Working Note

Guide to Available Mathematical Software Network Server

Ronald F. Boisvert*
National Institute of Standards and Technology

Current revision : December 2, 1993

This is a very brief description of the services supported by the Guide to Available Mathematical Software (GAMS) server under development at NIST.

1 The GAMS Repository Model

A virtual software repository is a collection of modules. A *module* represents a reusable software part, such as a Fortran subroutine, a C procedure, a self-contained program, or a command in a self-contained system. Modules may have alternate *versions*, identical except for the precision with which the computation is done. In addition, modules are characterized by a set of predefined *attributes*, such as precision and language, which can be used to establish user-specified *filters* for module selection. *Packages* are collections of modules that are related in some way; they may solve related problems or be distributed and maintained as a unit.

Associated with each module and package is a set of *components*. These are retrievable objects such as abstracts, documentation, source code, examples, and tests. Component types are not restricted to a predefined set, so that retrievable objects unique to a particular module or package are easily accommodated. Module and package components physically reside at one or more *repositories*. Each repository may have a different access mechanism, such as electronic mail, anonymous FTP, or a repository server.

An important feature of our model is a *taxonomy* of mathematical and statistical problems which is used to classify software modules. The GAMS problem classification system [1]

*Author's address: Computing and Applied Mathematics Laboratory, National Institute of Standards and Technology, Gaithersburg, MD 20899. Email: boisvert@nist.gov

is a 736-node tree-structured taxonomy extending to seven levels. Each child node in the tree denotes a more specific instance of the mathematical problem represented by its parent. In addition, a two-level *index* of related terms is maintained. This index is mapped to the problem taxonomy, allowing one to translate alternate terminology into that used by the taxonomy.

Finally, the GAMS repository model supports a set of *services* which provide access to GAMS. These include operations such as: retrieval of information about module attributes; retrieval of information about problem classes; search for modules by class (subject to module attribute filters); search for modules by package or module name; module and package abstract retrieval; module and package component retrieval; and index search. These services are provided by the GAMS server.

2 Basic Server Operation

The GAMS server is based on the server developed for *xnetlib*. It may be accessed by making a socket connection to port 5555 at the host doc.cam.nist.gov (129.6.80.2). Server requests consist of 3 text strings separated by newlines[1]:

1. *Identifier.* This is usually the email address of the person making the request. It is used to maintain a log of requests.
2. *Service Name.* At present, this is the fixed string `gams-serve`.
3. *GAMS Request.* This is one of the GAMS service requests described in the next section.

The server is stateless. Each request is independent of all other requests, and the server closes the connection as soon as the response is sent. Each server response is simple ASCII text interspersed with newlines. The first line returned by the server takes the form

```
<status> <n> :  <request>
```

where `<status>` is either `OK` or `BAD`, `<n>` is a count of the number of characters to follow (including newlines), and `<request>` is an echo of the server request. If `<status>` is `BAD` then the remaining lines of the response constitute a free-format error message.

Server operation can be illustrated using telnet as follows. Here the client requests information about the GAMS problem class with tag D4a. The lines typed by the user have the character > at the beginning, while those returned by the server have < at the beginning. (These are included for illustration only; they are not part of the actual interaction syntax.)

[1]The Unix newline is a single line-feed character

```
> telnet doc.cam.nist.gov 5555
Trying 129.6.80.2 ...
Connected to doc.
Escape character is '^]'.
> boisvert@nist.gov
> gams-serve
> class d4a
< OK 198 : class d4a
< D4a
< Ordinary matrix eigenvalue problems (Ax = (lambda) * x)
< 7
< D4a1
< Real symmetric
< D4a2
< Real nonsymmetric
< D4a3
< Complex Hermitian
< D4a4
< Complex non-Hermitian
< D4a5
< Tridiagonal
< D4a6
< Banded
< D4a7
< Sparse
< 0
Connection closed by foreign host.
```

3 Summary of GAMS Requests

Each GAMS request takes the form

```
name arguments
```

The request name is given below as a hyphenated word. The word will be recognized regardless of the case of the characters; the hyphens may be omitted or replaced by underscores.

In Table 1 we list the syntax of the GAMS requests. In the format specifications we use notation of the form `<name>` to represent various fields. Table 2 gives some standard fields names used in the specifications. Items enclosed in square brackets are optional (the brackets

Table 1: GAMS server requests.

```
class [<tag>]
class-index [<keyphrase>]
explain-filter <filter> [= <value>]
list-services
get-module-component <component> <mid> [<site>]
get-news-item [<item>]
get-package-component <component> <package> [<site>]
list-classes
list-filters
list-module-components <mid>
list-modules-in-class[+] <tag> [<filter-spec> ...]
list-modules-in-package <package>
list-modules-with-name <module> [<package>]
list-news-items
list-package-components <package>
list-packages
parent-class <tag>
register
```

themselves do not appear in the commands). We next provide detailed information on the GAMS requests and the format of the responses which they obtain.

1. `register`

   This request is a no-op that returns an empty file. It can be used to determine whether the server is alive. Clients are requested to issue this request on startup so that each use of the client can be logged.

2. `list-services`

   Obtains text that describes the server operation.

   Format : free

3. `list-news-items`

   Obtains a list news items. The news items are brief articles that describe the GAMS project. The articles themselves are retrived with the `get-news-item` request. Only one item is guaranteed to be present : `Motd` (message of the day). This will contain a few sentences describing GAMS, as well as timely messages about the status of the system. Clients are strongly urged to show this message to users on startup.

Table 2: Some fields used in server description.

Field	(1)	Description
<alternate-module>	y	The name of a module which is the same as the current one except for precision.
<component>	n	The name of a retrievable object associated with a package or module.
<description>	y	Arbitrary text.
<filter>	y	The name of a filter.
<filter-spec>	y	A encoded filter specification.
<item>	n	The name of a news item.
<keymatch>	y	An entry in the keyword index.
<keyphrase>	y	A keyword or phrase.
<m>	n	An integer.
<mid>	n	The unique identifier for a module.
<module>	y	A module name.
<mtype>	n	Module precision type: P for primary module, A for alternate precision module.
<n>	n	An integer.
<package>	n	A package name.
<remote-id>	y	An identifier for a component at a remote site. (A command that retrieves the component.)
<site>	n	A repository site name.
<tag>	n	A problem classification identifier.
<value>	y	A filter value.

(1) Does this field contain embedded blanks?

Format : `<m>`
`<item>`
`<item>`
`...`

Here `<m>` is the number of item names to follow. Item names have no embedded blanks.

Example : `list-news-items`
Response : `OK 38 : list-news-items`
`5`
`Packages`
`Sites`
`Motd`
`References`
`News`

4. `get-news-item [<item>]`

   Obtains the named news item. Note that `<item>` must be one of the names returned by the `list-news-items` request. The default item is `Motd`, which is always present.

   Format : free

5. `list-classes`

   Obtains the entire classification scheme.

   Format : `<tag>. <description>`
   `...`

   The classification scheme is formatted as an outline, with the number of spaces between tag and description proportional to the tree level of the class. Tags have no embedded spaces and are terminated by a period. Each class occupies exactly one line.

6. `class [<tag>]`

   Obtains information about the class whose tag is given, such as its description and a list of subclass tags with their descriptions. The number of modules classified at this class is also provided. If `<tag>` is omitted, then information about the root node of the classification hierarchy is returned.

Format :

```
<tag>
<long-description>
<m>
<tag-1>
<description-1>
<tag-2>
<description-2>
...
<tag-m>
<description-m>
<n>
```

<m> is the number of subclasses and <n> is the number of modules with the classification <tag>. Note that no module information other than the number of modules is returned.

Example :

```
class J1a
```

Response :

```
OK 103 :  class J1a
J1a
One-dimensional fast Fourier transforms
3
J1a1
Real
J1a2
Complex
J1a3
Sine and cosine transforms
0
```

7. `parent-class <tag>`

   Obtains information about the parent of the class whose tag is given. The response to this request has the same format as the class request.

Format :

```
<tag>
<long-description>
<m>
<tag-1>
<description-1>
<tag-2>
<description-2>
...
```

```
<tag-m>
<description-m>
<n>
```

8. `class-index [<keyphrase>]`

   Obtains a list of classes having to do with the given keyword or phrase.

   Format :
   ```
   <keymatch> <tag>
   <keymatch> <tag>
   ...
   ```

   This response has a special fixed format: `<keymatch>` starts in column 1 and `<tag>` starts in column 55. If `<keymatch>` is null, then `<tag>` represents an alternate class for the `<keymatch>` on the previous line. If `<tag>` is null, then `<keymatch>` provides information not associated with a class.

   Example :
   ```
   class-index laplace
   ```
   Response :
   ```
   OK 349 :  class-index laplace
   Laplace equation          I2b1a1
   Laplace transform         J3
   Partial differential equations  I2
       Laplace equation      I2b1a1
   Transforms                J
       Laplace               J3
   ```

9. `list-packages`

   Obtains a list of packages in the repository.

   Format :
   ```
   <m>
   <package>
   <site>|<site>| ...
   <description>
   ...
   ```

   `<m>` is the number of 3-line package records to follow.

10. `list-package-components <package>`

    Obtains a list of the components available for the named package. Also returned are the names of the repository sites where the components are located and an identifier for the component at that site. The remote identifier is the command used by GAMS to retrieve the component from the repository.

The response is a series of 3-line component records sorted first in order of component name and then in order of site name. Note that the component named Abstract, which is always available, is not returned.

Format :
```
<m>
<component>
<site>
<remote-id>
...
```

<m> is the number of module records to follow.

Example :
```
list-package-components BLAS3
```
Response :
```
OK 123 :  list-package-components BLAS3
3
Documentation
GRANTA
man blas3 | col -b
Documentation
NETLIB
NETLIBget blas/blas3-paper
Documentation
TIBER
man -D blas3
```

11. `get-package-component <component> <package> [<site>]`

Obtains the named component from the given package. A site named may be specified; if it is present then the component will be retrieved from this site.

Format : free

The component named Abstract is available for all packages in GAMS. The Abstract is partitioned into several sections. The first section contains generic, site-independent information about the module. Sections describing the installation of the module at each repository follow this. A marker in the text delineates the beginning of the section describing each of repository sites; it takes the form

```
#<n> <site>
```

where <n> is a sequence number, and the # is in column 1. For example, if a module is available on two sites, TIBER and GRANTA, then its abstract has the general form

```
     ...  site-independent portion of abstract ...
#1 GRANTA
     ...  portion of abstract describing GRANTA ...
#2 TIBER
     ...  portion of abstract describing TIBER ...
```

Although its format is fixed, clients are cautioned not to attempt to use any more information than this in parsing it as its format may change without notice.

12. `list-modules-in-package <package>`

Obtains a list of modules in the given package. The response is a series of 7-line module records sorted in order of module name.

Format :

```
<m>
<mid>
<module>
<mtype>
<alternate-module>|<alternate-module>| ...
<package>
<site>|<site>| ...
<tag>|<tag>| ...
<description>
...
```

`<m>` is the number of 7-line module records to follow. `<alternate-module>` is the name of a module that is the same as this one except for precision. (Note that more than one `<site>` and `<alternate-module>` may appear separated by the vertical bar character.) This relationship by precision can be used to partition modules in a given package into equivalence classes. In each of these classes one module is designated as the primary module and the others are designated as alternates. ¡mtype¿ is either P (for primary) or A (for alternate). This request will return all modules in a given precision equivalence class; exactly one will be designated as primary.

Example :

```
list-modules-in-package y12m
```

Response :

```
OK 341 :  list-modules-in-package y12m
3
11253
Y12MAE
P
Y12MAF
Y12M
NETLIB
```

```
D2a4
Solves a sparse system ...
11254
Y12MAF
A
Y12MAE
Y12M
NETLIB
D2a4
Solves a sparse system ...
11255
Y12MFE
P

Y12M
NETLIB
D2a4
Solves a large sparse system of ...
```

13. `list-modules-with-name <module> [<package>]`

    Obtains a list of modules with the given name. If a package name is specified, then non-null results are returned only if the named module is in the named package. The response is a series of 7-line module records sorted in order of package name.

    Format :

```
<m>
<mid>
<module>
<mtype>
<alternate-module>|<alternate-module>| ...
<package>
<site>|<site>| ...
<tag>|<tag>| ...
<description>
...
```

    The format is the same as the response of the list-modules-in-package request.

    Example :

```
list-modules-with-name sdot
```

    Response :

```
OK 515 :  list-modules-with-name sdot
4
7750
```

```
SDOT
P
DDOT
BLAS1
GRANTA|NETLIB|TIBER
D1a4
Compute dot product of two vectors.
4388
SDOT
P
DDOT
IMSLM
CAMSUN|GRANTA|TIBER
D1a4
Compute the single-precision dot pr ...
6435
SDOT
P
DDOT
PORT
CAMSUN|GRANTA|TIBER
D1a4
Compute single precision dot produ ...
7373
SDOT
P
DDOT
PORT
CAMSUN|GRANTA
D1a4
Compute the inner product of two ve ...
```

14. `list-modules-in-class[+] <tag> [<filter-spec> ...]`

    Obtains a list of modules in the named class subject to the given filters. The response is a series of module records sorted first by package name and then by module name. The format is the same as the response of the list-modules-in-package request, except that additional filter information is returned in the + version.

    Filter specifications take the form `<fid>=<i1>,<i2>,...` where `<fid>` is the identifier of the filter and `<i1>,<i2>,...` are the indices of the selected values. (See `list-filters` below.)

    Format : `<m>`

```
<mid>
<module>
<mtype>
<alternate-module>|<alternate-module>| ...
<package>
<site>|<site>| ...
<tag>|<tag>| ...
<description>
<n>
<filter-1> = <value>|<value>|...
<filter-2> = <value>|<value>|...
...
<filter-n> = <value>|<value>|...
...
```

The lines beginning `<filter> =` and ending with a line containing a period are only returned in the + version of this request. These lines module attributes which can be used as filters.

The + version of this request is primarily for clients who wish to manage filter information themselves, rather than have the server apply the filter with the `<filter-spec>` option. `<n>` gives the number of filter specifications that follow. Clients should not assume that the same set of filters will exist each time the server is used.

Example :

```
list-modules-in-class C10b4 3=1
```

Response :

```
OK 665 :  list-modules-in-class C10b4 3=1
6
7707
CBESI
P
ZBESI
AMOS
NETLIB
C10b4
Sequence of complex Bessel ...
7711
CBESK
P
ZBESK
AMOS
NETLIB
C10b4
Sequence of complex Bessel ...
```

```
7708
ZBESI
A
CBESI
AMOS
NETLIB
C10b4
Sequence of complex Bessel ...
7712
ZBESK
A
CBESK
AMOS
NETLIB
C10b4
Sequence of complex Bessel ...
11212
BESPAK
P

BESPAK
TIBER
C10a4|C10b4
Subroutines for Bessel and modif ...
8682
644
P

TOMS
NETLIB
C10a4|C10b4|C10d
A portable Fortran package for Bess ...
```

The following example shows the output of the + version of this same example.

Example : `list-modules-in-class+ C10b4 3=1`

Response :

```
OK 652 :  list-modules-in-class+ C10b4 3=1
6
7707
CBESI
P
ZBESI
```

```
AMOS
NETLIB
C10b4
Sequence of complex Bessel ...
5
Sites = NETLIB
Packages = AMOS
Access = Free
Precision = Single
Language = Fortran
7711
CBESK
P
ZBESK
AMOS
NETLIB
C10b4
Sequence of complex Bessel ...
5
Sites = NETLIB
Packages = AMOS
Access = Free
Precision = Single
Language = Fortran
7708
ZBESI
A
CBESI
AMOS
NETLIB
C10b4
Sequence of complex Bessel ...
5
Sites = NETLIB
Packages = AMOS
Access = Free
Precision = Double
Language = Fortran
7712
ZBESK
A
CBESK
```

```
AMOS
NETLIB
C10b4
Sequence of complex Bessel ...
5
Sites = NETLIB
Packages = AMOS
Access = Free
Precision = Double
Language = Fortran
11212
BESPAK
P

BESPAK
TIBER
C10a4|C10b4
Subroutines for Bessel and modif ...
5
Sites = TIBER
Packages = BESPAK
Access = Free
Precision = Single
Language = Fortran
8682
644
P

TOMS
NETLIB
C10a4|C10b4|C10d
A portable Fortran package for Bess ...
5
Sites = NETLIB
Packages = TOMS
Access = Free
Precision = Single
Language = Fortran
```

15. `list-module-components <mid>`

Obtains a list of the components available for the module whose module id is given. Also returned are the names of the repository sites where the components are located and an identifier for the component at that site. The remote identifier is the command used by GAMS to retrieve the component from the repository.

The response is a series of 3-line component records sorted first in order of component name and then in order of site name. Note that the component named Abstract, which is always available, is not returned.

Format :
```
<m>
<component>
<site>
<remote-id>
...
```

Example :
```
list-module-components 7707
```
Response :
```
OK 83 :  list-module-components 7707
2
Fullsource
NETLIB
NETLIBget -d amos/cbesi.f
Source
NETLIB
NETLIBget amos/cbesi.f
```

16. `get-module-component <component> <mid> [<site>]`

    Obtains the named component for the module whose id is given. Optionally, the name of a repository site may be specified.

    Format : free

    A component named Abstract is available for all modules in GAMS. For information on its format, see the discussion in `get-package-component`.

17. `list-filters`

    Obtains a list of supported filters. The possible values for each filter are enumerated, one per line. When specifying a filter value on a `list-modules-in-class` request one only specifies the index of the filter value (each filter's values are numbered beginning with 1).

Format :

```
<m>
<filter>
<n>
<value-1>
<value-2>
...
<value-n>
...
```

The following example illustrates this; note that only a subset of the actual GAMS filters are displayed here for brevity.

Example :

```
list-filters
```

Response :

```
OK 651 :  list-filters
5
Sites
4
CAMSUN
GRANTA
NETLIB
TIBER
Packages
66
AMOS
... 64 items omitted ...
Y12M
Access
2
Free
Proprietary
Precision
3
Single
Double
Multiple
Language
7
Fortran
Basic
C
Dataplot
Modula2
```

```
Pascal
SAS
Algol
Data
Assembly
CAL
MATLAB
```

18. `explain-filter <filter> [= <value>]`

    Provides text which explains the named filter. If the optional value field is included then information about the meaning of this value is returned instead. *This request is not implemented yet.*

    Format : free

References

[1] R. F. Boisvert, S. E. Howe, and D. K. Kahaner. The Guide to Available Mathematical Software problem classification system. *Comm. Stat. – Simul. Comp.*, 20(4):811–842, 1991.