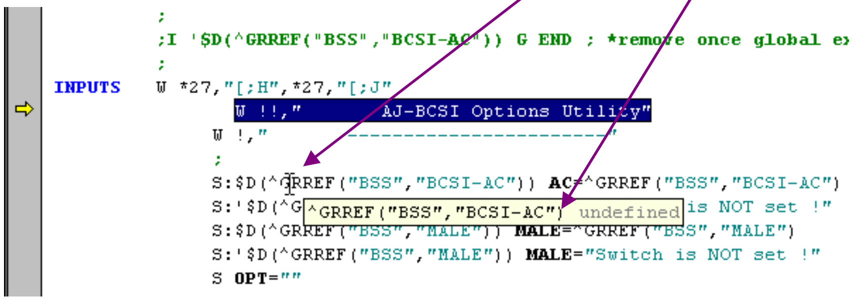
Script: Western

From the **Font** window choose the size of font required.

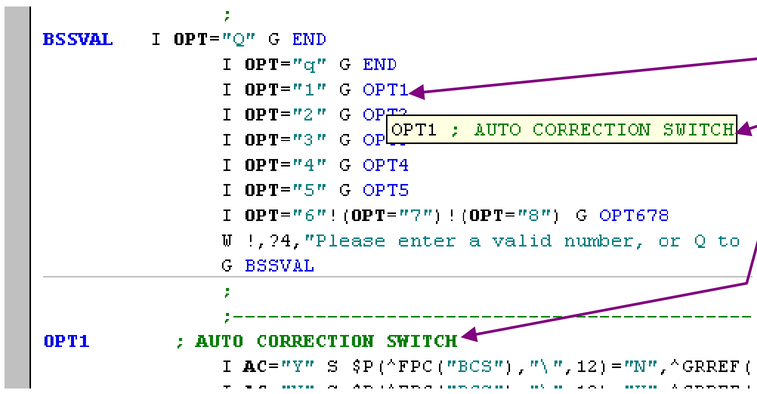
Users in some locales may need to choose a **non-Western script** so that all characters are displayed correctly.

When code is being displayed in debug mode, (i.e. when at a breakpoint), placing the **mouse pointer** over a local or global variable reference, a system variable or an expression will show a **data tip**.

The data tip will show the name of the object and the currently assigned value, if it has one.



```
              ;
              ;I '$D(^GRREF("BSS","BCSI-AC")) G END ; *remove once global ex
              ;
INPUTS        W *27,"[;H",*27,"[;J"
                  W !!,"            AJ-BCSI Options Utility"
                 W !,"          -------------------------"
                 ;
                 S:$D(^GRREF("BSS","BCSI-AC")) AC=^GRREF("BSS","BCSI-AC")
                 S:'$D(^G^GRREF("BSS","BCSI-AC") undefined is NOT set !"
                 S:$D(^GRREF("BSS","MALE")) MALE=^GRREF("BSS","MALE")
                 S:'$D(^GRREF("BSS","MALE")) MALE="Switch is NOT set !"
                 S OPT=""
```



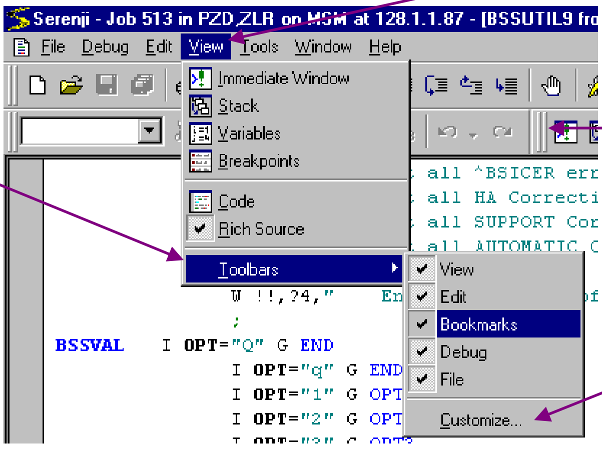
```
                 ;
BSSVAL     I OPT="Q" G END
                 I OPT="q" G END
                 I OPT="1" G OPT1
                 I OPT="2" G OPT2
                 I OPT="3" G OPOPT1 ; AUTO CORRECTION SWITCH
                 I OPT="4" G OPT4
                 I OPT="5" G OPT5
                 I OPT="6"!(OPT="7")!(OPT="8") G OPT678
                 W !,?4,"Please enter a valid number, or Q to
                 G BSSVAL
                 ;
                 ;-----------------------------------------------
OPT1         ; AUTO CORRECTION SWITCH
                 I AC="Y" S $P(^FPC("BCS"),"\",12)="N",^GRREF(
```

Holding the mouse pointer over a **call to a subroutine** or another routine will display any **comment on that line**.

This takes advantage of good coding practices such as providing comment lines at the line label.

Toolbars

By default all available toolbars are displayed. To switch toolbars on or off, from the **View** menu choose **Toolbars** then choose the toolbar required.


Serenji - Job 513 in PZD,ZLR on MSM at 128.1.1.87 - [BSSUTIL9 fro
File Debug Edit View Tools Window Help
Immediate Window
Stack
Variables
Breakpoints
Code
Rich Source
Toolbars
View
Edit
Bookmarks
Debug
File
Customize...

```
            all ^BSICER err
            all HA Correcti
            all SUPPORT Cor
            all AUTOMATIC C
     W !!,?4,"      En      of
     ;
BSSVAL    I OPT="Q" G END
     I OPT="q" G END
     I OPT="1" G OPT
     I OPT="2" G OPT
```

The toolbars are fully configurable, and may be docked or floating. They can be positioned by dragging them by the **gripper** at the left hand end of the toolbar.

Toolbars can be changed or you can define your own, grouping the buttons as required, by using the **Customize…** option.

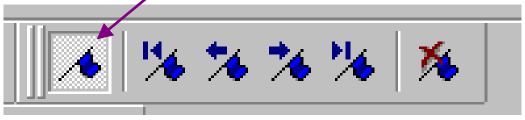
The toolbars all have Tool Tips. If you hold the mouse pointer over any button on the toolbars for a few seconds without clicking, a small label will appear telling you the title of the button. They are mostly self-explanatory and in keeping with most Windows-based applications. There are a few which are peculiar to Serenji and they are explained in the following section.

The Bookmarks Toolbar

Only one button on the Bookmark toolbar is active until the bookmarks are used. The other buttons then become active. Bookmarks are used to mark specified lines of interest in a routine. This will enable the line to be quickly and easily found later. Several bookmarks can be used at any one time and moving between them is managed from the bookmark toolbar.

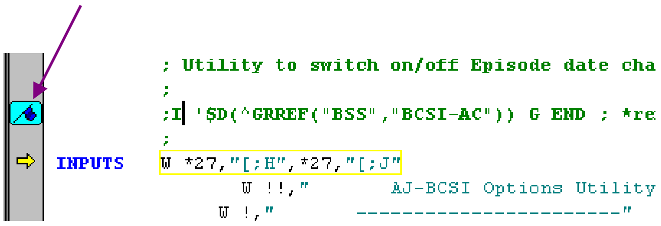
To bookmark any line in a routine, place the cursor on that line and click the **Bookmark** button.

At this point the other buttons on the bookmark toolbar will become active.

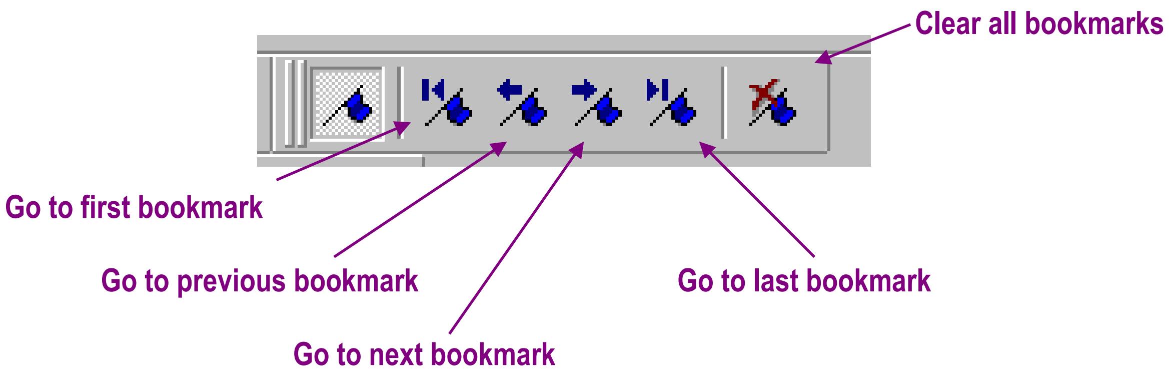


The chosen line in the routine will have a **bookmark flag** alongside it in the left margin.

When you have bookmarks set within a routine, you can navigate between them by clicking on the appropriate button.



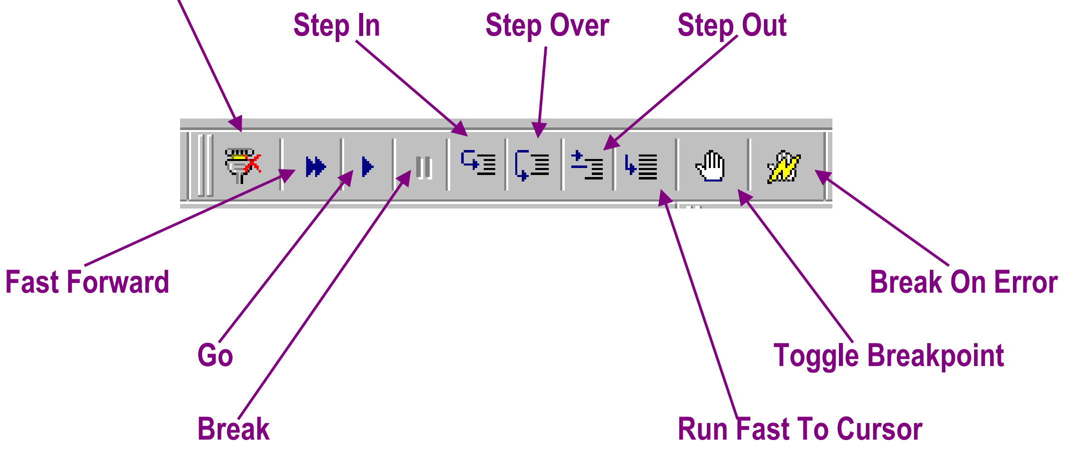
```
                ; Utility to switch on/off Episode date cha
                ;
                ;I '$D(^GRREF("BSS","BCSI-AC")) G END ; *re
                ;
INPUTS          W *27,"[;H",*27,"[;J"
                      W !!,"       AJ-BCSI Options Utility
                    W !,"       ------------------------"
```



Bookmarks only exist for the editing session currently open. If you close a routine, its bookmarks will be lost.

The Debug Toolbar

Apart from the **Disconnect** button, this toolbar only becomes active when a routine is being debugged.



Disconnect

The disconnect button will do just that: it will disconnect your PC’s Serenji session from your session on the host M or Caché system. Any debugging operation or editing session will be discontinued and the normal command prompt will return in the M or Caché session.

In the case of an editing session where a routine has been modified, you will be prompted to save the amended routine before it disconnects.

Step In (F8)

In this mode, code is executed command by command. Each line of code is highlighted **before it is executed**.

To progress to the next command, click the Step In button or press **F8** on the keyboard.

Step Over (Shift+F8)

The Step Over mode is used to skip over a subroutine, routine or extrinsic function call. It will treat such a call as one command. Execution will pause again once control returns from the call.

Step Out (Ctrl+Shift+F8)

This function is used when you have entered a subroutine, routine or function using Step In mode and you decide that no further investigation within that call is needed. Clicking the Step Out button will allow all the code to run up to the point where control returns from the call. Code execution will again pause at that point.

Fast Forward (Shift+F5)

Use Fast Forward to execute code which you don't want to monitor with Go mode. Fast Forward will allow the code to execute until a breakpoint is reached or until a 'Break-on-error' event is triggered.

Serenji cannot interrupt or break into the execution once fast forward has been initiated.

If you choose Fast Forward when there are no breakpoints set and break-on-error is not set, Serenji will revert to normal **GO** mode. You can click on Fast Forward again to force it to run in this mode.

In certain situations (e.g. data breakpoints set) Fast Forward mode is unavailable. Watch the Serenji status line for information.

Go (F5)
Clicking the Go button or pressing **F5** on the keyboard will cause code execution to continue. Execution will pause when a breakpoint is reached or a 'Break-on-error' event is triggered (if this is enabled).

Go mode retains the ability to be interrupted by clicking the Break button. Go mode executes code much more slowly than Fast Forward mode.

Break
When using Go mode or a step operation such as Step Out, you can use the Break button to pause the code execution at the next command.

Run Fast To Cursor (Ctrl+F8)
This feature allows you to run the code quickly to a specified point without having to set a breakpoint. From a paused position in the code, position the cursor on the line you want the code to stop at, then click the Run Fast To Cursor button, or press **Ctrl+F8** on the keyboard, or right-click and choose Run To Line.

It is not possible to break into the code execution during this operation. It is the equivalent of setting a breakpoint, Fast Forwarding to that point, and then deleting the breakpoint.

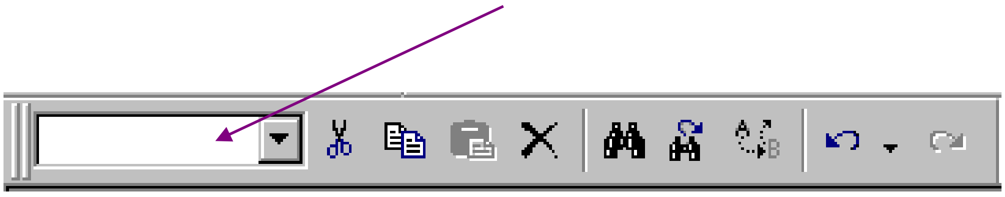
Toggle Breakpoint
This will insert a breakpoint at the first command of the line that the cursor is currently on, or delete one which has already been set.

Break On Error
This button toggles the break-on-error setting on or off. Break on error means that the code execution will pause at the point of an error being detected. When the M or Caché system detects a program error and an error handler routine is about to be entered, code execution stops. The current variables can be examined, and the stepping functions used.

The Edit Toolbar

This toolbar contains some of the standard Windows functions such as Cut, Paste, Delete, etc.

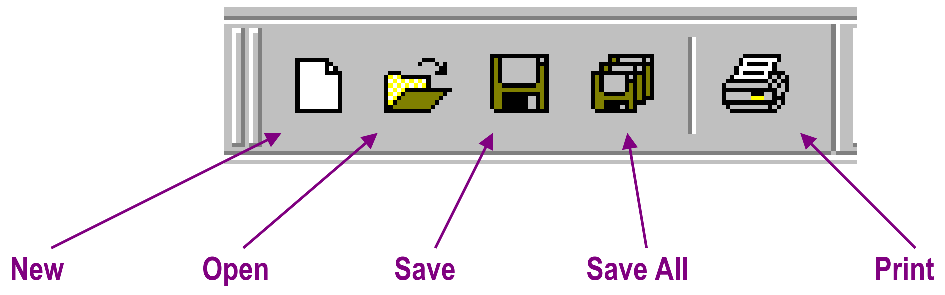
The one function that is not a Windows standard is the **Line Label List**.



When a routine is loaded, in either debug or edit mode, the list contains all the line labels of the routine. Choosing a line label from the list jumps to that line in the code window. You can also type into this list, and the auto-complete function will complete what you are typing. Offsets using + or – can be added to any line label to specify more exactly where you want to jump to.

The File Toolbar

This toolbar allows you to open and close routines in edit mode within Serenji. When Serenji is connected to the host M or Caché system, it can read the names of the routines available in the current UCI /namespace /directory.



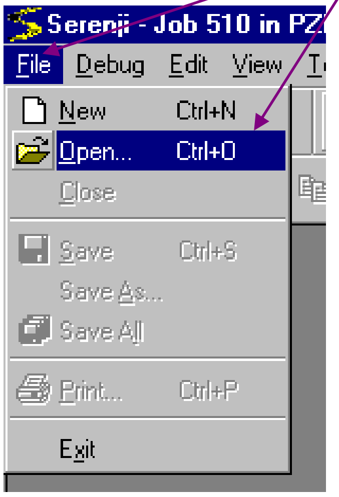
Editing Routines

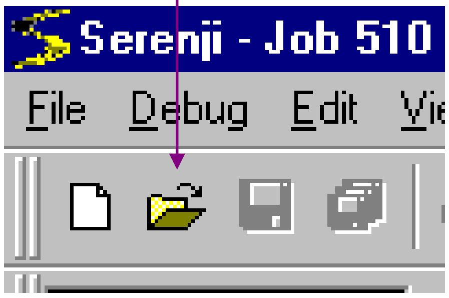
Loading a Routine

Once you have a Serenji session connected to the M or Caché system server (with the shell running), you can load routines into Serenji and take advantage of the advanced Windows-type functionality.

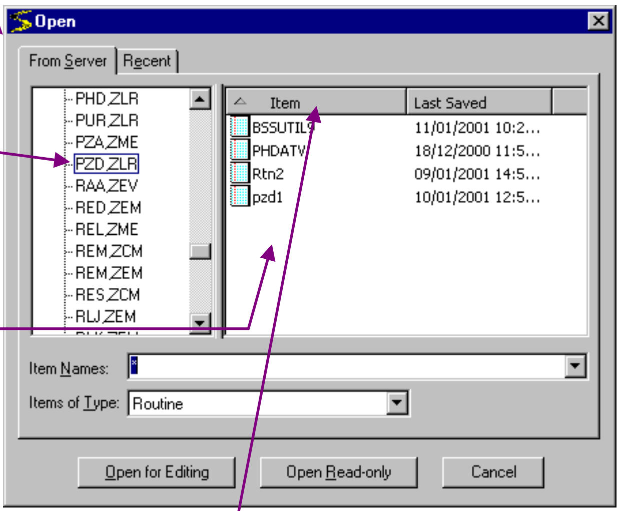
Serenji can be integrated with local configuration management and version control procedures. These may restrict your ability to open a routine for editing.

To open a routine, click on the **File** menu and choose **Open** or click the **Open** button on the toolbar.





The **Open** window will appear after a few seconds.



PHD ZLR
PUR ZLR
PZA ZME
PZD ZLR
RAA ZEV
RED ZEM
REL ZME
REM ZCM
REM ZEM
RES ZCM
RLJ ZEM

Item	Last Saved
BSSUTIL9	11/01/2001 10:2...
PHDATV	18/12/2000 11:5...
Rtn2	09/01/2001 14:5...
pzd1	10/01/2001 12:5...

Item Names: *

Items of Type: Routine

The current **UCI /namespace /directory** will be highlighted.

The items present there, in this case the **Routines** are listed in the right hand pane.

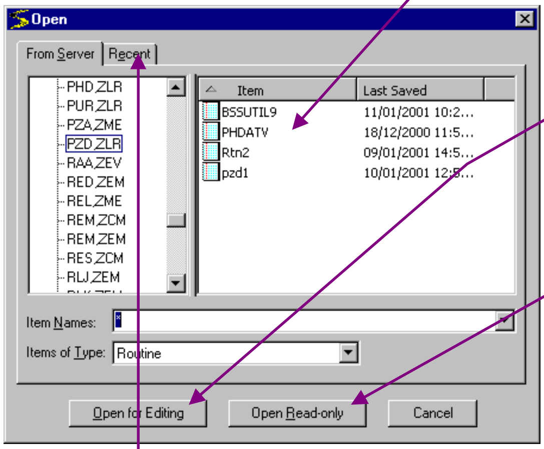
Additional information about the routine is also shown.

The sort order can be reversed each time the item **column heading** is clicked.

You can also sort the items by the date they were last saved.

The 'Item Names' box allows you to select which routines will be displayed. You can use ? as a wildcard to mean any single character and * to mean any number of characters. You can also enter more than one pattern, separating them by semicolons. Whatever you enter will be used to filter the item list so that only those whose names match the pattern(s) are displayed. The eight most recently used patterns

are available to be selected from the dropdown list. Alternatively, you can enter the name of a single item to be opened.
Choose the routine you require from **those listed**. You can also select several routines to open at the same time.



PHD.ZLR
PUR.ZLR
PZA.ZME
PZD.ZLR
RAA.ZEV
RED.ZEM
REL.ZME
REM.ZCM
REM.ZEM
RES.ZCM
RLJ.ZEM

Item	Last Saved
BSSUTIL9	11/01/2001 10:2...
PHDATV	18/12/2000 11:5...
Rtn2	09/01/2001 14:5...
pzd1	10/01/2001 12:5...

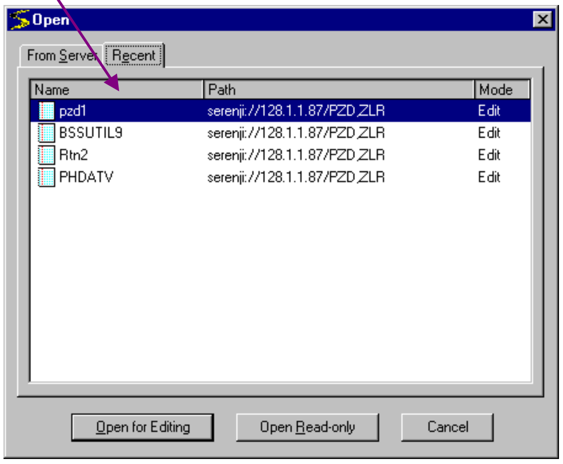
Item Names: *

Provided you have edit rights, you can click the **Open for Editing** button to open the routine(s).

Depending on how your local editing policy is implemented, read-only routines may show 'ghosted'. If you do not have edit rights to the routine(s), you can click the **Open Read-only** button.

You can also open routines as read-only even if you have edit rights. This would avoid accidental changes to the routine.

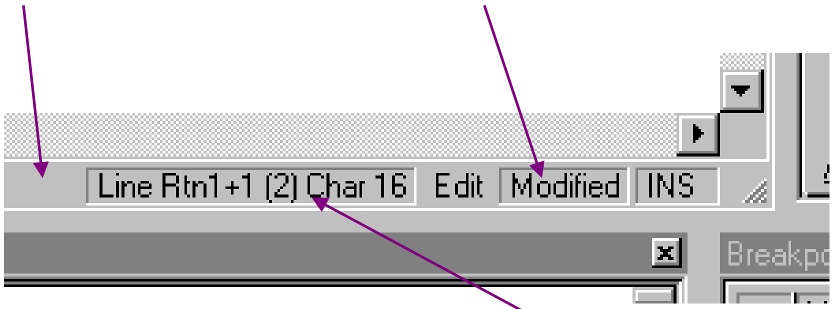
Clicking the **Recent** tab will show a list of the **Recently Opened Items**.



Name	Path	Mode
pzd1	serenji://128.1.1.87/PZD.ZLR	Edit
BSSUTIL9	serenji://128.1.1.87/PZD.ZLR	Edit
Rtn2	serenji://128.1.1.87/PZD.ZLR	Edit
PHDATV	serenji://128.1.1.87/PZD.ZLR	Edit

Checks carried out during the load may mean that the routine can only be opened in Read-Only mode - for example if you do not have edit rights to it. You will then be asked if you wish to proceed with the open.

If a routine is open for editing and changes are made to the routine, or a new routine is created, the **status line** will indicate that **changes have been made** to the routine.



The status line also indicates the current **cursor position** within the routine. You can double-click here to open the GoTo dialog.

Saving a Routine

Routines can be saved only once they have been amended. Click on the Save (or the Save All) button on the File toolbar, or enter Ctrl+S.

Frozen Routines

Routines can be edited during a debug session provided they are not in the current execution stack, (i.e. provided the debugger is not currently executing any of that routine's code).

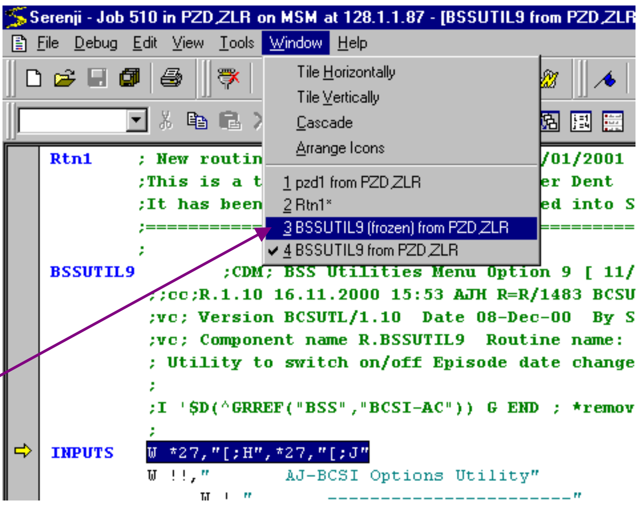
If a routine is loaded for editing and then the routine is entered in debug mode, the routine in the Edit window will be frozen.

In this example the routine ^BSSUTIL9 was first opened in edit mode.

Then the routine was opened in a debug session when execution stepped into it.

Both versions of the routine remain open in the Serenji session but the edit version is now **frozen** and can no longer be edited.

It will remain like this either until the debug session ends or until the code execution leaves the BSSUTIL9 routine.


Serenji - Job 510 in PZD.ZLR on MSM at 128.1.1.87 - [BSSUTIL9 from PZD.ZLR
File
Debug
Edit
View
Tools
Window
Help
Tile Horizontally
Tile Vertically
Cascade
Arrange Icons
1 pzd1 from PZD.ZLR
2 Rtn1*
3 BSSUTIL9 (frozen) from PZD.ZLR
✓ 4 BSSUTIL9 from PZD.ZLR

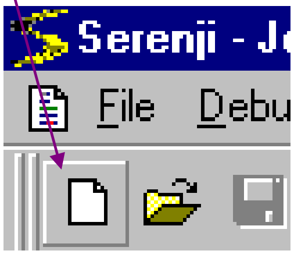
```
Rtn1      ; New routin                              /01/2001
          ;This is a t                              er Dent
          ;It has been                              ed into S
          ;==========                               ========
          ;
BSSUTIL9       ;CDM; BSS Utilities Menu Option 9 [ 11/
           ;;cc;R.1.10 16.11.2000 15:53 AJH R=R/1483 BCSU
           ;vc; Version BCSUTL/1.10  Date 08-Dec-00  By S
           ;vc; Component name R.BSSUTIL9  Routine name:
           ; Utility to switch on/off Episode date change
           ;
           ;I '$D(^GRREF("BSS","BCSI-AC")) G END ; *remov
           ;
INPUTS     W *27,"[;H",*27,"[;J"
           W !!,"        AJ-BCSI Options Utility"
```

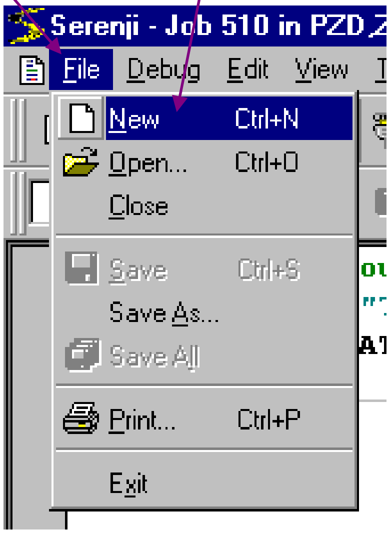
Preserved Routines

If the situation arises where the server-side component of Serenji unexpectedly disconnects from the PC client component and changes have been made to an unsaved routine, a message box will appear to explain how to recover the changes once the server connection has been re-established.

Creating a New Routine

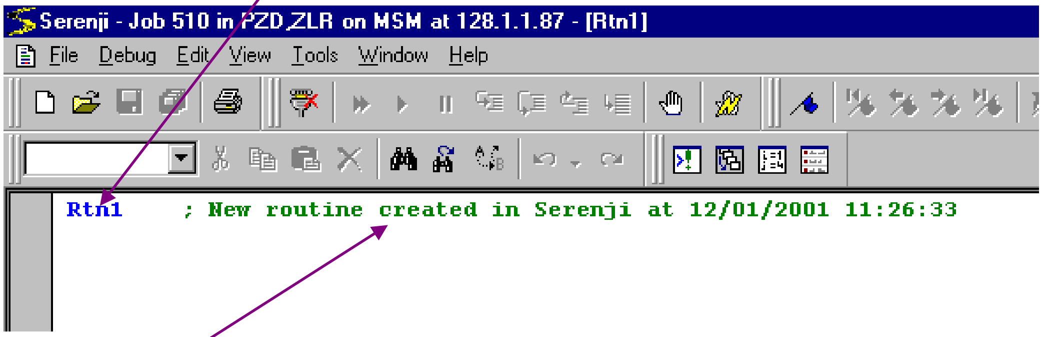
Creating new routines is easy within Serenji. Just click on the **File** menu and choose **New** or click on the **New** button.





A new routine is created in the editor.

A default line label is created **RtnX**, where X is the number of the routine created in the current session of Serenji.



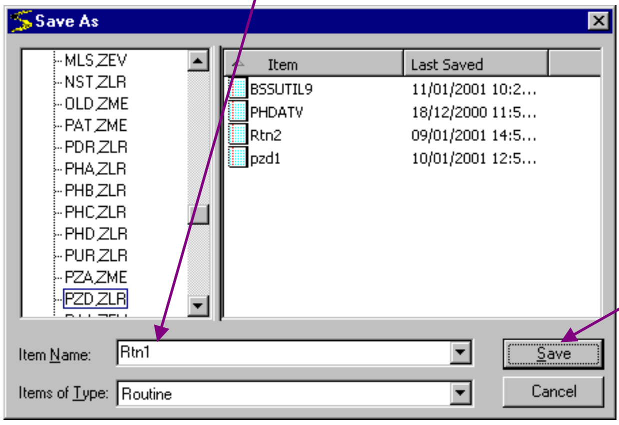
```
Rtn1     ; New routine created in Serenji at 12/01/2001 11:26:33
```

By default a **comment line** is also created as the first line of the new routine.

You should of course change the line label and the comment line.

The new routine can be created in the editor, which functions in the same way as any simple word processor. In the current version lines do not wrap around in the Serenji editor.

The first time you save a new routine you will be prompted to assign the routine a name. The default comes from the label of the first line in the routine. Type the required name into the **Item Name** box and check that the correct UCI /namespace /directory is highlighted.



Item	Last Saved
BSSUTIL9	11/01/2001 10:2...
PHDATV	18/12/2000 11:5...
Rtn2	09/01/2001 14:5...
pzd1	10/01/2001 12:5...

Item Name: Rtn1

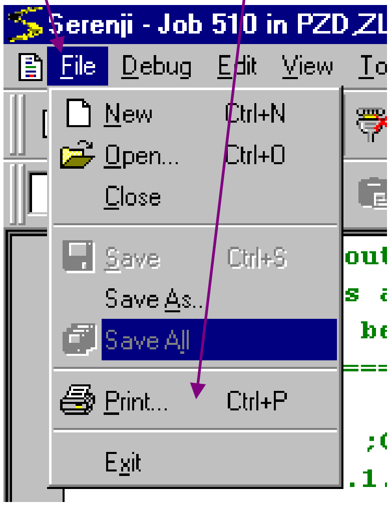
Items of Type: Routine

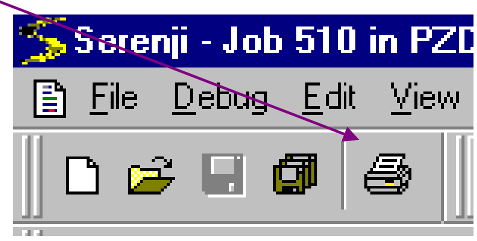
Click the **Save** button to save the routine in the highlighted UCI /namespace /directory with the specified name.

Printing

A print option is provided from within Serenji. You can direct the output to any of your available Windows printers.

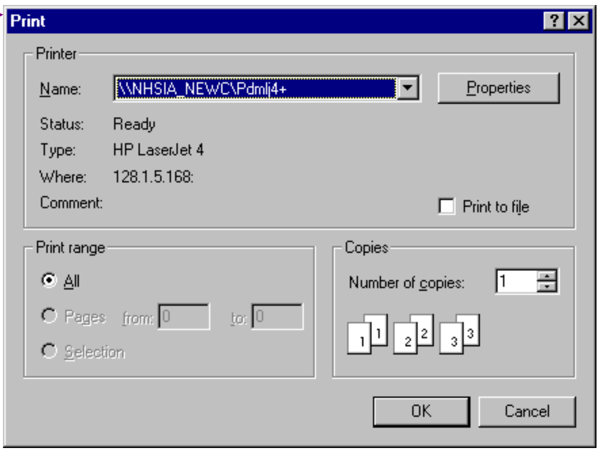
From the **File** menu, choose **Print** or click the **Print** button on the toolbar.





The standard Windows **Print** screen will appear.

Currently, long lines are truncated rather than wrapped on the print-out. Using the Print dialog's Properties button to select Landscape orientation may help work around this limitation.


Print
Printer
Name: \\NHSIA_NEWC\Pdmlj4+
Status: Ready
Type: HP LaserJet 4
Where: 128.1.5.168:
Comment:
Print range
All
Pages from: 0 to: 0
Selection
Copies
Number of copies: 1