



RepGrid • WebGrid • RepGrids  
RepNet • RepDoc • RepScript

# Rep Plus

Conceptual Representation Software

## RepServe Manual

**Managing the Rep Plus Web Server  
and its Scripts, including WebGrid**

May 2021

Brian R Gaines and Mildred L G Shaw

<http://cpsc.ucalgary.ca/~gaines/repplus/>

# Contents

Contents	i
1 Introduction to RepServe	1
1.1 WebGrid . . . . .	1
2 RepServe Operation	2
2.1 The RepServe pane in the Rep Plus Manager window . . . . .	2
2.2 Debugging features . . . . .	3
3 Installing WebGrid as an Internet Service	5
3.1 Hosting WebGrid on a Windows server virtual machine . . . . .	6
3.1.1 Router IP binding . . . . .	6
3.1.2 Security . . . . .	6
3.1.3 Autostart after a virtual server restart . . . . .	6
3.1.4 Remote Management . . . . .	6
3.1.5 Supported web clients . . . . .	7
4 RepServe Scripting	7
4.1 RepServe operation . . . . .	7
4.2 Example scripts: WebDoc, WebNet and WebGrid . . . . .	7
4.2.1 WebDoc . . . . .	7
4.2.2 WebNet . . . . .	8
4.2.3 WebGrid . . . . .	8
4.3 RepServe scripting architecture . . . . .	8
4.3.1 WebGrid elicitation and editing scripts . . . . .	8
4.3.2 WebGrid scripts for saving and accessing saved grids . . . . .	10
4.3.3 Webgrid analysis scripts . . . . .	10
4.3.4 Webgrid support scripts . . . . .	11
4.3.5 Alternative script subdirectories . . . . .	11
4.3.6 Command sequences . . . . .	12
4.3.7 Contextual help system . . . . .	13
4.4 WebGrid CSS and Javascript Libraries . . . . .	13
4.4.1 WebGrid.css . . . . .	13
4.4.2 WebGrid.js . . . . .	15
4.5 Some sample scripts . . . . .	17
4.5.1 Server.repscript . . . . .	17

4.5.2	Library.repscript . . . . .	20
4.5.3	WebGrid/Main.repscript . . . . .	29
5	Bibliography	38

# 1 Introduction to RepServe

*RepServe* is a scriptable web server shell embedded as an integral component of Rep Plus. The shell supports the HTTP protocol for World Wide Web communication with multiple simultaneous client web browsers. It decodes the incoming encoded HTML requests and passes any variables and values posted from forms to a set of scripts written in RepScript. The scripts decode the content of the request and generate a response in HTML, passing this back to RepServe to return to the requesting client.

*RepScript* is a general-purpose programming language so that the web services provided through RepServe can be of any nature. Since RepScript has access to the library of RepGrid and RepNet functionality, it is particularly well-suited to services relating to the use of grids and nets (see RepScript manual for details). This combination of a robust HTTP server, an object-oriented, incrementally compiled programming language, and a powerful libraries of conceptual representation tools makes RepServe a powerful tool for personal construct studies, knowledge management and market research applications.

The initial set of RepServe scripts provided in Rep Plus support: *WebDoc* for serving documents; *WebGrid* for the interactive entry, elicitation, editing and analysis of conceptual grids; and *WebGrid* for serving nets as clickable images. These scripts also serve to demonstrate how RepServe may be programmed to provide a wide variety of services, enabling users to modify and enhance them for their own purposes.

RepServe queues requests arriving while the current request is being processed and hence can handle multiple simultaneous users. The number of users that can be served at the same time depends on the acceptable average processing time and the actual processing time determined by the speed of the machine. RepServe is not intended to be a heavy-duty document server such as Apache, but rather a specialist conceptual representation tool that incidentally serves documents as part of providing specialist capabilities requiring significant computation.

## 1.1 WebGrid

RepServe was developed initially to make WebGrid functionality available as part of Rep Plus, and the sample scripts provided emulate an enhanced version of our original WebGrid service. WebGrid was first developed in 1994 as a port of RepGrid to operate over the World Wide Web (Gaines, 1995), (Gaines and Shaw, 1996, 1997, 1998; Shaw and Gaines, 1996, 1998). It has also been integrated into other knowledge elicitation systems (Tennison et al., 2002).

The WebGrid server at the University of Calgary has operated on a 24/7 basis from early 1995 through 2015 making it one of the earliest continuous services on the World Wide Web (Gaines and Shaw, 2007). It has been used for conceptual grid elicitation and analysis from over 100,000 sites worldwide and been the basis for experimental data collection for a wide range of academic and commercial studies. There are now several WebGrid servers publicly available worldwide (see <http://cpsc.ucalgary.ca/~gaines/repplus/>).

There are major advantages of offering web-based conceptual representation services:

- first, modern web browsers provide well-designed and robust general-purpose user interfaces, readily programmable through HTML which enables the user interface to be easily customized;
- second, they provide excellent typographic and page layout capabilities enabling dynamically generated pages with variable content to be presented attractively;
- third, they provide strong multimedia support enabling sounds, image and videos to be embedded and used, for example, to illustrate elements in a domain in grid elicitation;
- fourth, access to conceptual representation services over the Internet with no requirement for special-purpose software on the client machine facilitates distributed studies based on remote data collection.

The scripts that provide the WebGrid functionality are text files that may be edited by Rep Plus users to extend and customize WebGrid, or use as examples on which to base other forms of interaction. The text in the scripts may be translated into other languages to support versions of WebGrid in French, German, Dutch, Swedish, and so on. Since the text is in Unicode it is also possible to support non-Roman languages such as Chinese, Japanese, Arabic, Devanagari, and so on. Provision is made for the support of multiple simultaneous versions of WebGrid so that different customizations and translations can all be made accessible on one RepServe machine.

## 2 RepServe Operation

This section describes some of the technical aspects of RepServe operation. The Rep Plus interface is the same under Windows and OS X, with minor differences when native widgets are used, and illustrative screenshots have been taken from both systems.

### 2.1 The RepServe pane in the Rep Plus Manager window

RepServe is designed to run unattended. It keeps a record of the state of the controls in its window and sets them up automatically when Rep Plus is started. Hence, most users do not need to access the RepServe pane. Even those developing scripts can usually check their correct operation by testing them from a web client. However, sometimes it may be useful to monitor the server operation at a more detailed level and, if so, the capabilities are available.

The RepServe pane is accessed from the *Rep Plus Manager* window by clicking the *RepServe* tab at the top which displays the RepServe pane as shown in Figure 1.

The top line of the pane shows a set of controls for managing and monitoring the normal operation of the the server.:

- The *Port* field specifies the port on which RepServe should listen for HTTP communications. It is set by default under Microsoft Windows to port 80, the standard default port for the World Wide Web. It is set by default under Apple OS X to port 8080 since the apple unix

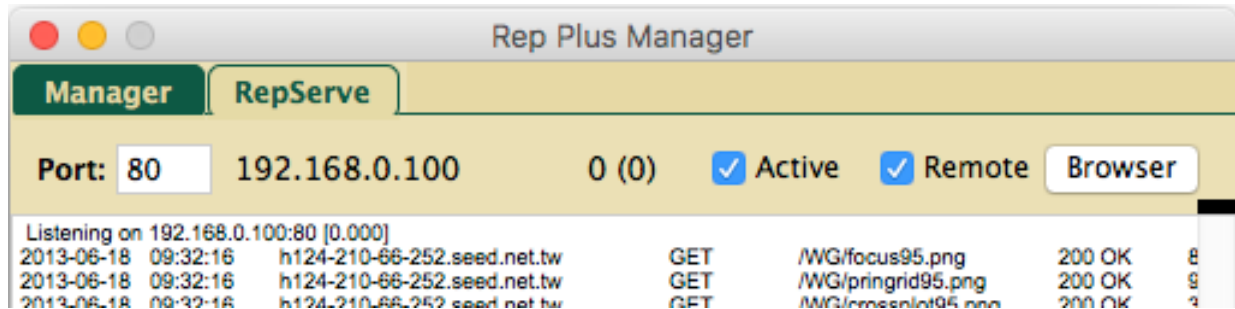


Figure 1: The RepServe pane in the Rep Plus Manager window

operating system does not allow the port to be set below port 1024 unless the application is running as a root application. Under Windows the port can be changed to any number but should not be below 1024 unless it is set to 80. Under OS X it may be set to any port not below 1024. For those using the WebGrid server on their local machine the default settings will usually be appropriate. Note that many port numbers above 1024 are regarded as *reserved* for particular applications, and that an institutional firewall may not allow access to a particular port unless you request it, so check with your IT services about before setting an Internet service.

- The Internet address of the machine follows in order to monitor the address at which RepServe is listening.
- The two numbers following show the currently active connections and, in parentheses, the maximum number of concurrent connections since the server was started in order to provide an indication of the load on the server.
- The *Active* check box makes RepServe active when checked, that is listening at the Internet address shown on the specified port. It can be unchecked to stop RepServe listening.
- The *Remote* check box specifies whether requests will be accepted across the Internet. It may be unchecked to restrict requests to come from a client on the same machine as RepServe.
- The *Browser* button starts the default browser on the local machine and requests the URL on which RepServe is listening, in this case <http://192.168.0.100:80>, fetching the default page */ServerDocs/index.html* which provides an overview of WebGrid.
- The text window at the bottom of the RepServe pane logs the server activity in the *Common Log Format* for web servers.

## 2.2 Debugging features

The black box on the right of the RepServe pane may be dragged upwards to leave only the log window exposed, or down to show additional sever options, intended primarily to support script debugging. Figure 2 shows it fully dragged down to make the following features accessible:

- The *Window* check box when checked specifies that the activity should be logged in the window (usually set to be limited to the 100 most recent events).
- The *File* check box when checked indicates that the activity log should also be appended to the file named */ServerLogs/Log*.
- The *Debug* check box when checked specifies that the common log format should be extended with additional information that may be useful in monitoring the performance of server scripts. This information is specified by a server script (as is the common log format data) and hence may vary. For the supplied WebGrid scripts it comprises the transaction time, the number of active objects, the memory in use, and the UID of the grid being processed.
- The *Client* check box when checked specifies that the common log format should be extended with the client identification supplied by the remote web browser. This information may be useful if there are problems with supporting HTML on particular types of browser.
- The *Capture* check box when checked specifies that new grids (as contrasted with demonstration grids) should be filed in the directory */ServerLogs/Capture/* using their UIDs as their file names. This is useful in monitoring remote studies and in supporting users who forget to save their grids locally. It should be used with care as it may raise privacy concerns in some situations.
- The *Clear* button clears the log window and, more importantly, also clears the cache of the scripts that have been dynamically compiled and cached in the operation of the server. This is useful when scripts are being changed and one wishes the changed script to be used rather than that previously cached.
- The *Email* field allows an email address to be entered that may be incorporated in the HTTP header and shown in web pages in order to provide remote users with a point of contact, usually the server manager.
- The *Sockets* checkbox when checked causes details of each socket process to be written in the log window.
- The *Request* checkbox when checked causes the raw request from the client to be written in the log window.
- The *Data* checkbox when checked causes the processed request as pairs of names and values to be written in the log window.
- The *Reply* checkbox when checked causes the HTML reply to the client to be written in the log window.

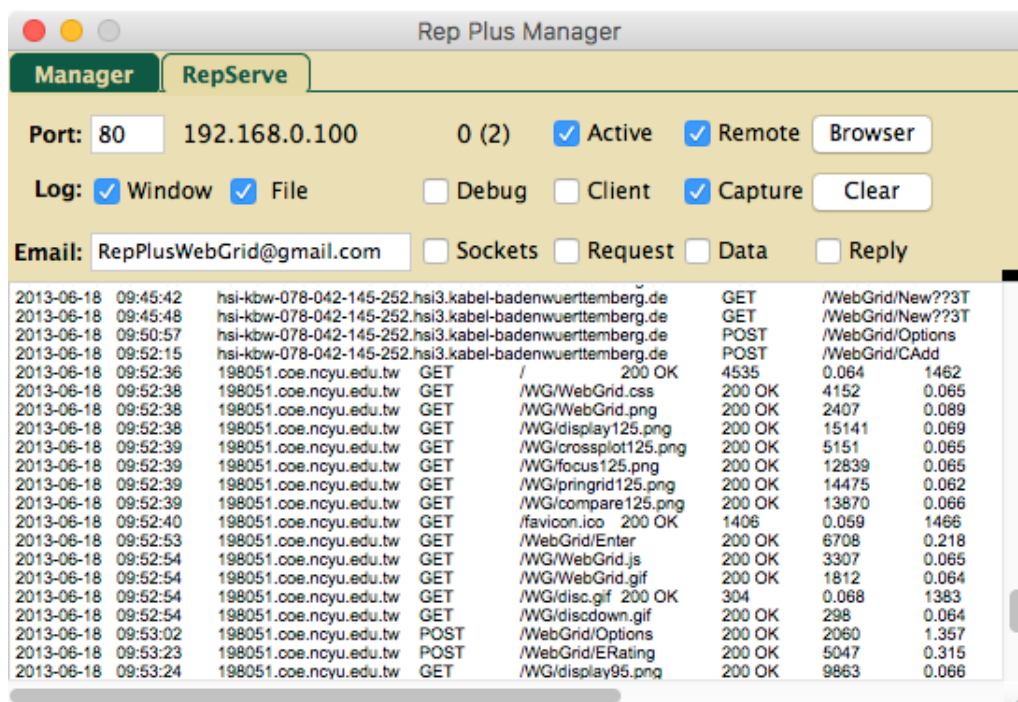


Figure 2: Full set of options in RepServe pane

### 3 Installing WebGrid as an Internet Service

When RepServe is used to provide WebGrid services on a computer running Rep Plus, its functionality is well-integrated with the other grid programs, such as RepGrid, and its usage is generally transparent with no security concerns. When it is used as an Internet server to make WebGrid capabilities available to remote users then appropriate precautions need to be taken to address potential security issues.

The Windows version of Rep Plus is the most appropriate for use as an Internet server as the Windows server operating system allows it run without super-user privileges to provide an HTTP server accessible on the standard web port, 80. Rep Plus will run under any Windows OS version from XP up, and WebGrid servers have successfully operated under Windows Server 2003, 2008 and 2012, and it should operate under Windows Server 2016. The security features of a Windows server virtual machine may be used to sandbox WebGrid operation to localize the consequence of any malicious hacking to the WebGrid service itself. The Windows server may operate stand-alone or as a virtual server under another operating system such as Linux.

Rep Plus/WebGrid Plus is distributed as a standard Windows installer file, *setup.exe*, created by Inno Setup Compiler. It is not code signed (and care should be taken that it has been downloaded from a trusted distribution point, currently [cpsc.ualgary.ca](http://cpsc.ualgary.ca)), and hence, if the server is well-protected, will require one or more reassurances to the protection system that it is known to be a trusted program.

### 3.1 Hosting WebGrid on a Windows server virtual machine

The default settings for RepServe under Windows are that it is active on port 80 as a HTTP service accessible on the Internet, so running Rep Plus should provide a WebGrid service through the IP address of the host machine with no further adjustment.

Section 2 provides detailed information on the controls. The *Active* and *Remote* checkboxes should be set in server mode, and this is the default condition.

#### 3.1.1 Router IP binding

It is recommended that the IP address of the virtual server hosting WebGrid be bound to a symbolic address commencing with *webgrid* followed by the hosting institutions symbolic address, e.g. *webgrid.uvic.ca* or *webgrid.kmlab.edu*. This provides a web address that is both easy to remember and associated with the host institution.

#### 3.1.2 Security

The WebGrid server scripts are designed to prevent hacker attacks aimed at accessing files other than those intended to be accessed, and the specialist nature of the server protects it from other known forms of attack.

However, for maximum security it is recommended that Rep Plus be run without administrator privileges. Such privileges will normally be necessary in order to maintain it by updating its files, but this should be done from a separate account with administrator privileges that is not used to run it.

#### 3.1.3 Autostart after a virtual server restart

If the Windows server is restarted, typically after OS updates, then Rep Plus need to be restarted. This can be automated by setting the main server to auto-login the virtual server hosting Rep Plus, and the virtual server to autostart Rep Plus on login.

#### 3.1.4 Remote Management

Microsoft remote desktop may be used for remote management of the virtual server and WebGrid. The normal management requirements are:-

- Mounting new versions of Rep Plus or of the WebGrid scripts;
- Clearing the log file occasionally if the File checkbox in the RepServe control pane is set;
- Clearing the Capture directory occasionally if the Capture checkbox in the RepServe control pane is set.

### 3.1.5 Supported web clients

WebGrid Plus 2020 supports the latest versions of the primary web browsers, notably Google Chrome, Microsoft Edge and Apple Safari. It uses some HTML features, such as **display: flex** and **input type = "color"**, which were not supported on many earlier web clients.

## 4 RepServe Scripting

RepServe operates by receiving requests over the Internet, or an intranet, encoded in the HTTP protocol. It passes the raw request string to a master script, `Server.repscript`, that decodes it and call other scripts to create a response string in the HTTP protocol which it returns to the requester. The following sections provide more details of the process.

### 4.1 RepServe operation

When the "Active" checkbox is activated it launches a socket that operates asynchronously and listens for incoming HTTP requests on the specified port. When one is received it launches an asynchronous process that calls the script `/ServerScripts/Server.repscript` and passes it the URL as the first item of a vector store with an empty name that can be accessed as `vGet(0)`. If it is a POST request from a form it also passes the name/value pairs from the form in a hash store that can be accessed by a script through the `GetPost` and `CheckPost` functions.

The script `Server.repscript` uses the URL passed to determine what script should handle the request. It provides the RepDoc functionality itself, returning the documents requested by the client, generally stored in `/ServerDocs`. It passes URL's commencing with *WebGrid* to the script `/ServerScripts/WebGrid/WebGrid.repscript` and that script, together with a family of associated scripts support the WebGrid functionality.

Note that `Server.repscript` can be changed to provide additional or completely different functionality. RepServe is a general-purpose, script-driven web server shell. In practice it is expected that most users of Rep Plus will leave the provided functionality in place, enhance it with other functionality, and customize the provided WebGrid functionality. A typical enhancement might be a general questionnaire data collection facility or a link to a description logic inference engine.

The RepScript Manual describes the general features of the RepScript programming language together with the additional features available when scripting RepServe.

### 4.2 Example scripts: WebDoc, WebNet and WebGrid

Example scripts are provided for three services, WebDoc, WebNet and WebGrid.

#### 4.2.1 WebDoc

WebDoc serves documents having URL's of the form `http://documentpath` from the directory `/WebDocs/` looking first in the `/User/Documents/Rep Plus` or `/User/My Documents/Rep Plus` directories and, if

the document is not found in the directory where the Rep Plus application resides. The first field in document path must not be *WebGrid* or *WebNet* as these are used as keywords to indicate the other services.

WebDoc has a macro substitution capability in that fields at the end of the URL separated by ? are substituted for tags within the document having the form  $\langle Mn \rangle$  where n is a number indexing the string of ?-separated fields, with that after the first ? being number 1. This capability is used in the WebGrid contextual help facilities to pass user-defined terminology to the help documents.

#### 4.2.2 *WebNet*

WebNet serves nets having the URL's of the form *http://WebNet/netpath* from *WebDocs* as above, but expects the document to be a net. If the document name has the form *netname.rnet.png* then WebNet returns a PNG image of the net which may be embedded in a user-defined document. If the document name has the form *netname.rnet.click* then WebNet returns a document having a link to the net embedded in it (for test purposes and to show the form of an embedded link). When a node in a net embedded in a document is clicked WebNet takes the note attached to a node to be a URL and redirects to it.

#### 4.2.3 *WebGrid*

WebGrid is the most complex of the three applications and exposes much of the functionality of RepGrid to web clients. Further details are provided in the following sections.

The WebDoc and WebNet functionality is included in the Server script. The WebGrid functionality is scripted in a set of modular scripts described in the next section.

### 4.3 RepServe scripting architecture

Figure 3 shows the flow structure for the RepServe scripts that are normally issued as examples supporting RepDoc, WebNet and WebGrid. The arrows indicate the main flow of calls between scripts.

The **Library** script comprises a set of utility routines that can be called from any script and support commonly required computations across all scripts, such as generating HTML headers and encoding the grid data in hidden fields.

The **Server** script is that called by RepServe when a request is received. It services some requests itself, notably those for RepDoc and RepNet functionality, and passes others to script families providing more complex services, such as those of WebGrid.

#### 4.3.1 *WebGrid elicitation and editing scripts*

For ease of development and maintenance the WebGrid service implementation is split across some twenty five modular scripts, each providing specific functionality.

When a WebGrid request is received by the Server script it passes it to the WebGrid script which analyzes it and passes it on to an appropriate script. The structure of most of these scripts is

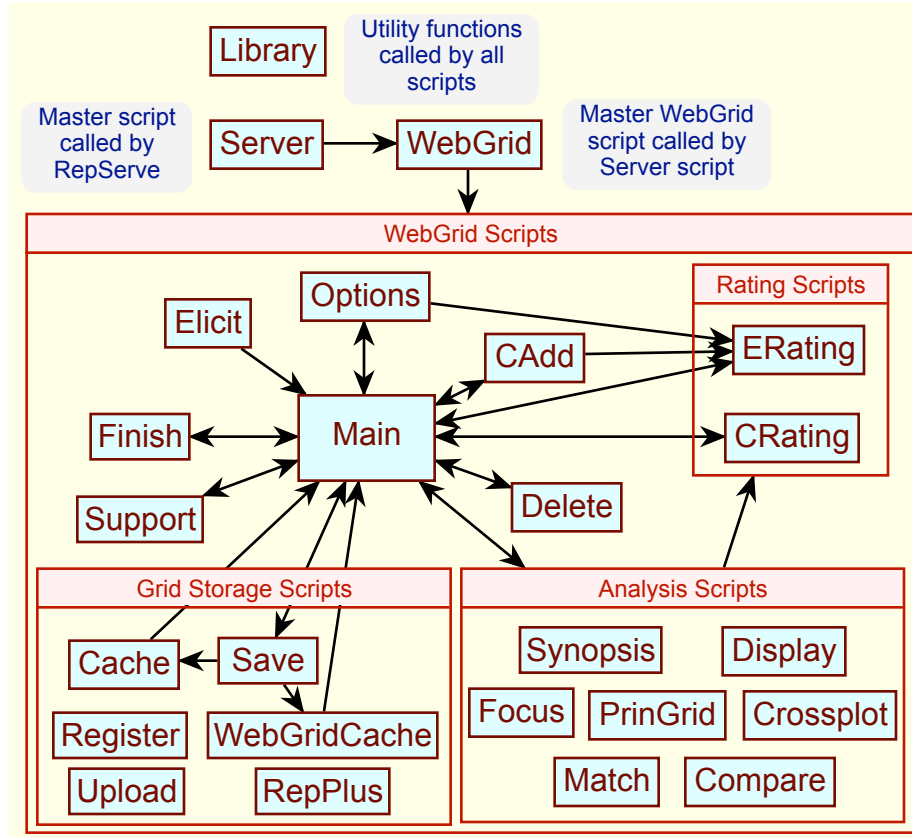


Figure 3: RepServe scripting architecture

based on two routines, one of which writes out the HTML for its associated web page and the other of which analyses the response from that page and calls other scripts to perform the requested service.

The modularity and structure is not necessary. WebGrid could be written as one large script. However, its partition into relatively small modules, each providing routines to generate a specific type of page and to respond to requests from that page, makes the scripts easier to develop, understand, enhance, translate and maintain.

The grid data is stored within hidden fields stored in every WebGrid document. This avoids the need to manage its storage at the server, enables the web clients capability to save web pages to be used to save the grid locally, and makes the client's *Back* button an effective *Undo* feature since it is always possible to revert to a previous version of the grid.

Control in WebGrid generally passes back to the main page generated by the **Main** script which lists recommended actions based on analysis of the grid, list the elements and constructs, provides access to the graphic analysis programs and so on. The Main script can transfer control to nearly all the other WebGrid scripts (except Register, WebGridCache.rgrid and Cache). It does so either in response to the users clicking on a button in the main page or through a command string that can be entered to control the initial sequence of elicitation, for example triadic elicitation of constructs.

The major options associated with setting up grid elicitation are managed through an options page generated by the **Options** script. This also supports commencing a new grid and either going through an interactive guided elicitation process or an expedited data entry process.

The **Elicit** script provides interactive elicitation of elements and significant constructs as an alternative to bulk entry of the elements through the Options script. When the interaction is completed it transfers control to the Main script. The script provided is generic and is readily customized for specific studies.

The **Delete** script prompts the user to delete either selected elements or selected constructs, and transfers control back to the Main script.

The **CAdd** script prompts the user to enter a new construct and then transfers control to the **ERating** script to edit it.

The **ERating** script allows the user to enter or edit the ratings of all the elements on one or more constructs and then transfers control to the Main script.

The **CRating** script allows the user to enter or edit the ratings of one or more elements on all the constructs, including entering a new element, and then transfers control to the Main script.

#### 4.3.2 *WebGrid scripts for saving and accessing saved grids*

WebGrid provides a number of options for saving a grid locally, caching it temporarily at the server for purposes of comparison, or caching it longer term at the server in a password-protected cache directory registered to, and managed by, a user.

When the *Save Grid* button is clicked in the main page the Main script transfers control to the **Save** script which provides various options for saving, caching or downloading the grid. The page generated also informs users how they may save the grid on their local machine and/or download it as a RepGrid file. This script is named *WebGrid.rgrid* because that becomes the value of the form action attribute which is used by the client as the name of the file it downloads.

If the user chooses to cache the grid in the general cache directory control is transferred to the script **Cache** which lists the URLs through which the cached grid may be accessed.

If the user chooses to cache the grid in a registered cache directory control is transferred to the script **WebGridCache** which, if the user's password is valid, caches the grid in the user's registered cache directory and displays the contents of that directory together with the means to control it.

The **Register** script stands alone and is called directly to enable a user to register a cache directory, providing their identification, intended use, and their password to access the cache.

The **Upload** script stands alone and is called directly to enable a user to upload a grid file to continue elicitation or analysis. It accepts grid files in a very wide range of formats from various grid programs, including WebGrid files saved as HTML.

#### 4.3.3 *Webgrid analysis scripts*

WebGrid provides a number of options for analyzing grids and displaying the results in textual and graphic forms. The graphics can be styled and the styles specified for one automatically apply to the

other, and are also interactive providing links to the appropriate editing capabilities. They may also be downloaded in svg, png and jpeg formats for use in other documents.

The **Synopsis** script provides an overview of grid for quality checking, showing a histogram of the ratings of elements on constructs and a scree plot of the variance accounted for by the principal components with an estimate of the number of significant components as an indicator of the complexity of the grid.

The **Display** script displays a matrix of the ratings of the elements on the constructs.

The **Focus** script displays the matrix of ratings of elements on constructs sorted to bring similar items closer together together with hierarchical tree structure of the similarities.

The **PrinGrid** script displays a plot of the principal components of the grid in 1-, 2- or 3-dimensional form.

The **Crossplot** script displays a 2- or 3-dimensional plot of the ratings of the elements on 2 or 3 selected constructs

The **Match** script displays tables of the matches between the elements and between the constructs.

The **Compare** script displays a comparison of two grids having elements, constructs, or both, in common.

#### 4.3.4 *Webgrid support scripts*

The **Support** script provides links to blogs, mailing lists, tutorials, publications and other sources of information relevant to using WebGrid, as well as the opportunity to send comments to the specified email address. It also provides links for string a new grid, uploading grid data and registering a cache.

The **Finish** script provides the opportunity to thank the user for their participation, make sure they know how to analyze and save their data and send comments to the specified email address. This facility is not necessary since interaction can be stopped at any time. However, it has proved to be useful to users to support them when they feel they have finished their interaction with WebGrid. This script could be modified for remote data collection studies to ask users for further information or to check that they have rated all the elements on all the constructs.

#### 4.3.5 *Alternative script subdirectories*

When interaction with WebGrid commences an alternative script directory may be specified as the location of the scripts to be used by putting its name after a ? at the end of the URL. For example, the URL `http://myserver/WebGrid/Open/cars/hatchback.rgrid?Spanish` would access the script `/ServerScripts/WebGrid.repscript` with the command to open the grid `ServerData/cars/hatchback.rgrid` and transfer to the script `/ServerScripts/Spanish/Main.repscript` which might be the Main script translated into Spanish.

All subsequent scripts would be called from the `/ServerScripts/Spanish` subdirectory. However, if a particular script did not exist in that directory then an attempt would be made to call it from the directory that contains it, in this case the default `/ServerScripts` directory. This enables some scripts to be customized without requiring this to be done for all of them.

When alternative script subdirectories are available a menu listing them is shown on the Options page in the *WebGrid Customization* section which allows users to select which one to use.

Subdirectories may be nested so that, for example, a *Spanish* subdirectory might have a further subdirectory *Simplified* that offers simplified WebGrid interaction in Spanish.

If a script name contains an underline character `_` then it will not be listed and can only be accessed through a `?` in a URL. This enables special-purpose scripts to be used without being made generally available options. If such a script is being used then the menu providing the option to switch scripts is not shown.

#### 4.3.6 *Command sequences*

A command sequence may be added to the WebGrid URL after a second `?` and is decoded by the Main script to execute the commands prior to any other action. The commands are encoded as a string of command fields separated by semicolons. The command fields have the format:

command arguments

where the command is currently represented by an alpha character and any arguments are separated by commas.

The command for triadic elicitation is:-

T en1, en2, en3

which elicits a construct from a triad of the three elements whose numbers are specified in en1, en2 and en3. There may be from zero to three elements specified and any that are missing or out of range are replaced by randomly selected elements. Thus the command string:

T 1, 2, 3; T 4, 5, 6; T 1, 3, 5; T 2, 4, 6

can be used to start elicitation with four specified triads. The command strings:

T; T; T and T 1; T 2; T 3

start elicitation with three randomly chosen triads, or three partly specified and partly random, respectively. The command can be preceded with a number specifying it be repeated that number of times, e.g. *3T* is equivalent to *T; T; T*.

The command for construct rating, generally used for given constructs, is:

C n

which shows the construct rating page for construct number n. If n is missing it is taken as 1. The command can be preceded with a plus sign to generate successive construct rating pages starting with that specified, e.g. *+C* will generate rating pages for all given constructs.

The commands, *D*, *F*, *P* and *X* generate Display, Focus, PrinGrid and Compare analyses, respectively. The command *O origin* sets up the origin of the grid to be as specified to enable grid comparison. These commands are used in the WebGridCache.rgrid script to support the analysis of grids in a cache.

### 4.3.7 Contextual help system

One of the facilities provided by the Library script is a contextual help system that allows a clickable ? icon to be inserted in a WebGrid page, and a specified help document to be opened if the icon is clicked. The help documents are held in the directory */ServerDocs/WG/Help/* and are always opened in the same window that can be positioned by the user and kept open.

The user terms for *element*, *elements*, *construct* and *constructs*, and the user-defined purpose are passed as parameters in the URL generated for a help document, and the macro-substitution facilities in WebDoc are used to embed them appropriately in the help text. This enables the help messages to be customized to be more readily understood by the user.

## 4.4 WebGrid CSS and Javascript Libraries

The appearance of WebGrid pages is determined by a stylesheet, and some of the functionality is provided by a Javascript library. both of which may be edited to change the appearance and operation.

### 4.4.1 WebGrid.css

WebGrid pages are linked to the cascading style sheet, */ServerDocs/WG/WebGrid.css*, which defines a set of standard styles for WebGrid pages. Users may override these by incorporating style definitions in their grid options data. The predefined styles are:

```
* {font-family: sans-serif; font-size: 12px; line-height: 1.4; color: #0e5944;}

body {max-width: 800px; margin: 0 auto; padding: 8px; background: #cec495;}

a {color: #006666; background: transparent; text-decoration: none; font-weight: bold;}
a:hover {background: #880000;}

div.a {border: 4px solid #b7ae85; margin-top: 0; padding-top: 2px; background: #e5daa6; text-align: center;}
div.a + div.a {border-top: 0px;}
div.data {background: #ffffe1;}
table {width: 100%; border-collapse: collapse;}
table.ele, table.con {margin: 0 auto; width: auto;}
table.center, table.items {margin: 0 auto; width: auto; text-align: left;}
table.items {border-width: 2px; border-spacing: 0px; border-style: solid; border-color: #0e5944;}
table.items th, table.items td {border-width: 1px; border-style: solid;}

table.cmat, table.emat {margin: 4px auto; width: auto; border-width: 2px; border-spacing: 0px; border-style: solid; border-color: #0e5944;}
table.cmat td, table.emat td {border-width: 1px 0 0 0; padding: 1px 8px; border-style: solid; text-align: center; font-size: 12px;}
table.cmat th, table.emat th {border-width: 2px 0 1px 0; padding: 1px 8px; border-style: solid; font-size: 13px;}
table.cmat tr>td, table.cmat tr>th {text-align: right;}
table.cmat td+td, table.cmat th+th {text-align: left;}
table.cmat td+td+td+td, table.emat td+td {text-align: center; border-left-width: 1px;}
table.cmat tr.a td, table.emat tr.a td {border-top-width: 2px;}
table.cmat th.a {text-align: center;}
table.cmat tr.a {text-align: center;}

table.cache, table.cachecon {margin: 0 auto; width: 99%; border-width: 2px; border-spacing: 0px; border-style: solid; border-color: #0e5944;}
```

```

table.cache td, table.cache th {border-width: 1px; padding: 2px 3px; border-style: solid; border-
    color: #BBDDBB;}
table.cache td, table.cache a {font-size: 11px;}
table.cache td {text-align: right;}
table.cache td.a {text-align: left;}
table.cache tr.a0 {background: #FFFFFF;}
table.cache tr.a1 {background: #FFEEEE;}
table.cache tr.a2 {background: #F7FFCE;}
table.cache tr.a3 {background: #C6EFF7;}
table.cache tr.a4 {background: #E0E0E0;}
table.cache tr.a5 {background: #FFE7C6;}
table.cache tr.a6 {background: #FFFFFF;}
table.cachecon {border-top: 0px;}
table.cachecon td, table.cachecon th {border-width: 0px; padding: 4px 0px;}

h1, h2, h3 {text-align: center; line-height: 1.4; margin: 2px; color: #880000}
h1, h1 * {font-size: 24px;}
h2, h2 * {font-size: 16px;}
pre {font-family: monospace; font-size: 9px; text-align: left;}
input.a {font-family: monospace; font-size: 10px; text-align: left;}
input.b {font-family: monospace; font-size: 12px;}
p {margin: 0px; padding: 4px; border: 0px;}
p.center {margin: 0px; padding: 4px; border: 0px; text-align: center;}
hr {height: 3px; color: #b7ae85; background: #b7ae85; border: 0;}
br.a {clear: all;}
select {box-sizing: content-box; padding: 5px 0;}
input[type="color"] {width: 25px; height: 20px; padding: 0; background-color: transparent; border
    : 0; border-radius: 100%;}

.heading {display: flex; justify-content: space-around; align-items: center; text-align: center;
    margin-bottom: 4px;}

div.note {text-align: left; font-size: 10px; color: #80795d;}
div.left {text-align: left;}
div.center {text-align: center;}
div.right {text-align: right;}
div.hide {display: none;}
div.butr {text-align: right; margin: 4px;}

table.opt td {padding: 4px 2px; text-align: right; font-weight: bold;}
table.opt td.b {padding: 4px 2px; text-align: right; font-weight: normal;}
table.opt td.a {text-align: left; font-weight: normal;}

table.analysis {border-width: 1px;}
table.analysis td {border-width: 2px; padding: 6px; border-style: solid; border-color: #CEC495;
    font-size: 12px; font-weight: bold;}
span.swatch {font-size: 8px; border-width: 7px 0 0 0; border-style: solid; outline: black solid 1
    px;}

button {background-color: #cec495; margin: 0px; font-weight: bold;}
button:hover .dbut a:hover {background-color: #b7ae85;}
button:disabled {background-color: #DDDDDD; color: #AAAAAA;}
button, .butw {width: 100px;}
button.small {width: 48px;}
button.large {width: 150px;}
button.auto {width: auto;}

.dbut {position: relative; display: inline-block;}
.dbut-men {position: absolute; top: 100%; display: none; margin: 0; padding: 0; list-style: none;
    width: 140px;}
.dbut:hover .dbut-men {display: block;}
.dbut a {display: block; padding: 0.2em 0.8em; background: #cec495; color: #0e5944; border: 2px
    solid #c0c0c0; text-align: left;}
.dbut a:hover {background: #b7ae85;}

```

```

span.pop {float: right; color: #579dfe;}

button.wg {width: 70px; height: 70px; font-size: 24px; color: #880000; background: url(/WG/
WebGrid.png); background-repeat: no-repeat;}
button.icon {width: 95px; height: 57px; font-size: 16px;}
button.dis {background: url(/WG/display95.png);}
button.syn {background: url(/WG/synopsis95.png);}
button.foc {background: url(/WG/focus95.png);}
button.pri {background: url(/WG/pringrid95.png);}
button.cro {background: url(/WG/crossplot95.png);}
button.mat {background: url(/WG/match95.png);}
button.com {background: url(/WG/compare95.png);}

span.help {color: white; background: #0e5944; text-decoration: none; font-size: 12px; font-weight
: bold;}
span.help:hover {color: transparent; background: transparent; cursor: help;}
.qw {width: 18px;}
`

```

#### 4.4.2 WebGrid.js

WebGrid pages all include *jquery.js* and the Javascript library, */ServerDocs/WG/WebGrid.js* which defines some common functions used in several WebGrid pages. These are:

```

function bitSet(num,bit) {return num|bit;}
function bitClear(num,bit) {return num&~bit;}

function getHidden(item,def) {
    var els=document.getElementsByName(item);
    if (els.length>0) return els[0].value;
    return def;
}

function addHidden(item,value) {
    var el=document.createElement('input');
    el.type='hidden';
    el.name=item;
    el.value=value;
    document.grid.appendChild(el); // append to form after type set to hidden
    return el;
}

function setGridValue(item,value) {
    var els=document.getElementsByName(item);
    if (els.length==0) {
        addHidden(item,value);
    } else {
        els[0].value=value;
    }
}

function setswatch(el,id,bord) {
    if (!el) return;
    var s=el.value;
    for (var i=s.length; i<6; i++) {s="0"+s;}
    el.value=s;
    s='#'+s;
    var els=document.getElementById(id+'x');
    els.style.backgroundColor=s;
    if (bord) els.style.borderColor=s;
}

```

```

function initswatches() {
    var s;
    for (var i=0; i<initswatches.arguments.length; i++) {
        s=arguments[i];
        setswatch(document.getElementById(arguments[i]),s,1);
    }
}

function help(screen,link) {
    var s;
    s='?' +getHidden('_E','element')+'?' +getHidden('_Es','elements')
    s=s+'?' +getHidden('_C','construct')+'?' +getHidden('_Cs','constructs')
    s=s+'?' +getHidden('_Context','purpose')
    if (link!='') s=s+'#'+link
    w=window.open('/WG/Help/'+screen+'.html'+s,'wghelp','',status=0,toolbar=0,location=0,directories
        =0,menubar=0,resizable=1,scrollbars=1,width=500,height=400');
    w.focus();
}

function showImage(imurl,title,ww,hh) {
    ww=ww+20;
    hh=hh+20;
    w=window.open('',title,'toolbar=0,location=0,directories=0,status=0,menubar=0,resizable=1,width
        ='+ww+',height='+hh);
    var s='<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/
        html4/loose.dtd">'
    w.document.write(s+'<html><head><title>'+title+'</title></head><body><img src='+imurl+'></body
        ></html>');
    w.document.close();
    w.focus();
}

function changeVisible(id,vis) {
    document.getElementById(id).style.display=(vis)?'block':'none';
}

function switchVisible(id,vis,bit){
    changeVisible(id,vis);
    changeVisible(id+'x',!vis);
    var el=document.getElementById(id);
    var n=(el)?el.value:0;
    n=(vis)?bitClear(n,bit):bitSet(n,bit);
    setGridValue('_Control',n);
}

function selInit(selid) {
    var sel=document.getElementById(selid);
    if (!sel) return;
    var opt=sel.options;
    var len=opt.length;
    var n=0;
    for (var i=0; i<len; i++) if (opt[i].selected) n++;
    var dis=n==0;
    for (var i=1; i<arguments.length; i++) {
        var el=document.getElementById(arguments[i]);
        if (el) el.disabled=dis;
    }
}

function selAll(selid,b) {
    var sel=document.getElementById(selid);
    if (!sel) return;
    var opt=sel.options;
    var len=opt.length;
    for (var i=0; i<len; i++) opt[i].selected=b;
}

```

```

}

function xsubmit() {
    if (document.getElementsByName('****').length>0) {return;}
    var a=new Array();
    var j=0;
    var len=document.grid.elements.length;
    for (var i=0; i<len; i++) {
        var el=document.grid.elements[i];
        if (el.type=='button') {
            var nam=el.id;
            if (nam!='DirCache' & nam!='Download') a[j++]=nam;
        }
    }
    len=a.length;
    j=Math.floor(len*Math.random());
    doBtn(a[j]);
}

function auto(tim) {
    window.setInterval('xsubmit()',tim*1000);
}

function doSubmit(name,value) {
    var el=addHidden(name,value);
    var elh=addHidden('*', '*');
    document.grid.submit();
    el.parentNode.removeChild(el);
    elh.parentNode.removeChild(elh);
}

function doBtn(name) {
    doSubmit(name, '')
}

```

## 4.5 Some sample scripts

All the RepServe scripts are accessible in the ServerScripts directory of the Rep Plus application directory. They may be edited but usually it is more appropriate to shadow them with a script having the same name in the ServerScripts directory of the Rep Plus directory in the Documents (or MyDocuments) directory. RepServe will scripts found in the latter directory in preference to the ones in the application.

It may also be appropriate to develop variant scripts by placing them in a subdirectory, for example *dev* of the ServerScripts directory of the Rep Plus directory in the Documents (or MyDocuments) directory. This script will shadow one of the same name in the primary script directories if (as discussed in §4.3.5) WebGrid is accessed with a URL having a question mark followed by the subdirectory name, i.e. *?dev*, after the usual URL.

The following sections list some of the major scripts.

### 4.5.1 *Server.repscript*

The **Server** script manages the process of receiving HTTP requests, queuing them if necessary, and despatching responses.

```

// Called by RepServe when an HTTP request is received

dim URL As string, netpng As Boolean

Function GetExtension() As string
    dim i As Integer, s as string=URL
    do
        i=s.instr(".")
        if i=0 then Exit
        s=s.Mid(i+1)
    loop
    return s
end function

Function ProblemURL() As Boolean // return true if this seems to be an attack (in request use %2F
    for / and %5C for "\" to simulate attack)
    if URL.instr("./")>0 or URL.instr("\.")>0 then return true
end function

Sub CleanCache(tim As Double)
    dim i As integer, t As Double, k As String, s() As String=hKeysA("_"), bnd As Integer=s.UBound
    if bnd=-1 then return
    t=Time-tim
    for i=0 to bnd
        k=s(i)
        if k.Left(1)="/" and k.Right(4)=".png" and val(k.Mid(2,k.len-5))<t then
            // print "Clean cache: "+k
            hRemove(k,"_")
        end if
    next
end Sub

Function GetFileStr() As string
    dim s as string=hGet(URL,"_")
    if s<>"" then
        hRemove(URL,"_")
        CleanCache(10E8) // 100 sec
    else
        s=FileStr("ServerDocs"+URL,"Rep")
        if s="" then s=FileStr("ServerDocs"+URL,"App")
    end if
    return s
end function

Sub GetMacroFile()
    dim t As String, i as Integer, n As Integer=sGetI(URL,"?")
    hEmpty
    for i=2 to n
        hSet(sGet(URL,"?",i),str(i-1))
    next
    URL=sGet(URL,"?",1)
    dim s As String=GetFileStr
    if s="" then
        ScriptCall("$URLNotValid",URL)
    else
        for i=2 to n
            t=str(i-1)
            s=s.ReplaceAll("<M"+t+">",hGet(t))
        next
        ScriptCall("$SendFile",s,"html")
    end if
end sub

Sub LogGet()
    dim s as string

```

```

s=NthField(URL,"/",3)
if s="" then s=GetLog else s=FileStr("ServerLogs/"+s)
ScriptCall("$SendLog",s)
End Sub

Sub DebugGet()
dim s as string
s=NthField(URL,"/",3)
if s="" then
s="<pre>Objects="+Debug("Count")+" Memory="+Debug("Memory")+"</pre>"
s=s+"<pre>"+ScriptsList+"</pre>"
else
s="<pre>"+Debug(s)+"</pre>"
end if
ScriptCall("$SendLog",s)
End Sub

Sub WebNetGet()
dim s,t As string
URL=Right(URL,URL.len-7) // strip off /WebNet
if URL.Right(4)=".png" then // image wanted -- .png for browsers
URL=URL.Left(URL.len-4)+".rnet"
s=GetNetPNG(GetFileStr)
if s="" then ScriptCall("$URLNotValid",URL) else ScriptCall("$OKResponse",s)
elseif Right(URL,6)=".click" then // put out HTML for clickable image
URL=URL.Left(URL.len-6)
s="<html><head><title>Clickable "+URL+"</title></head><body><p align=""center""><a href=""/
WebNet"+URL+"""><img src=""/WebNet"+URL+".png"" ISMAP></p></body></html>"
ScriptCall("$OKResponse",s)
else // click in map
t=URL.NthField("?",2)
URL=URL.NthField("?",1)+".rnet"
s=GetNetClickNote(GetFileStr,sGetI(t,"",1),sGetI(t,"",2))
if s="" then ScriptCall("$NullResponse") else ScriptCall("$RedirectResponse",s)
end if
End Sub

Sub FileGet()
dim s As string
if URL.Instr("?")>0 then
GetMacroFile
return
elseif URL.Right(1)="/" then
URL=URL+"index.html"
elseif URL.Right(9)=".rnet.png" then // .png necessary for browsers to recognize data
URL=URL.Left(URL.len-4)
netpng=true
end if
s=GetFileStr
if s="" then
ScriptCall("$URLNotValid",URL)
else
if netpng then
s=GetNetPNG(s)
ScriptCall("$SendFile",s,"png")
else
ScriptCall("$SendFile",s,GetExtension)
end if
end if
End Sub

Sub ServiceGetRequest()
Select case NthField(URL,"/",2)
case "WebGrid"
ScriptFlow("/WebGrid/WebGrid/GET")

```

```

    case "WebNet"
        WebNetGet
    case "Log"
        LogGet
    case "Debug"
        DebugGet
    else // normal file request
        FileGet
    end select
End Sub

Sub ServicePostRequest()
    Select case NthField(URL,"/",2)
    case "WebGrid"
        ScriptFlow("/WebGrid/WebGrid/POST")
    else
        ScriptCall("$URLNotValid",URL)
    end select
End Sub

// main program

dim action As string

Call xGet("System","System")
select case ScriptState
case "Request"
    Call Time(Time)
    action=NthField(vGet(0),CRLF,1)
    //print "action="""+action+""
    URL=sURLDecode(NthField(action," ",2))
    //print "URL="""+URL+""
    hSet(URL,"URL") // for other scripts -- used once in WebGrid.txt
    if ProblemURL then
        //print NthField(action," ",2)
        ScriptCall("$SendMessage","400 Bad Request\400 Bad Request\Bad Request\The URI "+NthField(
            action," ",2)+" is ill-formed")
    else
        select case NthField(action," ",1)
        case "GET"
            ServiceGetRequest
        case "POST"
            ServicePostRequest
        else
            ScriptCall("$SendMessage","405 Method Not Allowed\405 Method Not Allowed\Method Not Allowed
                \The URI "+NthField(action," ",2)+" is not a valid method")
        end select
    end if
end select
end select

```

#### 4.5.2 *Library.repscript*

The **Library** script provides functionality common to many scripts such as HTTP header and trailer, colour palettes, popup menus, and so on.

```

// Various procedures used as utilities by other scripts
// enter with vGet(0) as action required and vGet(1) as parameter to it
// Last updated: 9-Jan-2016
// 5-Jul-2015 SendData takes filename and extension as second arg
// 15-Jan-2020 OutSwatch uses native Input type="color" and SetupStyle has to take into account
// the # before the color value

```

```

// param: 1 fsize, 2 font, 3 backcolor, 4 tit, 5 el, 6 con, 7 val, 8 connect lines (Display), 9
//         ba1, 10 ba2, 11 ba3, 12 links (PrinGrid), 13 dim, 14 axes, 15 plane, 16 IDFlags, 17 back/BW
//         flags,
// 18 number, 19 scree, 20 compare, 21 scales, 22 class, 23 semantic, 24 logical, 25 values, 26
//         text size, 27 text font, 28 Voronoi

Sub SetupStyle()
    dim s,styl,font,size,tcol,ecol,ccol,vcol,bcol,lcol,mcol,hcol,acol,dicol,pcol,lkcol,lncol,sccol,
        srcol,sncol,sscol,vorcol As string, m,n As Integer
    styl=gGet("Style")
    if CheckPost("TSize",s) then size=str(minmax(val(s),8,48)) else size=sGet(styl,"",1) // font
    size
    if not CheckPost("Font",font) then font=sGet(styl,"",2) // font name
    if not CheckPost("BCol",bcol) then bcol=sGet(styl,"",3) else bcol=Mid(bcol,2) // background
    if not CheckPost("TCol",tcol) then tcol=sGet(styl,"",4) else tcol=Mid(tcol,2) // title
    if not CheckPost("ECol",ecol) then ecol=sGet(styl,"",5) else ecol=Mid(ecol,2) // elements
    if not CheckPost("CCol",ccol) then ccol=sGet(styl,"",6) else ccol=Mid(ccol,2) // constructs
    if not CheckPost("VCol",vcol) then vcol=sGet(styl,"",7) else vcol=Mid(vcol,2) // values
    if not CheckPost("LnCol",lncol) then lncol=sGet(styl,"",8) else lncol=Mid(lncol,2) // line --
    Display
    if not CheckPost("LCol",lcol) then lcol=sGet(styl,"",9) else lcol=Mid(lcol,2) // low rating
    if not CheckPost("MCol",mcol) then mcol=sGet(styl,"",10) else mcol=Mid(mcol,2) // mid rating
    if not CheckPost("HCol",hcol) then hcol=sGet(styl,"",11) else hcol=Mid(hcol,2) // high rating
    if not CheckPost("LkCol",lkcol) then lkcol=sGet(styl,"",12) else lkcol=Mid(lkcol,2) // link --
    PrinGrid
    if not CheckPost("DiCol",dicol) then dicol=sGet(styl,"",13) else dicol=Mid(dicol,2) //
    dimension
    if not CheckPost("ACol",acol) then acol=sGet(styl,"",14) else acol=Mid(acol,2) // axes
    if not CheckPost("PCol",pcol) then pcol=sGet(styl,"",15) else pcol=Mid(pcol,2) // plane
    if not CheckPost("SsCol",sscol) then sscol=sGet(styl,"",21) else sscol=Mid(sscol,2) // scales
    if not CheckPost("SnCol",sncol) then sncol=sGet(styl,"",18) else sncol=Mid(sncol,2) // number
    of components
    if not CheckPost("SrCol",srcol) then srcol=sGet(styl,"",20) else srcol=Mid(srcol,2) //
    comparison plot
    if not CheckPost("ScCol",sccol) then sccol=sGet(styl,"",19) else sccol=Mid(sccol,2) // scree
    plot
    if not CheckPost("VorCol",vorcol) then vorcol=sGet(styl,"",28) else vorcol=Mid(vorcol,2) //
    Voronoi
    if CheckPost("IDID") then m=m+1
    if CheckPost("IDName") then m=m+2
    if CheckPost("IDNote") then m=m+4
    if CheckPost("IDDate") then m=m+8
    if CheckPost("IDTime") then m=m+16
    if CheckPost("NBG") then n=n+1
    if CheckPost("BW") then n=n+2
    s = size+", "+font+", "+bcol+", "+tcol+", "+ecol+", "+ccol+", "+vcol+", "+lncol+", "+lcol+", "+mcol+", "+
    hcol+", "+lkcol+", "+dicol+", "+acol+", "+pcol+", "+str(m)+", "+str(n)
    s= s+", "+sncol+", "+sccol+", "+srcol+", "+sscol+", "+sGet(styl,"",22)+", "+sGet(styl,"",23)+", "+
    sGet(styl,"",24)+", "+sGet(styl,"",25)+", "+sGet(styl,"",26)+", "+sGet(styl,"",27)
    gSet("Style", s+", "+vorcol)
End Sub

Sub OutStyle(whose As String) // whose = "PG" PrinGrid, "CP" Crossplot, "SY" Synopsis otherwise
    empty
    dim styl As String=gGet("Style"), m As Integer=sGetI(styl,"",16), n As Integer=sGetI(styl
    , "",17)
    Out("<tr><td rowspan=""4"" width=""10%"">")
    Out("Style<td colspan=""3"">")
    vCountSet(0, "M") // empty vector
    vPush("Font", "M") // menu name
    vPush(sGet(styl,"",2), "M") // selected item
    vPush("Sans-Serif", "M")
    vPush("Serif", "M")
    //vPush("Monospace")
    //vPush("Cursive")

```

```

OutMenu
Out(" &nbsp; ")
OutNumber("TSize",sGet(styl,"",1),"3","")
Out(" pt &nbsp; ")
Out("Style<td colspan="3">")
OutCheckbox("NBG",bTest(n,1),"")
Out("No Background &nbsp; ")
OutCheckbox("BW",bTest(n,2),"")
Out("Black & White")
Out("<tr><td colspan="6">")
OutSwatch("Back",sGet(styl,"",3),"BCol")
OutSwatch("Title",sGet(styl,"",4),"TCol")
OutSwatch(gGet("E"),sGet(styl,"",5),"ECol")
OutSwatch(gGet("C"),sGet(styl,"",6),"CCol")
if whose<>"CP" then OutSwatch("Value",sGet(styl,"",7),"VCol")
Out("<tr><td colspan="6">")
if whose="" then // Display, Focus, Compare
    OutSwatch("Lines",sGet(styl,"",8),"LnCol")
    OutSwatch("Low",sGet(styl,"",9),"LCol")
    OutSwatch("Middle",sGet(styl,"",10),"MCol")
    OutSwatch("High",sGet(styl,"",11),"HCol")
elseif whose="PG" or whose="CP" then // Pringrid, Crossplot
    OutSwatch("Links",sGet(styl,"",12),"LkCol")
    if whose<>"CP" then OutSwatch("Dimensions",sGet(styl,"",13),"DiCol")
    OutSwatch("Axes",sGet(styl,"",14),"ACol")
    OutSwatch("Plane",sGet(styl,"",15),"PCol")
    OutSwatch("Voronoi",sGet(styl,"",28),"VorCol")
end if
if whose="PG" or whose="SY" then // scree 18 number, 19 scree, 20 compare, 21 scales
    Out("<tr><td colspan="6">")
    OutSwatch("Scree",sGet(styl,"",19),"ScCol")
    OutSwatch("Compare",sGet(styl,"",20),"SrCol")
    OutSwatch("Number",sGet(styl,"",18),"SnCol")
    OutSwatch("Scales",sGet(styl,"",21),"SsCol")
end if
Out("<tr><td colspan="6"><b>Identifier:</b> &nbsp; ")
OutCheckbox("IDID",bTest(m,1),"")
Out("ID &nbsp; ")
OutCheckbox("IDName",bTest(m,2),"")
Out("Name &nbsp; ")
OutCheckbox("IDNote",bTest(m,4),"")
Out("Note &nbsp; ")
OutCheckbox("IDDate",bTest(m,8),"")
Out("Date &nbsp; ")
OutCheckbox("IDTime",bTest(m,16),"")
Out("Time")
End Sub

Sub OutImageScript(imurl As String,imid As String,w As String,h As String)
    Outln("<script type="text/javascript">")
    Outln("$ (document).ready(function(){")
    Outln("    $('dbut-men li a').bind('click',function(){")
    Outln("        s=$(this)[0].id;")
    Outln("        if (s!='') {")
    //Outln("            alert('*'+s+'*');")
    Outln("            if (s=='ICP') {")
    Outln("                showImage('"+imurl+"','"+imid+"','"+w+"','"+h+"');")
    Outln("            } else {")
    Outln("                doSubmit('Image',s);")
    Outln("            }")
    Outln("        return false;}")
    Outln("    });")
    Outln("}");")
    Outln("</script>")
End Sub

```

```

Sub OutImageButton()
    Outln("<div class=""dbut"">")
    Outln("<button type=""button"">Image <span class=""pop"">&#x25bc</span></button>")
    Outln("<ul class=""dbut-men"">")
    Outln("<li><a id=""ICP"" href=""#">Copy to a Window</a></li>")
    Outln("<li><a id=""svg-1"" href=""#">Download as svg</a></li>")
    Outln("<li><a id=""png-1"" href=""#">Download as png</a></li>")
    Outln("<li><a id=""png-2"" href=""#">Download as png x2</a></li>")
    Outln("<li><a id=""png-3"" href=""#">Download as png x3</a></li>")
    Outln("<li><a id=""jpg-1"" href=""#">Download as jpg</a></li>")
    Outln("<li><a id=""jpg-2"" href=""#">Download as jpg x2</a></li>")
    Outln("<li><a id=""jpg-3"" href=""#">Download as jpg x3</a></li>")
    Outln("</ul>")
    Outln("</div>")
End Sub

Function WrapText(s As String, aclass As String) As string
    Dim t,u,b() As String, a() As String=Split(s," "), i,len As integer, bnd As integer=a.UBound
    select case aclass
        case "auto"
            return s
        case "small"
            len=7
        case "large"
            len=21
        else
            return s
            len=14
    end select
    for i=0 to bnd
        t=a(i).Trim
        if t="" then Continue
        if u.len+t.len<len then
            if u="" then u=t else u=u+" "+t
        else
            b.Append u
            u=t
        end if
    next
    if u<>"" then b.Append u
    return Join(b,"<br>")
End Function

Function WGIConStr() As String // return HTML for WGICon making it an active button if server is
    local
    if hGetB("Local","System") then
        return "<button type=""button"" id=""ExportGrid"" onclick=""doBtn('ExportGrid')"" class=""wg
            "">Rep</button>"
    else
        return "<img src=""/WG/WebGrid.png"">"
    end if
End Function

Sub AuthenticateCache(cache As string)
    dim sig,pw,auth,fil,path as string, b As Boolean
    vSet("",1) // assume authorization failed
    fil="ServerData/Cache/"+cache
    path="Rep"
    b=FileExists(fil,path,true)
    if not b then
        path="App"
        b=FileExists(fil,path,true)
    end if
    vSet(path,2)

```

```

if b then
    fil=fil+"/@.txt"
    if FileExists(fil,path,false) then
        sig=sGet(sReplaceEOL(FileStr(fil,path),EOL),EOL,1)
        pw=sGet(sig,2)
        auth=Base64Decode(GetRequestField("Authorization: Basic "))
    end if
    //Print "*" + pw + "*" + auth + "*" + EOL
    if pw<>"" and pw<>auth then
        SendAuthenticate
    else
        if pw="" then vSet("NONE",1) else vSet("VALID",1) // authorization is not required or is
            valid
        end if
    else
        SendMessage("404 Not Found\Cache does not exist\Cache does not exist\The cache you specified
            , "+cache+", does not exist. Go back and try again.")
    end if
End Sub

Sub LogEntry(resplen As integer, code As string)
    dim flgs As integer=hGetI("Flags","System")
    if not bTest(flgs,3) then return
    dim s As String=GetRequest
    dim cmd As String=s.NthField(" ",1)
    dim UID As String
    if cmd="POST" then UID=gGet("UID")
    s=hGet("Date","System")+TAB+hGet("Time","System")+TAB+hGet("RemoteDNS","System")+TAB+cmd+TAB+s.
        NthField(" ",2)+TAB+code+TAB+str(resplen)
    if bTest(flgs,4) then s=s+TAB+format(Time(-1)*10E-6,"0.000")+TAB+hGet("Objects","System")+TAB+
        format(hGetD("Memory","System"),"###,###,###")+TAB+UID
    if bTest(flgs,8) then s=s+TAB+GetRequestField("User-Agent: ")
    if bTest(flgs,2) then FileAppend(s+EOL,"ServerLogs/Log.txt")
    if bTest(flgs,1) then SendLog(s,false)
End Sub

Sub SendResponse(code As string, headers As string, response As string, cache As Boolean)
    // headers should be CRLF terminated or empty, response may be empty
    dim s As String
    LogEntry(response.lenb,code)
    s="HTTP/1.1 "+code+CRLF+"Date: "+hGet("GMT","System")+CRLF+"Server: "+WebGrid+CRLF+"
        Connection: close"+CRLF+"Allow: GET, POST"+CRLF+"From: "+hGet("Email","System")+CRLF
    //s=s+"X-XSS-Protection: 0"+CRLF // only needed to allow javascript in customization
    if cache then s=s+"Cache-Control: max-age=100000000"+CRLF else s=s+"Cache-Control: no-cache, no
        -store, must-revalidate"+CRLF
    SendSocket(s+headers+CRLF+response)
End Sub

Sub SendRedirectResponse(redirectURL As string)
    SendResponse("302 Found","Location: "+redirectURL+CRLF,"",false)
End Sub

Sub SendNullResponse()
    SendResponse("204 OK","","",false)
End Sub

Sub SendAuthenticate()
    SendSocket "HTTP/1.0 401 Authenticate"+CRLF+"WWW-Authenticate: Basic realm=""WebGrid"""+CRLF+
        CRLF
End Sub

Sub SendAuthenticate()
    SendResponse("401 Authenticate","WWW-Authenticate: Basic realm=""WebGrid"""+CRLF,"",false)
End Sub

```

```

Sub SendOKResponse(response As string, cache As Boolean)
    SendResponse("200 OK", "Content-length: "+str(response.lenb)+CRLF, response, cache)
End Sub

Sub SendFile(response As string, content As string)
    dim header As String="Last-Modified: Wed, 22 Jul 2009 11:44:21 GMT"+CRLF // some browser caches
    need a last-modified date
    select case content
    case "png"
        content="image/png"
    case "gif"
        content="image/gif"
    case "css"
        content="text/css"
    case "js"
        content="text/javascript"
    else
        content=""
    end select
    if content<>"" then header="Content-type: "+content+CRLF+header
    SendResponse("200 OK", header, response, true)
End Sub

Sub SendData(response As string, filename As String)
    if filename="" then filename="Untitled.txt"
    //SendResponse("200 OK", "Content-type: "+ "application/x-rgrid; charset=utf-8"+CRLF, response,
    true)
    dim s As String="Content-type: "+ "application/forcedownload"+CRLF
    s=s+"Content-Transfer-Encoding: binary"+CRLF
    s=s+"Content-disposition: attachment; filename="+""+filename+"""+CRLF
    SendResponse("200 OK", s, response, true)
End Sub

Sub SendMessage(mess As string) // mess is: status code\window title\message heading\message
    dim s As string
    s="<html><head><title>"+NthField(mess, "\", 2)+"</title></head><body bgcolor=""#FFDDDD""><font
    color=""#AA0000"" size=""+1"">"
    s=s+"<h1 align=""center"">"+NthField(mess, "\", 3)+"</H1><p align=""center"">"+NthField(mess
    , "\", 4)
    s=s+"</font><p><hr/><address><P align=""right"">Server at "+hGet("IPAndPort", "System")+"</
    address></p></body></html>"
    //SendOKResponse(s, false)
    SendResponse(NthField(mess, "\", 1), "Content-length: "+str(s.lenb)+CRLF, s, false)
End Sub

Sub SendLog(log As string)
    dim s As string
    s="<html><head><title>Log Data</title></head><body bgcolor=""#FFFDD""><font color=""#AA0000""
    size=""+1"">"
    s=s+"<h1 align=""center"">Log Data</h1>"
    s=s+"<pre>"+log+"</pre>"
    s=s+"</body></html>"
    SendOKResponse(s, false)
End Sub

Sub SendURLNotValid(URL As string)
    SendMessage("404 Not Found\404 Not Found\Not Found\The requested URI "+URL+" was not found on
    this server")
End Sub

Sub WGHead(path As String, action As String, title As string, extra As string, multipart As
    Boolean)
    Outln("<!DOCTYPE html>")
    //Outln("<!DOCTYPE HTML PUBLIC ""-//W3C//DTD HTML 4.01 Transitional//EN"" ""http://www.w3.org/
    TR/html4/loose.dtd"">")

```



```

    Out("<input type=""color"" name="" + id + "" value=""# + value + "" id="" + id + "">" + "
        &nbsp;");
End Sub

Sub OutCheckBox(namevalue As string, checked As Boolean, onclick As string)
    dim s As string
    if checked then s=" checked=""checked""
    if onclick<>"" then s=s+"" onclick="" + onclick + ""
    Outln("<input type=""Checkbox"" name="" + namevalue + "" value="" + namevalue + "" + s + "">")
End Sub

Sub OutRadio(name As string, value As string, checked As Boolean, onclick As string)
    dim s As string
    if checked then s=" checked=""checked""
    if onclick<>"" then s=s+"" onclick="" + onclick + ""
    Outln("<input type=""Radio"" name="" + name + "" value="" + value + "" + s + "">")
End Sub

Sub OutSubmit(name As string, value As string, aclass As string)
    dim s As String
    if aclass<>"" then s=" class="" + aclass + ""
    Outln("<button type=""button"" id="" + name + "" onclick=""doBtn(' + name + ')"" + s + "">"+WrapText
        (value,aclass)+"</button>") // workaround for IE6
End Sub

Sub OutButton(value As string, onclick As string, aclass As string)
    if aclass<>"" then aclass=" class="" + aclass + ""
    Outln("<button type=""button"" onclick="" + onclick + "" + aclass + "">"+value+"</button>")
End Sub

Sub OutPassword(name As string, value As string, size As string)
    Outln("<input type=""password"" name="" + name + "" value="" + value + "" size="" + size + "">")
End Sub

Sub OutNumber(name As string, value As string, size As string, extra As string)
    if extra<>"" then extra=" "+extra
    Outln("<input type=""text"" class=""b"" name="" + name + "" value="" + value + "" size="" + size
        + "" + extra + "">")
End Sub

Sub OutText(name As string, value As string, size As string, extra As string)
    if extra<>"" then extra=" "+extra
    Outln("<input type=""text"" name="" + name + "" value="" + value + "" size="" + size + "" + extra + "">")
End Sub

Sub OutTextArea(name As string, value As string, rows As string, cols As string, extra As string)
    if extra<>"" then extra=" "+extra
    Outln("<textarea name="" + name + "" class=""input"" rows="" + rows + "" cols="" + cols + "" + extra
        + "">"+value+"</textarea>")
End Sub

Sub OutHidden(name As string, value As string, id As string)
    if id<>"" then id=" id="" + id + ""
    Outln("<input type=""hidden"" name="" + name + "" value="" + value + "" + id + "">")
End Sub

Sub OutImage(name As string, src As string, wid As string, hgt As string)
    Outln("<input type=""image"" name="" + name + "" src="" + src + "" width="" + wid + "" height="" + hgt
        + "">")
End Sub

Sub OutInput(type As string, name As string, value As string, extra As string)
    if extra<>"" then extra=" "+extra
    Outln("<input type="" + type + "" name="" + name + "" value="" + value + "" + extra + "">")
End Sub

```

```

Sub OutMenu()
    dim i,n As integer, v,s,name,selitem As string
    name = vGet(0, "M")
    selitem = vGet(1, "M")
    n = vCount("M") - 1
    Outln("<select name=""+">")
    for i = 2 to n
        v = vGet(i, "M")
        if v=selitem then s=" selected=""selected"" else s=""
        Outln("<option"+s+" value="+v+" class=""input"">"+v)
    next
    Outln("</select>")
End Sub

Function PtInRect(rect As String, x As Integer, y As Integer, ByRef t As Integer, ByRef h As
    Integer) As Boolean
    dim l,w As integer, s As string
    l=sGetI(rect," ",1)
    t=sGetI(rect," ",2)
    w=sGetI(rect," ",3)
    h=sGetI(rect," ",4)
    if x<l or x>=l+w or y<t or y>=t+h then return false
    return true
End Function

Sub GetHit(sx As String, sy As String, loc As String)
    dim s As String, x,y,t,h As integer, n As Double
    x=val(sx)
    y=val(sy)
    if PtInRect(sGet(loc,";",1),x,y,t,h) then // title hit
        vSet("T",0)
    elseif PtInRect(sGet(loc,";",2),x,y,t,h) then // element hit
        vSet("E",0)
        s=sGet(loc,";",4)
        n=CountFields(s," ")
        vSet(sGet(s," ",Ceil(n*(y-t)/h)),1)
    elseif PtInRect(sGet(loc,";",3),x,y,t,h) then // construct hit
        vSet("C",0)
        s=sGet(loc,";",5)
        n=CountFields(s," ")
        vSet(sGet(s," ",Ceil(n*(y-t)/h)),1)
    else
        vSet("",0)
    end if
End Sub

// main program

select case ScriptState
case "GetHit"
    GetHit(vGet,vGet(1),vGet(2)) // x, y, loc string
case "OutCheckBox"
    OutCheckBox(vGet,vGetB(1),vGet(2)) // name/value, checked as Boolean, onclick
case "OutRadio"
    OutRadio(vGet,vGet(1),vGetB(2),vGet(3)) // name, value, checked as Boolean, onclick action
case "OutSubmit"
    OutSubmit(vGet,vGet(1),vGet(2)) // name, value, class
case "OutButton"
    OutButton(vGet,vGet(1),vGet(2)) // value, onclick, class
case "OutSwatch"
    OutSwatch(vGet,vGet(1),vGet(2))
case "OutNumber"
    OutNumber(vGet,vGet(1),vGet(2),vGet(3))
case "OutText"

```

```

    OutText(vGet,vGet(1),vGet(2),vGet(3))
case "OutTextArea"
    OutTextArea(vGet,vGet(1),vGet(2),vGet(3),vGet(4))
case "OutHidden"
    OutHidden(vGet,vGet(1),vGet(2))
case "OutImage"
    OutImage(vGet,vGet(1),vGet(2),vGet(3))
case "OutInput"
    OutInput(vGet,vGet(1),vGet(2),vGet(3))
case "OutPassword"
    OutPassword(vGet,vGet(1),vGet(2))
case "OutMenu"
    OutMenu
case "WGHelp"
    WGHelp(vGet)
case "WGHead"
    WGHead(vGet,vGet(1),vGet(2),vGet(3),vGetB(4))
case "WGTail"
    WGTail(vGet)
case "OutImageScript"
    OutImageScript(vGet,vGet(1),vGet(2),vGet(3))
case "OutImageButton"
    OutImageButton()
case "OutStyle"
    OutStyle(vGet)
case "SetupStyle"
    SetupStyle
case "OKResponse"
    SendOKResponse(vGet,true)
case "SendFile"
    SendFile(vGet,vGet(1))
case "SendData"
    SendData(vGet,vGet(1))
case "AuthenticateCache"
    AuthenticateCache(vGet)
case "Authenticate"
    SendAuthenticate
case "NullResponse"
    SendNullResponse
case "RedirectResponse"
    SendRedirectResponse(vGet)
case "SendMessage"
    SendMessage(vGet)
case "SendLog"
    SendLog(vGet)
case "URLNotValid"
    SendURLNotValid(vGet)
case "ExportGrid"
    gCopy // export Grid file
    SendNullResponse
else
    SendMessage("Request not Understood\
Request not Understood\
Your Utilities request, "+vGet+", is
not understood")
end select

```

### 4.5.3 WebGrid/Main.repscript

The **Main** WebGrid script is central to all WebGrid interaction, presenting feedback on ongoing grid elicitation and editing, context-sensitive options for further elicitation, options for analysis, and so on. All other scripts return to the Main script when they have completed their activities.

```
// main page when grid loaded -- new grids start at status page and then transfer to main
```

```

// most grid operations are initiated from here and return here

dim ne,nc As integer, te,tes,ten,tc,tcs,tcn As string, eok,cok As Boolean
ne=gGetI("ne")
nc=gGetI("nc")
te=gGetW("E")
tes=gGetW("Es")
if ne=1 then ten=te else ten=tes
tc=gGetW("C")
tcs=gGetW("Cs")
if nc=1 then tcn=tc else tcn=tcs
eok=ne<gGetI("LimitE")
cok=nc<gGetI("LimitC")

Function GetVisible(msk As Integer, ByRef t As String) As String
    dim s As String
    if bTest(gGetI("Control"),msk) then
        s=" class=""a""
        t=" class=""a hide""
    else
        s=" class=""a hide""
        t=" class=""a""
    end if
    return s
End Function

Function GetCount(ByRef t As String) As integer
    dim i,l,n As Integer, s As String
    l=t.len
    i=1
    While i<=l
        if not sNumeric(t.Left(i)) then Exit
        i=i+1
    Wend
    if i>1 then
        n=Val(t.Left(i-1))
        t=LTrim(t.Mid(i))
    end if
    return n
End Function

function SelectElements(inc As string, num As Integer, shuf As Boolean) As string
// enter with a tab-saparted string to be included, possibly empty
// return a tab-separated string of num different element numbers
// randomized if shuf is true
    dim i,k,n As integer, src(),dest() As String
    for i=0 to ne-1
        src.append str(i)
    next
    src.Shuffle
    n=sGetI(inc)
    for i=1 to n
        k=src.IndexOf(sGet(inc,i))
        if k>-1 then
            dest.append src(k)
            src.Remove k
        end if
    next
    n=num-2-UBound(dest)
    for i=0 to n
        dest.append src(i)
    next
    redim dest(num-1)
    if shuf then dest.Shuffle
    return Join(dest,TAB)

```

```

end function

Function TryCommand() As Boolean
// Get and execute a command if any -- return false if reply sent, otherwise true
// D - Display, F - Focus, P - PrinGrid, Xcachedfile -- compare with cachedfile
// Ocachedfile - set the original file as specified (can be empty string)
// Cn - rate construct n, +Cn rate construct n and rest of constructs (for exchange grids), +C
//      equivalent to +C1
// T - triadic elicitation (triad can be empty or up to 3 element numbers), nT do n triads as
//      specified (for compare grids)

    dim s,t,cmd,com,cstr,cache,a() As string, i,j,n,cnt As integer, done,plus As Boolean

// Save the grid if origin is a registered cache
s=gGet("Origin")
cache=NthField(s,"/",2)
If gGetI("Status")>1 and NthField(s,"/",1)="Cache" and cache<>"0" and cache<>" " and FileExists
    ("ServerData/Cache/"+cache+"/@.txt") then
    gSet("Save","ServerData/Cache/"+cache+"/"+gGet("UID")+".rgrid")
end if

// Check for commands
do
    done=true
    plus=false

    // extract command string if any
    cstr=gGet("Command")
    if cstr="" then return true
    com=Trim(NthField(cstr,";",1))
    cstr=sget(cstr,";",2,-1)
    gSet("Command",cstr)

    // decode command string
    plus=com.Left(1)="/" // does it begin with a +
    if plus then com=com.Mid(2)
    cnt=GetCount(com) // does it begin with a count
    cmd=com.Left(1) // first letter is command type
    com=Mid(com,2)
    select case cmd
    case "D" // Display grid
        gSet("Control",str(bSet(gGetI("Control"),8,0))) // show analysis parameters
        ScriptFlow("Display/")
    case "F" // Focus cluster grid
        gSet("Control",str(bSet(gGetI("Control"),8,0)))
        ScriptFlow("Focus/")
    case "P" // PrinGrid Map grid
        gSet("Control",str(bSet(gGetI("Control"),8,0)))
        ScriptFlow("PrinGrid/")
    case "X" // Compare grid with that setup in Origin
        gSet("Control",str(bSet(gGetI("Control"),8,0)))
        ScriptFlow("Compare/")
    case "O" // Setup Origin for comparisons
        gSet("Origin",com)
        done=false
    case "C" // rate given constructs. e.g. C4;C7
        if com="" then n=0 else n=val(com)-1
        if n>=0 and n<nc then
            if plus then gSet("Command","+C"+str(n+2)+"."+cstr)
            ScriptFlow("ERating/Rate "+str(n))
        else
            done=false
        end if
    case "T"
        if ne>=3 then

```

```

        if cnt>1 then gSet("Command",str(cnt-1)+"T"+com+";"+gGet("Command"))
        n=com.CountFields(",")
        for i=1 to n
            a.append str(sGetI(com,",",i)-1)
        next
        ScriptFlow("CAdd/Triad "+SelectElements(Join(a,TAB),3,false))
    else
        done=false
    end if
else
    done=false
end select
loop until done
End Function

sub DoWrite()

dim s,t,note As string, i,j,n,control As integer, x As double, com As boolean

control=gGetI("Control")
ScriptCall("$WGHead",ScriptPath,"Main","Elicitation and Analysis")
Outln("<script type=""text/javascript"">")

Outln("function init() {")
Outln("  setupE();")
Outln("  setupC();")
Outln("}")

Outln("function setupE() {")
Outln("  selInit('selectE','EDelete','EEdit','ENotes','ESort');")
Outln("}")

Outln("function setupC() {")
Outln("  selInit('selectC','CDelete','CEdit','CNotes','CSort');")
Outln("}")

Outln("</script>")

// Continue
Outln("<div id=""choices"" class=""a""><table><tr><td>")
Outln("You are considering <b>"+str(ne)+"</b> "+ten+" and <b>"+str(nc)+"</b> "+tcn+" in the")
Outln("context of<br><b>"+gGetW("Context")+"</b>")
Outln("<br>You can choose from the options listed below, have the system choose, or request")
Outln("other choices")
Out("<td class=""butw"">")
ScriptCall("$OutSubmit","Continue","Choose for me","large")
ScriptCall("$OutSubmit","Other","Other choices","large")
Out("<td class=""qw"">")
ScriptCall("$WGHelp","hMainChoose")
Outln("</table></div>")

// Add elements if less than 6
if ne<6 then
    Outln("<div id=""adde"" class=""a""><table><tr><td>")
    Outln("You should add additional "+tes+" if possible")
    Out("<td class=""butw"">")
    ScriptCall("$OutSubmit","EAdd","Add "+tes,"large")
    Out("<td class=""qw"">")
    ScriptCall("$WGHelp","hMainMoreE")
    Outln("</table></div>")
end if

// Rate unspecified values
n=gGetI("Open")
if n>0 then

```

```

Outln("<div id=\"unspecified\" class=\"a\"><table><tr><td>")
if n=1 then
  Outln("There is <b>1</b> unspecified rating")
else
  Outln("There are <b>"+str(n)+"</b> unspecified ratings")
end if
Out("<td class=\"butw\">")
ScriptCall("$OutSubmit","Specify","Specify","large")
Out("<td class=\"qw\">")
ScriptCall("$WGHelp","hMainUnspecified")
Outln("</table></div>")
end if

// Element matches
if cok and nc>3 then
  s=gGet("MatchE","75","0") // not selected or weighted
  if s<>"" then
    Outln("<div id=\"ematch\" class=\"a\"><table><tr><td>")
    Outln("The following "+tes+" are very similar,<br><b>"+gGetW("E",sGet(s,1))+</b> and <b>"+
      gGetW("E",sGet(s,2))+</b><br>Do you want to enter another "+tc+" to distinguish them?")
    Out("<td class=\"butw\">")
    ScriptCall("$OutSubmit","EBreak","Add "+tc,"large")
    Out("<td class=\"qw\">")
    ScriptCall("$WGHelp","hMainBreakE")
    ScriptCall("$OutHidden","BreakE",s)
    Outln("</table></div>")
  end if
end if

// Construct matches
if eok and ne>4 then
  s=gGet("MatchC","75","0") // not selected or weighted
  if s<>"" then
    Outln("<div id=\"cmatch\" class=\"a\"><table><tr><td>")
    Outln("The following "+tcs+" are very similar,<br><b>"+gGetW("C",sGet(s,1))+</b><br><b>"+
      gGetW("C",sGet(s,2))+</b><br>Do you want to enter another "+te+" to distinguish them?")
    Out("<td class=\"butw\">")
    ScriptCall("$OutSubmit","CBreak","Add "+te,"large")
    Out("<td class=\"qw\">")
    ScriptCall("$WGHelp","hMainBreakC")
    ScriptCall("$OutHidden","BreakC",s)
    Outln("</table></div>")
  end if
end if

// Triad
if cok and ne>2 then
  if bTest(gGetI("Control"),4) then s=gGet("SelectE") else s=""
  s=SelectElements(s,3,true)
  Outln("<div id=\"triad\" class=\"a\"><table><tr><td>")
  Outln("Can you think of a "+tc+" that distinguishes between the three "+tes+",<br/>")
  for i=1 to 3
    Out("<b>"+gGetW("E",sGet(s,i))+</b>")
    if i=2 then Out(" and ") else Out(", ")
  next
  Outln("<br/>such that two are alike and differ from the third?")
  Out("<td class=\"butw\">")
  ScriptCall("$OutSubmit","Triad","Add "+tc,"large")
  Out("<td class=\"qw\">")
  ScriptCall("$WGHelp","hMainTriad")
  ScriptCall("$OutHidden","TriadE",s)
  Outln("</table></div>")
end if

// Pair

```

```

if cok and ne>2 then
  if bTest(gGetI("Control"),4) then s=gGet("SelectE") else s=""
  s=SelectElements(s,2,true)
  Outln("<div id=""pair"" class=""a""><table><tr><td>")
  Outln("Can you think of a "+tc+" that distinguishes between the two "+tes+",<br/>")
  for i=1 to 2
    Out("<b>">gGetW("E",sGet(s,i))+</b>")
    if i=2 then Out("?") else Out(" and ")
  next
  Out("<td class=""butw"">")
  ScriptCall("$OutSubmit","Pair","Add "+tc,"large")
  Out("<td class=""qw"">")
  ScriptCall("$WGHelp","hMainPair")
  ScriptCall("$OutHidden","PairE",s)
  Outln("</table></div>")
end if

// Elements
Outln("<div id=""Ex"""+GetVisible(16,t)+><h2><img src=""/WG/disc.gif"" onclick={switchVisible('E',1,16)}> "+tes.titlecase)
ScriptCall("$WGHelp","hMainElements")
Outln("</h2></div>")
Out("<div id=""E"""+t+><p>")
Outln("<img src=""/WG/discdown.gif"" onclick={switchVisible('E',0,16);}>")
Out("You may ")
if eok then Out("add, ")
if ne>0 then Out("delete, edit, or sort ")
Outln(tes)
ScriptCall("$WGHelp","hMainElements")
Outln("</p><table class=""ele""><tr><td>")
if ne>0 then
  Outln("<select name=""Elements"" id=""selectE"" multiple size="" "+str(ne)+" "" onchange=""setupE()"">")
  for j=0 to ne-1
    s=gGetW("E",str(j),"")
    if hGetB("Select") then t="selected=""selected"" " else t=""
    Outln("<option "+t+"value="" "+str(j)+" "">"+s)
  next
  Outln("</select>")
end if
Outln("<td class=""butw"">")
if eok then ScriptCall("$OutSubmit","EAdd","Add")
if ne>0 then
  ScriptCall("$OutSubmit","EDelete","Delete")
  ScriptCall("$OutSubmit","EEdit","Edit")
  ScriptCall("$OutSubmit","ENotes","Edit note")
end if
if ne>1 then
  ScriptCall("$OutSubmit","ESort","Sort")
end if
if ne>0 then
  ScriptCall("$OutButton","Select none","selAll('selectE',false);setupE();")
end if
if false then // advanced condition ***
  ScriptCall("$OutSubmit","ETest","Test case")
end if
Outln("</table>")
Out("<p>Click on "+tes+" to select those to be used (")
ScriptCall("$OutCheckbox","TSelect",BooStr(BitwiseAnd(control,4)>0))
Outln("use them in pairs or triads)</p>")
Outln("</div>")

// Constructs
Outln("<div id=""Cx"""+GetVisible(32,t)+><h2><img src=""/WG/disc.gif"" onclick=""switchVisible('C',1,32)""> "+tcs.titlecase)

```

```

ScriptCall("$WGHelp","hMainConstructs")
Outln("</h2></div>")
Outln("<div id=\"\"C\"\"+t+\"><img src=\"\"/WG/discdown.gif\"\" onclick=\"\"switchVisible('C',0,32);\"\">")
Out(" You may ")
if cok then Out("add, ")
if nc>0 then Out("delete, edit or sort ")
Outln(tcs)
ScriptCall("$WGHelp","hMainConstructs")
Outln("<table class=\"\"con\"\"><tr><td>")
if nc>0 then
    Outln("<select name=\"\"Constructs\"\" id=\"\"selectC\"\" multiple size=\"\""+str(nc)+"\"\" onchange=\"\"
        setupC();\"\">")
    for i=0 to nc-1
        t=gGetW("C",str(i),"")
        if hGetB("Select") then s="selected=\"\"selected\"\" " else s=""
        Outln("<option "+s+"value=\"\""+str(i)+"\"\">"+t)
    next
    Outln("</select>")
end if
Outln("<td class=\"\"butw\"\">")
if cok then ScriptCall("$OutSubmit","CAdd","Add")
if nc>0 then
    ScriptCall("$OutSubmit","CDelete","Delete")
    ScriptCall("$OutSubmit","CEdit","Edit")
    ScriptCall("$OutSubmit","CNotes","Edit note")
end if
if nc>1 then
    ScriptCall("$OutSubmit","CSort","Sort")
end if
if nc>0 then
    ScriptCall("$OutButton","Select none","selAll('selectC',false);setupC();")
end if
Outln("</table>")
Outln("<p>Click on "+tcs+" to select those to be used</p>")
Outln("</div>")

// Analyses
Outln("<div id=\"\"analyses\"\" class=\"\"a\"\">")
com=gGet("Origin")<>""
Out("<p>You can view and interpret your grid content in different ways ")
ScriptCall("$WGHelp","hMainAnalysis")
Outln("</p><p>")
ScriptCall("$OutSubmit","Synopsis","Synopsis","icon syn")
ScriptCall("$OutSubmit","Display","Display","icon dis")
if ne>1 and nc>1 then
    ScriptCall("$OutSubmit","Focus","Cluster","icon foc")
    ScriptCall("$OutSubmit","PrinGrid","Map","icon pri")
    ScriptCall("$OutSubmit","Crossplot","Crossplot","icon cro")
    ScriptCall("$OutSubmit","Match","Match","icon mat")
    if com then ScriptCall("$OutSubmit","Compare","Compare","icon com")
end if
Outln("</p></div>")

// Miscellaneous
Outln("<div id=\"\"misc\"\" class=\"\"a\"\">")
Outln("<p>You may edit your options, save, exchange or cache your grid, send us a comment, finish
    , or adjust help")
ScriptCall("$WGHelp","hMainMisc")
Outln("</p><p>")
ScriptCall("$OutSubmit","Options","Options")
ScriptCall("$OutSubmit","Save","Save")
ScriptCall("$OutSubmit","Support","Support")
ScriptCall("$OutSubmit","Finish","Finish")
if hGetB("Local","System") then ScriptCall("$OutSubmit","ExportGrid","Rep Plus")
if BitwiseAnd(control,1)=0 then s="Help ? off" else s=" Help ? on"

```

```

ScriptCall("$OutSubmit","Help",s)
Outln("</p></div>")

// Finalize
ScriptCall("$WGTail","<script type='text/javascript'>init();</script>"+CRLF)

end sub

sub DoRead()

dim i,n,k As integer, done As Boolean, s As string

if CheckPost("TSelect") and not bTest(gGetI("Control"),16) then n=4
gSet("Control",str(bSet(gGetI("Control"),4,n)))

gSet("SelectE",GetPost("Elements"))
gSet("SelectC",GetPost("Constructs"))

if CheckPost("Continue") then
done=false
do
select case GetRandom(1,6)
case 1
done=ne<6 and eok
if done then ScriptFlow("CRating/Add") // Add elements
case 2
done=gGetI("Open")>0
if done then ScriptFlow("ERating/Unspecified "+gGet("OpenC")) // Rate unspecified
case 3
done=CheckPost("BreakC",s)
if done then ScriptFlow("CRating/Break") // CBreak
case 4
done=CheckPost("BreakE",s) and eok
if done then ScriptFlow("CAdd/Break "+s) // EBreak
case 5
done=CheckPost("PairE",s)
if done then ScriptFlow("CAdd/Pair "+s) // Pair
case 6
done=CheckPost("TriadE",s)
if done then
ScriptFlow("CAdd/Triad "+s) // Triad
else // stop loop if here from converted WG III file
done=true
DoWrite
end if
end select
loop until done
elseif CheckPost("Specify") then
ScriptFlow("ERating/Unspecified "+gGet("OpenC"))
elseif CheckPost("CBreak") then
ScriptFlow("CRating/Break")
elseif CheckPost("EBreak") then
ScriptFlow("CAdd/Break "+GetPost("BreakE"))
elseif CheckPost("Triad") then
ScriptFlow("CAdd/Triad "+GetPost("TriadE"))
elseif CheckPost("Pair") then
ScriptFlow("CAdd/Pair "+GetPost("PairE"))
elseif CheckPost("EDelete") then
ScriptFlow("Delete/E")
elseif CheckPost("EAdd") then
ScriptFlow("CRating/Add")
elseif CheckPost("EEdit") then
ScriptFlow("CRating/Edit "+gGet("SelectE"))
elseif CheckPost("ENotes") then
ScriptFlow("CRating/Notes "+gGet("SelectE"))

```

```

elseif CheckPost("ESort") then
    gSet("MoveE",gGet("SelectE"))
    DoWrite
elseif CheckPost("ETest") then
    DoWrite // ***
elseif CheckPost("CDelete") then
    ScriptFlow("Delete/C")
elseif CheckPost("CAdd") then
    ScriptFlow("CAdd/Add")
elseif CheckPost("CEdit") then
    ScriptFlow("ERating/Edit "+gGet("SelectC"))
elseif CheckPost("CNotes") then
    ScriptFlow("ERating/Notes "+gGet("SelectC"))
elseif CheckPost("CSort") then
    gSet("MoveC",gGet("SelectC"))
    DoWrite
elseif CheckPost("Synopsis") then
    ScriptFlow("Synopsis/")
elseif CheckPost("Display") then
    ScriptFlow("Display/")
elseif CheckPost("Focus") then
    ScriptFlow("Focus/")
elseif CheckPost("PrinGrid") then
    ScriptFlow("PrinGrid/")
elseif CheckPost("Crossplot") then
    ScriptFlow("Crossplot/")
elseif CheckPost("Match") then
    ScriptFlow("Match/")
elseif CheckPost("Compare") then
    ScriptFlow("Compare/")
elseif CheckPost("Induct") then
    DoWrite // ***
elseif CheckPost("Support") then
    ScriptFlow("Support/")
elseif CheckPost("Finish") then
    ScriptFlow("Finish/")
elseif CheckPost("Options") then
    ScriptFlow("Options/Main")
elseif CheckPost("Save") then
    ScriptFlow("Save/")
elseif CheckPost("Help") then
    gSet("Control",str(BitwiseXor(gGetI("Control"),1)))
    DoWrite
else
    ScriptCall("$NullResponse")
end if

end sub

// main program

//SetLimitEC(6,6)
if Scriptstate="*" then
    DoRead
else
    if TryCommand then DoWrite
        //print Join(ScriptFolderSubs("WebGrid"),CR)
    end if

```

## 5 Bibliography

- Gaines, B. R. (1995). Porting interactive applications to the web. In 4th International World Wide Web Conference Tutorial Notes, pages 199--217. O'Reilly, Sebastopol, CA.
- Gaines, B. R. and Shaw, M. L. G. (1996). Webgrid: Knowledge modeling and inference through the World Wide Web. In Gaines, B. and Musen, M., editors, Proceedings of Tenth Knowledge Acquisition Workshop, pages 65--1--65--14.
- Gaines, B. R. and Shaw, M. L. G. (1997). Knowledge acquisition, modeling and inference through the World Wide Web. *International Journal of Human-Computer Studies*, 46(6):729--759.
- Gaines, B. R. and Shaw, M. L. G. (1998). Developing for web integration in Sisyphus-IV: WebGrid-II experience. In Gaines, B. and Musen, M., editors, Proceedings of Eleventh Knowledge Acquisition Workshop.
- Gaines, B. R. and Shaw, M. L. G. (2007). Webgrid evolution through four generations 1994-2007. Technical Report TR-2007-WG,  
<http://pages.cpsc.ucalgary.ca/~gaines/reports/KBS/WGEvolution>.
- Shaw, M. L. G. and Gaines, B. R. (1996). Webgrid: Knowledge elicitation and modeling on the web. In Maurer, H., editor, Proceedings of WebNet96, pages 425--432. Association for the Advancement of Computing in Education, Charlottesville, VA.
- Shaw, M. L. G. and Gaines, B. R. (1998). Webgrid II: Developing hierarchical knowledge structures from flat grids. In Gaines, B. and Musen, M., editors, Proceedings of Eleventh Knowledge Acquisition Workshop.
- Tennison, J., O'Hara, K., and Shadbolt, N. R. (2002). Apecks: using and evaluating a tool for ontology construction with internal and external KA support. *International Journal Human-Computer Studies*, 56:375--422.

**Most of our publications cited in the references are freely available on the web**