Inforum Software Development in 2011

Inforum World Conference – Hazyview, South Africa
August 21, 2011
Ronald Horst [1]

1. Introduction

This document describes recent development of *G7*, Inforum's econometric software package. Accompanying documentation provide additional technical details and demonstrations. This document also summarizes developments in companion software, including *Compare*, Inforum's report-generating program. Finally, this document reviews changes to the Inforum website[2] and file server.[3]

In 2011, Inforum software and IT services again have evolved to provide better tools and resources for economists. Most of these resources freely are available for use by Inforum employees, partners, clients, and others. New features and enhanced stability provide a suite of tools and services that are more powerful and robust while still more accessible to those with limited experience.

2. *G7* Development

The *G7* software package is used to construct and analyze data, estimate econometric equations, and to build large-scale structural econometric models and report the results. We summarize some of the most significant changes, though many other changes were made to polish and improve the program. The *G7* Reference Manual and Help files have been extended and updated to reflect recent changes in the software.

2.1 String Management and Manipulation

In 2010, *G7* gained the ability to create, manage, and employ strings. These strings are assigned a name, and subsequent routines may reference the names to employ the string anywhere within a *G7* script to perform a wide variety of tasks. For example, strings may be employed to execute particular commands, to provide special dates, to build a switch to turn on or off certain portions of code, to construct titles for graphs, and so on.

A set of routines to complement the string routines also was developed. These features include keywords, text functions, and tools to extract and manipulate strings. Most of these routines may be identified easily by their first character: '%'.

[1] Ron Horst, Inforum, University of Maryland, College Park, USA. Ronald.Horst@gmail.com.
[2] www.inforum.umd.edu
[3] sartoris.umd.edu

In 2011, the underlying mechanisms for reading and processing these commands were revised heavily. In particular, the portion of the program that processes the users' script that immediately follows a '%' character was revised to allow improved stability and to make extensions far easier to implement.

Many new keywords and functions are introduced here, and all of those introduced earlier have been revised to employ the new mechanisms. The routines are summarized below, and more details are provided in the Help files and in the Reference Manual. Demonstration routines also provide examples.

We begin with a review of the string routine. Its syntax is very simple. The following command will construct the string.

str <string title> = "<rhs>"
str <string title 1> = <string title 2>
str <string title 1> = <string title 2> + "<text>" [+ …]

Strings are declared using the *str* command. The string definition *rhs* may be specified as text presented within quotation marks, given as the name of a previously-defined string, or defined as a sequence of text and/or strings linked by '+'.

```
EX:          str equipment = "Textile Machinery "
EX: str industry = equipment + " Manufacturing"
```

Strings may be expanded in arbitrary locations by employing the %s() routine. A new feature of this routine allows a substring to be extracted, while the default behavior specifies the full string.

%s(<string name> [, <first>[, <last>]])

To employ a string in an arbitrary location within a script, use the %s() function with the name of the desired string. This function will expand the string, replacing the %s() command with the contents of the specified string. In this way, strings may be employed with a variety of *G7* routines. Optional parameters *first* and *last* indicate the positions of the first and last characters of a desired substring of string. For example, strings can be used to define a graph title.

```
EX: str   industry = "Retail Trade"
          ti  %s(title)
```

Several other commands are provided to manage strings. New or modified features are presented here. See the Reference Manual or Help files for the full listing with additional details.

str clear [<stringname>]

The *str clear* command erases all defined strings from memory. If a string name is specified, then only that string will be removed from the list.

str print

Print each string to the screen.

The following routines may be employed to parse a text file.

str open <filename>

Open a file for parsing with the *string* command.

str close

Close the file that was opened with the *str open* command.

str getline [<string_name>]

Read a line of text from the file opened by *str open*. Store the text as a string named *string_name*. Implicitly store a copy of the string with the name "line"; this string may be referenced by related routines and accessed with the %line keyword, but it does not appear in the list of user-defined strings.

str parse ["<string>"] ["<separators>" ["<string_root>"]]

Split the string named *string* into words separated by any one of the characters listed as separators. If no string name is provided, then the routine acts on the last line read by *str getline*. The default separators list is composed of the space and tab characters, or " \t". The words are stored as a list of strings that may be accessed with the *%w()* function, which has a syntax similar to the *%s()* function described above. If a word *string_root* is provided, then it is used as the root name for these words.

str replace [<string_name>] <"search_text"> <"replacement_text">

This function operates on the string *string_name*, if provided, or the last line read by *str getline*. Any text in this string matched by *search_text* is replaced by *replacement_text*. Matching employs regular expressions as defined for the Perl regular expressions syntax incorporated in the C++ Boost library.

Note that the Visual C++ Redistributable package must be installed in order to use some of the newest features of *G7*. The VC++ installer should be placed at C:\PDG\C++Install

when *G7* is installed. Run the VC++ installer (run as Administrator if using Vista or Windows 7) before attempting to employ the *str parse* or the *str replace* commands.

Several keywords and functions aid in the parsing of text files. Experienced *G7* users are familiar with variables specified as %1, %2, and so on that are employed in add files, fadd files, do loops, and elsewhere. Several related features that have been introduced. The appearance of a percent symbol (%) within a *G7* script always signifies that the following character or word should be interpreted as a variable, a keyword, or a function. In each case, when the variable, keyword, or function is encountered and evaluated, the value of the result effectively will be inserted into the script and subsequently processed by *G7*.

%linelen

Returns the length of the last line read by *str getline*. If *getline* encounters the end of the file, then the value of *%linelen* will be "-1".

%line

The *%line* keyword allows the user to access the string last read by *str getline*. It especially is useful when no string name was provided.

%numwords

The *%numwords* keyword returns the number of words parsed from the string last processed by the *str parse* command. After calling *str getline*, the number of words is 1 until a subsequent command *str parse* is issued.

%w(<index> [,<first>[,<last>]])

The *%w()* function provides access to the list of words created by the *str parse* routine. *index* is an integer that specifies the position of the desired string within the list of words created by *str parse*. The optional *first* and *last* parameters are integers that specify the positions of the first and last characters to define a substring of the word in position *index*.

2.2 Keywords and Scripting Functions

In addition to the keywords and scripting functions listed above, a number of additional features have been added to *G7*. Many are useful for parsing text or in other manipulation of text, and some perform basic numerical work. These keywords and functions begin with the percent character, and they may be employed virtually anywhere within a *G7* script.

Numerical Routines

Several numerical routines have been added. In contrast to the @-functions that are employed with the *f* command and similar routines, these routines do not read or write to a data bank. Instead, they operate on text or text variables specified in a script.

%floor(<number>, <number>)
%ceiling(<number>,<number>)

These function return the value of nearest integer, either rounding down or up, respectively.

%min(<number>,<number>)
%max(<number>,<number>)

This function returns the minimum or maximum values of the two numbers provided.

%mod(<number>,<number>)

The modulus function returns the remainder of dividing its first argument by the second.

Finally, the %{} routine was introduced earlier. Its primary use is to modify variables that are defined within a *G7* script. Such variables are defined as %1, %2…. The %{} operation allows such calculations based on numerical variables and constants that are referenced within the brackets. Addition, subtraction, multiplication, and division of variables and constants are permitted, and sub-calculations may be made by employing parentheses. Numerical keywords also may be employed.

```
EX:     do{ xl write A %{%1 + 5} right X%1 2000 2010 }( 1-10 )
```

Keywords

Several keywords were introduced earlier. The following is a summary of keywords in *G7*. Availability of particular regression statistics depends on the type of regression performed.

- %date Insert the date.
- %time Insert the time.
- %NARGS Recover the number of arguments that currently are in scope.
- %% Treat this as text, evaluated as a single '%'.
- %fdateX Recover an *fdates* setting, where X = {1,2}.
- %gdateX Recover a *gdates* setting, where X = {1,2,3}.
- %tdateX Recover a *tdates* setting, where X = {1,2}.

- %limX Recover a *limits* setting, where X = {1,2,3}.
- %title Recover the current graph title.
- %subtitle Recover the current graph subtitle.
- %vaxtitle Recover the current graph vertical axis title.
- %ncoef Recover the number of coefficients in the last regression.
- %betaX Recover parameters from the last regression, X={1,…,*ncoef*}.
- %mexvalX Recover mexvals from the last regression, X={1,…,*ncoef*}.
- %see Recover the standard error of the estimate of last regression.
- %rsq Recover the r-square from the last regression.
- %rho Recover rho from the last regression.
- %rbarsq Recover the r-bar-square from the last regression.
- %mape Recover mean absolute percentage error from last regression.
- %dw Recover the Durbin-Watson statistic from the last regression.
- %nobs Recover the number of observations in the last regression.
- %obsinregression Recover the number of observations in the last regression.
- %logl Recover the log likelihood statistic from the last regression.

Other Scripting Functions

Several additional scripting functions have been introduced.

%xlcol(<int>)

This function maps positive digits to letters. Typically, <int> will be a variable (e.g. '%1') that is assigned an integer value. For integer arguments greater than 26, the results will follow the convention for identifying Excel spreadsheet columns. The following is the mapping for the positive sequence of integers: A…Z AA … AZ BA … BZ ….

%getval(<expression>, <date>)

The value of expression in period date is retrieved. Any algebraic expression that is legal for the *f* command may be given, or a variable may be named either from the workspace or from the specified bank.

The following functions operate on strings. The strings either may named strings that were defined by the *str* command or they may be provided as text or a variable that is surrounded by quotation marks (e.g. "x2" or "%1").

%strlen(<string>)

Calculate the number of characters in a string.

```
EX:   str   industry = "Retail Trade" # A string of length 12.
      ic %strlen(industry)            # This calculates and prints the number 12.
```

%strcmp(<string 1>,<string 2>)
%strcmpi(<string 1>,<string 2>)

Compare two strings. The *%strcmp()* function is case-sensitive, while the *%strcmpi* function is not. Arguments may be given as character sequences in quotes or by providing the name of a declared string. In most cases, these functions will be used within if-else statements. These functions are very similar to their C/C++ counterparts. If the strings match, then a value of zero (“0”) will be returned. If the strings do not match, then the return value will not be zero.

```
EX:   str a = "Nominal Sales"
      str b = "Real Sales"
      str c = "nominal sales"

      if (%strcmp(a,b)== 0){ text The strings are identical  }
      else {                 text The strings are different }

      if (%strcmpi(a,c)==0){text Strings are identical except for capitalization }
      else {                 text The strings are different }
```

%strncmp(<string1>, <string2>, <N>)
%strncmpi(<string1>, <string2>, <N>)

The *%strncmp* and *%strncmpi* functions compares the first *N* characters of the specified strings. The *%strcmp* function is case-sensitive, while the *%strcmpi* function is case-insensitive. Strings can be named or specified as a character string surrounded by quotes ("…"). These functions are very similar to their C/C++ counterparts. If the strings match, then a value of zero (“0”) will be returned. If the strings do not match, then the return value will not be zero.

```
EX:   str a = "Nominal Sales"
      str b = "Nominal Output"

      if (%strncmp(c,d,9)== 0){  text The first 9 characters are identical }
      else {                     text The strings are different }
```

%strstr(<string1>,<string2>)

Look for the first instance of *string2* within *string1*. If found, then return the portion of *string1* beginning at that point. Otherwise, return an empty string. This is similar to the function in the standard C libraries.

%strnset(<string>, <N>)

Construct a string composed of *N* repetitions of *string*. This function loosely follows the standard C libraries.

%lower(<string>)
%upper(<string>)
%titlecase(<string>)

Returns a copy of *string* after converting it to lowercase letters, capital letters, or title case (where the first letter of each word is capitalized).

%trim(<string>[,<separators>])
%trimleft(<string>[,<separators>])
%trimright(<string>[,<separators>])

Returns a copy of *string* after removing extraneous characters from the beginning, end, or both ends of *string*. The default behavior is to remove whitespace and control characters. Other characters *separators* optionally may be given in the syntax of the *%eliteral()* routine.

%literal("<string>")

The *%literal()* function prevents the characters of *string* from modification by the *G7* parsing engine. This allows character sequences to be passed through that otherwise might prompt an error or other undesirable action.

%eliteral("<string>")

The *%literal()* function is identical to the *%literal()* function except that it escapes the sequences '\0', '\a', '\b', '\t', '\f', '\n', '\r', '\\', converting each to its corresponding ASCII code. For example, '\t' is converted to the ASCII code for the tab character.

For additional examples, please see the *G7* demonstration routines that are provided on the installation CD, that are included with the *G7* installation package provided on the Inforum web site, and that are available for individual download on the web site.

2.3 The *G7* Resector Command

Over the past year, *G7* has gained a family of *resector* commands. The purpose of these tools is to provide a convenient way to aggregate and disaggregate data among various sectoral levels. The routine begins by reading a file that contains concordances between various sectoral aggregation schemes. The column heading on the first line must specify the number of sectors in that aggregation scheme. This number of sectors is also the name that is used to refer to

that column in the *resector* routines. Please see the demo section of the Inforum web site for examples of the *resector* capabilities.

rs create <filename> [<Number of Columns> <Maximum Number of Sectors>]

The create function opens the key file *filename* and reads in up to 10 columns of aggregation information (6 columns is the default). An optional argument can be supplied to limit the reading to more or less columns than the default. In principle, the routines handle aggregation or disaggregation between any two of the schemes read from the file. With six aggregation schemes, this results in 36 possible combinations of aggregation.

The beginning of a sample file is displayed below:

```
575  495  360  432   85  BEA82
  1    1    1    1    1  10100   Dairy farm products
  2    2    2    2    1  10200   poultry and eggs
  3    3    3    3    1  10301   Meat animals
  4    4    3    3    1  10302   Miscellaneous livestock-horses,bees,ho
  5    5    4    4    1  20100   Cotton
  6    6    5    5    1  20201   Food grains: wheat,rye,rice,buckwheat
  7    7    5    6    1  20202   Feed grains: corn,oats,barley,hay,sorg
  8    8    5    6    1  20203   Grass seeds
  9    9    6    7    1  20300   Tobacco
 10   10    7    8    1  20401   Fruits
 11   11    7    8    1  20402   Tree nuts
 12   12    7    9    1  20501   Vegetables
 13   13    7   10    1  20502   Sugar crops
```

rs formagg <Number From> <Number To>]

This is the first function that should be called after creation. It performs the most important initialization tasks. It sets up all of the information that is needed to aggregate from the scheme indicated as *Number From* to the scheme indicated as *Number To*. It sets up concordance lists and "split lists" that will be used by other functions listed below. In some programs, several of these commands might be needed, especially when using some of the “cross-aggregation” techniques described below.

One of the functions of initialization is to set up lists of correspondences between sectors, as well as lists of splits, where one sector from one scheme corresponds to one or more sectors from the other scheme. Since this initialization is time and memory consuming, an explicit function called *formagg* performs this task, and this function is called for only needed aggregation relationships. *formagg* takes as its arguments the maximum sector numbers of the source and destination schemes. Until *formagg* has been called with a certain aggregation pair, no other functions using that pair are allowed.

rs aggvector <Number From> <Number To> <Input Vector> <Output Vector> <Split Vector>]

This function is designed to aggregate or split a vector from one aggregation scheme to a vector of another aggregation scheme. *Number From* should be the number of sectors of

the source vector, and *Number To* should be the number of sectors of the destination vector, as shown in the heading in the key file that was read from the create routine. *Split Vector* must be of the same aggregation level as the destination vector. It is used by the routine when there is a one-to-many relationship going from source to destination sectors. For pure aggregation, the *SplitVector* will not be used and its values may be set to arbitrary levels.

rs ctrlvec <Number From> <Number To> <Detailed Vector> <Aggregate Vector>

This function controls a detailed vector to values of a more aggregate vector.

rs rdctrlvec <Number From> <Number To> <Detailed Vector> <Aggregate Vector>

This function controls elements of a detailed vector to values of a more aggregate vector, using right-direction scaling.

rs crossaggvector <Number From> <Number To> <Number In Between> <Input Vector> <Output Vector> <Split Vector>

Sometimes conversion of a sectoral scheme is more complicated than merely a combination of aggregations and splits. For example, there may exist a many-to-many relationship, where a *SplitVector* of either sectoring level would not provide enough information on how the flows should be allocated. The name "cross-aggregation" indicates the method of translation used by this function, whereby the source vector is split to the level of a more detailed intermediary vector, which then can be aggregated directly to the destination vector. It must be possible to aggregate the intermediate sectoring level both to the source and to the destination levels for this function to work.

In the argument list, *Number From* is the number of sectors of the source vector, *Number To* is the number of sectors of the destination vector, and *Number In Between* is the number of sectors of the intermediary vector. *Input Vector* is the source vector and *Output Vector* is the destination vector. In this function, *Split Vector* is a vector of length *Number In Between* that is used to split the source vector.

rs aggmatrows <Number From> <Number To> <Input Matrix> <Output Matrix> <Split Vector>

rs aggmatrows <Number From> <Number To> <Input Packed Matrix> <Output Matrix> <Split Vector>

This function is similar to *aggvector* but aggregates the *Input Matrix* by row to obtain the *Output Matrix*. The routine also will accept a packed matrix as the input, though the *Output Matrix* must be a full matrix.

rs aggmatrix <Number From> <Number To> <Input Matrix> <Output Matrix>

This function aggregates a matrix both by rows and columns. Both the source and destination matrices must be square.

rs ctrlmatrows <Number From> <Number To> <Input Matrix> <Output Matrix>

rs ctrlmatrows <Number From> <Number To> <Input Packed Matrix> <Output Packed Matrix>

This function controls a more detailed matrix to elements of a less detailed matrix.

rs ctrlmat <Number From> <Number To> <Input Matrix> <Output Matrix>

rs ctrlmat <Number From> <Number To> <Input Packed Matrix> <Output Packed Matrix>

This function controls all cells within a block of the more detailed matrix to a single cell of the less detailed matrix. The function handles all of the different cases: single cell to single cell, row vector to single cell, column vector to single cell, and sub-block to single cell. The function is useful for updating a more detailed matrix such as a benchmark IO table to a more aggregate (and more current) matrix.

rs crossaggmatrows <Number From> <Number To> <Number In Between> <Input Matrix> <Output Matrix> <Split Vector>

This function works much like *crossaggvector* but aggregates rows of a matrix.

2.4 Alternative Methods for Running InterDyme Models

Clopper Almon has developed an alternative means of running *InterDyme* models from within *G7*. This work will be presented by Dr. Almon at the conference. Models may be run as before by clicking the *Model* menu item in the main window and then selecting *Run Dyme Model*. Those familiar with past versions of *G7* quickly will notice that the graphical interface is different. It still is possible to employ the previous interface. This can be done by clicking the *Edit* menu and selecting *Options*. Select the *InterDyme Options* tab and select *Inforum Style*. With this option selected, the *Run Dyme Model* menu item once again will display the graphical interface and tools of recent years.

2.5 Other *G7* Developments

Several additional changes are listed here.

Command Cache

The *G7* main menu now provides control over the command-line cache, where each command-box entry is recorded. Available menu items allow printing of the cached commands, clearing of the cache, and execution of the cached commands.

function list

This command will print a list of names of the user-defined functions. Such functions are created with the *function* command.

function print [<function_name>]

This routine will print the specification of the function *function_name*. If no name is provided, then the specification of every function will be printed.

The *G7* XL routines for reading, modifying, and creating spreadsheets employs an interface with the Microsoft Excel software. The *Compare* document creating software uses a similar interface. The software has been tested successfully with Office 2010. Though more testing is necessary, no problems were evident in the initial trials.

A number of other changes and improvements also have been made. Additional detail may be found in the *G7* Help file and is summarized there in the "New for 2011" page of that document. Advanced users may find the update logs helpful; these are provided on the Downloads page of the web site.

3. Compare

Compare, Inforum's report generating and data publication program, is used to export *G7* and model data to text files, to printer format files, or to spreadsheet files via Excel. Since the last Inforum conference in Hikone, *Compare* has become more stable and gained several new features. Several of these changes are summarized below. Please see the *Compare* reference manual for additional details; it is available on the Inforum web site and is bundled in the software installation package.

Please note that while *Compare* can generate documents of several types without the help of auxiliary software, it requires the installation of Microsoft Excel in order to create spreadsheet documents in Excel (XLS) format. In addition, the formatting and other features that have been designed for creating spreadsheets will work only when the "\xls" command is given within the *Compare* stub file, where the "stub" file contains the *Compare* commands to generate a document.

Compare allows the user to format specific parts of an spreadsheet by inserting commands into a stub file. The font in the header, title, dates, banks, series names, and data may be controlled separately, or a single specification may be given to for all data that subsequently are printed. Controls are available for the typeface, size, color, vertical and horizontal alignment, bold status, italic status, and underline status and type. The various settings that are desired may be given in any order.

The typeface is specified by giving the name surrounded by double quotes; the quotation marks are required for any name with multiple words. A complete listing of Excel font settings are listed in the *Compare* Reference Manual.

For example, recall the following specification that is used control the standard font for text and data that are printed to a spreadsheet.

\font <settings>

This command is used to set the default formatting and will override all other font commands.

In addition to the list of typefaces that were listed in earlier documentation, several new options have been added. These include "times", "timesbold", "timesbolditalic", "timesitalic", "courier", "courierbold", "courierbolditalic", and "courieritalic". Recall that these apply only to the printing of spreadsheet data using the "\xls" feature.

A new feature has been added to provide an alternative means of comparing multiple but similar data sets. Most often, such comparisons are made among several alternative forecasts or model simulations. In previous versions, *Compare* would print one line of data for each alternative data set, where each line would be printed for a particular item before printing data for the next item. Three presentation styles were permitted. First, data for each alternative could be printed in levels. Second, figures for the first data set could be printed in levels, and data for other data sets could be printed as differences from the first. A final option was to print the first series in levels and to print the alternative series in percentage differences from the first.

Though these three alternative presentations usually prove satisfactory, it sometimes is desirable to print only the deviations from a baseline but to avoid printing the baseline levels. *Compare* now has the means to present data in this format

\printbase <true | false>

Set *printbase* to "true" (or "1" or "on" or "yes") to print the baseline levels. Set *printbase* to "false" (or "0" or "off" or "no") to print only deviations from baseline levels. Deviations may be presented in differences or percentage differences, and this is controlled with the *mode* command.

The maximum number of sectors that *Compare* can read from a Vam bank has been increased to 3500. That is, the maximum allowable vector length has been increased.

Additional changes improve reliability and offer more subtle but useful improvement. Some are posted in the *Compare* log file available on the Inforum web site.

4. The Inforum Website

The Inforum web server was updated in 2011 to employ the current release of the OpenSuSe operating system. This update provided current editions of MySQL, PHP, Apache, and other

server software. In addition, the adoption of new tools for administering MySQL servers and databases now makes much easier the correction and extension of the Inforum documentation archives.

Most other work may be described as basic maintenance and modest improvements. Among the most important are the addition of many Inforum documents on the Research page, including the papers from the Hikone conference, and the maintenance and extension of the *EconData* repository.

The *EconData* portion of the site provides a variety of data, primarily for the U.S. These include macroeconomic and industry data, in addition to regional and other data. These data originally are published by U.S. statistical agencies; Inforum republishes the data without modification in *G7* database formats. *EconData* continues to be updated as new data publications are released. In most cases, these banks are updated within a few days of publication. A wide range of annual, quarterly, and monthly data have been compiled in Inforum database formats. Several banks were added in the past year. Among the most important is the Fixed Assets database published by the Bureau of Economic Analysis. This data set includes long time series of investment, depreciation, capital stocks, and prices for equipment and software and structures, both by type and by purchasing industry. Also included is data on government assets and consumer durable goods.

Current news items can be found on the Inforum homepage and on the News page. Typical news items include announcements of the publication of recent work, the appearance of Inforum staff in the media, and the announcement of Inforum conferences. Recent news includes Inforum's sponsorship and participation in the 19th International Input-Output Association conference that was held this summer near the University of Maryland campus.

Most of the software developed and maintained by Inforum is available on the Software section of the web site, along with documentation. A growing list of demonstration routines also is available. Recent additions include demonstrations of string manipulation and the *resector* routines now available in *G7*. Software and documentation for *G7* and *Compare* has been updated.

A page has been created for each of the Inforum partners and for each International Conference. We attempt to maintain these pages with current information, but we are successful only with the help of each team. A page has been created for the 2011 Inforum World Conference, and documents from the conference will be posted.

5. The Inforum File Server "Sartoris"

For many years, Inforum has operated an FTP file server at the sartoris.umd.edu address. The machine was replaced in 2010 following failure of the previous machine. At that time, the popular but inherently risky FTP (File Transfer Protocol) protocol was replaced with the SFTP

(SSH File Transfer Protocol) protocol. SFTP communications are encrypted, thus offering security that is far superior to the unencrypted FTP communications. No major changes have been made in the past year.

6. Conclusion

Since the 2010 Inforum World Conference, *G7* and companion software have been refined and extended. Most work described in this document is available for download on the Inforum web site. Inforum team members already are making progress on new features of *G7*. We look forward to cooperation with Inforum partners to further improve and extend the work.

Additional support is available; please contact me directly or e-mail the Inforum webmaster at Inforum.Webmaster@gmail.com.