EQ Simulator Container File Format

Michael Barall 06/28/2010
Version 0.4

Note: Container format version 0.4 is identical to version 0.3.

1. Purpose and Goals of the Container Format

For earthquake simulators, we need files that can hold different types of data. For example, we want to use meshes which can contain either rectangles or triangles.

The container format provides a standard way for a file to describe its own contents. This makes it possible to put different types of data into a file without the use of *ad hoc* rules, and to add more types of data in the future. The input and output file formats are all special cases of the container format, and we expect that in the future other files can use the container format as well.

Specific goals of the container format are:

- Be very easy to read and write, using either C/C++ or Fortran.
- Be extensible. It is possible to add more data to a file (either additional data fields within a record, or additional kinds of records) without breaking existing code that reads the file.
- Be self-describing. Software can determine what data is in the file, select the portion of interest, and skip over anything not of interest (even if it doesn't understand the uninteresting portion).
- Be sufficiently similar to the simple ASCII tables used in the past, so that simulator codes will not require extensive revision to use these files.

2. Units

All physical quantities appearing in a file are expressed in SI units.

All angles appearing in a file are expressed in decimal degrees.

This simple rule eliminates any confusion about the appropriate units to use. It also eliminates the need to devise formats and computer codes that work with multiple units.

3. Container File Structure

A container file is an ASCII text file.

Each line in the file is one record.

The file is divided into 7 parts as shown below.

	Part	Description
1	Signature	One line that identifies the type of data in the file, and marks the start of the file.
2	Metadata	Multiple lines that contain data about the file, such as title, author, and date.
3	End-of-metadata	One line that marks the end of the metadata and start of the descriptors.
4	Descriptor	Multiple lines that describe the data records and data fields contained in the file.
5	End-of-descriptor	One line that marks the end of the descriptors and start of the data.
6	Data	Data, one record per line.
7	End-of-file	One line that marks the end of the file.

Parts 1 through 5 are collectively called the *file header*.

The specific types of records that may appear in each part of the file are described later.

The file header may appear to be complicated, but it is easy to deal with because the header is fixed for any given file format.

- You can write a file by writing out the fixed header, with your metadata inserted in the appropriate places. Then write your data one record to a line, followed by a final record at the end of the file. The only reason why you would need to change the header is if you are including extra data in your file.

- When you read a file, you can easily skip over the header and get straight to the data records. Then read your data one record at a time, stopping when you encounter the end-of-file record. If you want to be more sophisticated, you can examine the header to find more information about the file. The amount of header information that you use is entirely up to you.

4. Record Structure

Each record occupies one line in the file. Each record must have the following format:

- Character positions 1-3 hold a 3-digit number, which identifies the kind of record.
- Character position 4 must be a blank space.
- Character positions 5 and above hold the content of the record.

The content of the record is not allowed to be blank. The total length of a record is limited to a maximum of 2504 characters (not including any linefeed or carriage return character that marks the end of the line). Therefore, the content of the record can be a maximum of 2500 characters.

For example, a comment record is identified by number 100. So a comment record could be:

```
100 This is a comment
```

For another example, this could be a data record of kind 200 that contains three numeric values:

```
200 1.234E+03 93.82 -0.462
```

5. Control Records

The following table shows the kinds of records that control the overall structure of the file.

Kind	Description
100	Comment record
101	Signature record
102	End-of-metadata record
103	End-of-descriptor record
999	End-of-file record

Each kind of record is described below. For each record, we give the general format, an example, and an explanation.

5.1. Comment Record

```
100 comment-text

100 This is a comment
```

The comment record lets you insert comments into the file. The comment record may appear anywhere in the file, except that it may not appear before the signature record or after the end-of-file record. Note that the *comment-text* cannot be blank.

When reading the file, you should skip over any comment records.

5.2. Signature Record

```
101 signature-keyword spec-level

101 EQSim_Input_1 1
```

The signature record must be the first record in the file. The *signature-keyword* is a word that identifies what type of data is in the file. The documentation for each particular type of file will identify the required keyword. Keywords are case-sensitive.

The keyword can contain letters, digits, and the following special characters: underscore, hyphen, plus, and period. Its maximum length is 100 characters.

The *spec-level* is an integer that gives the specification level, which is the revision level of the specification document. Any time a file format is changed to include additional data, the specification level is increased. (The additional data may take the form of new kinds of data records, or new fields added to existing kinds of data records.)

Code that is designed to read a particular specification level will also be able to read any higher specification level. It will automatically skip over the additional data present at the higher specification level.

When reading the file, we recommend that you check the signature record to make sure the file contains what you are expecting it to contain. The correct way to check is: (a) check the *signature-keyword* to verify it has the value you expect, and (b) check the *spec-level* to verify that it is greater than or equal to the value you expect.

Note: If a file format is ever changed in a way that makes it incompatible with existing software, then the signature keyword must be changed so the new file format cannot be accidentally given to old software.

5.3. End-of-Metadata Record

```
102 End_Metadata
```

```
102 End_Metadata
```

The end-of-metadata record must appear after the last metadata record and before the first descriptor record. It marks the end of the metadata and the beginning of the descriptors. The string “End_Metadata” must appear in the record exactly as shown.

The end-of-metadata record must appear even if there is no metadata.

5.4. End-of-Descriptor Record

```
103 End_Descriptor
```

```
103 End_Descriptor
```

The end-of-descriptor record must appear after the last descriptor record and before the first data record. It marks the end of the file header and the beginning of the data. The string "End_Descriptor" must appear in the record exactly as shown.

5.5. End-of-File Record

```
999 End
```

```
999 End
```

The end-of-file record must be the last record in the file. The string "End" must appear in the record exactly as shown.

When reading the file, the end-of-file record tells you when to stop.

6. Metadata Records

The following table shows the kinds of records that may appear in the metadata part of the file.

Kind	Description
110	Information record
111	Title record
112	Author record
113	Date record

In the metadata part of the file, records do not have to appear in any particular order.

All metadata records are optional. When writing a file, you can omit the metadata part entirely if you want to. But we recommend that you include at least a title record.

Each kind of metadata record is described below. For each record, we give the general format, an example, and an explanation.

6.1. Information Record

```
110 information-text

110 Created by Acme EQ simulator version 0.12.
110 Duration: 10000 years.
```

The information record lets you insert anything you want into the metadata portion of the file. Note that the *information-text* cannot be blank.

6.2. Title Record

```
111 title-text

111 Problem 3, Acme Simulator
```

The title record should contain a concise summary of the file. Note that the *title-text* cannot be blank.

There should be at most one title record in a file.

In principle the *title-text* can be up to 2500 characters long. In practice, code that reads the file may truncate the title to approximately 80 characters, so the most important information should be listed first.

6.3. Author Record

```
112 author-text
```

```
112 Joe Simulator, University of Somewhere
```

The title record should identify the author responsible for the file. Note that the *author-text* cannot be blank.

There can be multiple author records if needed.

6.4. Date Record

```
113 date-text
```

```
113 03/31/2010 22:04:10
```

The date record should identify when the file was created. The *date-text* is text that gives a date and time.

You may select whatever date and time format you wish. The example above shows a recommended format for March 31, 2010, at 10:04:10 PM.

There should be at most one date record in a file.

7. Descriptor Records

The following table shows the kinds of records that may appear in the descriptor part of the file.

Kind	Description
120	Record descriptor record
121	Field descriptor record

The descriptor part of the file describes the data that the file contains.

Each kind of data record that appears in the file **must** have corresponding entries in the descriptor part of the file. For each kind of data record, there is one record descriptor record, followed by one or more field descriptor records.

It is OK to include descriptors for a kind of data record that does not actually appear in the file. This makes it possible to write a fixed header, without knowing in advance which kinds of data records will end up in the file.

Each kind of descriptor record is described below. For each record, we give the general format and an explanation. An example follows at the end of the section.

7.1. Record Descriptor Record

```
120 kind name n_field comment-text
```

This describes a kind of data record. The items appearing in the record are shown in the following table.

	Item	Type	Description
1	kind	integer	An integer in the range 200 to 399 that gives the kind of data record being described.
2	name	character	A character string that gives the name of the data record. The character string must consist of letters, digits, and the special characters underscore, hyphen, plus, and period. Its maximum length is 100 characters.
3	n_field	integer	An integer giving the number of fields in the data record. It must be in the range 1 to 100.

The items must be separated by one or more blank spaces. The *comment-text* is optional, but if included it must be separated from the last item by one or more blank spaces.

7.2. Field Descriptor Record

```
121 index name type comment-text
```

This describes one field within a data record. The items appearing in the record are shown in the following table.

	Item	Type	Description
1	index	integer	An integer in the range 1 to 100 that gives the position of the field within the data record. It must be 1 for the first field descriptor following a record descriptor, and must be incremented by 1 for each succeeding field descriptor for the same kind of record.
2	name	character	A character string that gives the name of the field. The character string must consist of letters, digits, and the special characters underscore, hyphen, plus, and period. Its maximum length is 100 characters.
3	type	integer	An integer giving the type of data in the field. The allowed values are: 1 – Integer. 2 – Real number. 3 – Character string.

The items must be separated by one or more blank spaces. The *comment-text* is optional, but if included it must be separated from the last item by one or more blank spaces.

7.3. Example

```
120 201 station-pga 5  PGA recorded at a station
121 1 id 1             Station identification number
121 2 name 3           Station name
121 3 pga-ns 2         North-south PGA, + = north
121 4 pga-we 2         West-east PGA, + = east
121 5 pga-v 2          Vertical PGA, + = up
```

This example shows descriptors for a hypothetical data record that might be used to hold the peak ground acceleration recorded by a seismometer station.

The record is named “station-pga” and is kind 201. It has 5 data fields. The first field is an integer called “id” that holds the station identification number. The second field is a character string called “name” which holds the station name. The third, fourth, and fifth fields are real numbers called “pga-ns”, “pga-we”, and “pga-v” which hold the peak ground acceleration in three directions.

The following is an example of what a corresponding data record might look like.

```
201 1184 Firehouse27 7.632e-02 -1.021e-01 8.906e-03
```

8. Data Records

The following table shows the kinds of records that may appear in the data part of the file.

Kind	Description
200-299	Standardized data record
300-399	Private data record

The data part of the file contains the data.

Data appears as a series of records. Each record contains a sequence of fields. Each field contains an integer, a real number, or a character string. In addition, a record may optionally contain a comment.

There are two categories of data record. A *standardized data record* is a data record that is defined in some standards document, for example, the document that defines the simulator input file format. Standardized data records are used to transfer data from one program to another program. A *private data record* is a data record that you invent yourself, for your own private purposes.

Each kind of data record that appears in the file **must** have corresponding entries in the descriptor part of the file.

Note that it is not possible for a record to contain a variable number of fields. The number of fields is fixed, at the value given in the descriptor part of the file.

8.1. Standardized Data Record

```
kind field-1 ... field-N comment-text
```

A standardized data record begins with a number in the range 200-299 that gives the kind of record. Then comes a series of fields, which are shown above as “field-1” through “field-N”. Fields must be separated by one or more blank spaces. (Tab characters are not allowed as field separators.) Each field must contain one of the following:

- An integer. It is a decimal integer, optionally preceded by a plus or minus sign. Leading zeros should not be used. Examples: 17, -358, 0, +34.

- A real number. It may be given in three forms: (a) A decimal integer, optionally preceded by a plus or minus sign. Examples: 23, -76. (b) A fixed-point number like sdd.dddd,

where d is a decimal digit and s is an optional plus or minus sign. Examples: -84.8521, 0.0035. (c) A floating-point number like sd.ddddEsdd, where d is a decimal digit and s is an optional plus or minus sign. The exponent separator may be E or e, and the decimal point is optional. Examples: 5.152E+02, -34.895e-06.

- A character string. It must consist of letters, digits, and the following special characters: underscore, hyphen, plus, and period. The maximum permitted length is 100 characters. Notice that character strings cannot be empty, cannot contain blank spaces, and cannot contain any special characters other than the four just mentioned.

The *comment-text* is optional, but if included it must be separated from the last field by one or more blank spaces.

A standardized data record is described by some standard document. The document will give the following information:

- The name of the record.
- The number of fields.
- The name of each field.
- The type of data (integer, real, or character) in each field.

When writing a file, it is acceptable for you to include additional fields, following the fields mandated by the standard document. For example, if your earthquake simulator produces some output that other simulators do not, you could include that extra output as additional fields in the data record.

If you include additional fields, you must list the additional fields in the descriptor part of the file. This ensures that tools will be able to recognize the presence of the additional fields.

If you include additional fields, and the file is read by a program that is not aware of the additional fields, then the additional fields will become part of the *comment-text* for the record. So, the presence of the additional fields causes no harm.

8.2. Private Data Record

```
kind field-1 ... field-N comment-text
```

A private data record has the same format as a standardized data record, except that the kind is in the range 300-399.

A standard document will never define a data record of kind 300-399. So, you are free to invent private data records for your own purposes.

If you write a file that contains private data records, you must describe them in the descriptor part of the file.

When the file is read by a program that does not understand your private data records, the program simply skips over the private data records as if they were comments.

In general, whenever you read a file you should skip over any private data records you find, unless you are re-reading a file that you wrote yourself.