

GammaLib - Feature #633

Implement handling of INDEF etc in GPar

12/14/2012 04:08 PM - Knödlseider Jürgen

Status:	Closed	Start date:	12/14/2012
Priority:	Normal	Due date:	
Assigned To:		% Done:	0%
Category:		Estimated time:	0.00 hour
Target version:	00-07-00		
Description			
The GPar interface should handle the following special parameter values:			
<ul style="list-style-type: none">• INDEF• NONE• INF• INFINITY• NAN			
See the corresponding APE code (ape_utils.c):			
<pre>/* Strictly convert input string to output bool, ignoring leading/trailing white space, and flagging every possible error. */ static int string2bool(const char * input, char * result) { int status = eOK; if (0 != result) { *result = 0; if (0 != input) { /* Truncate white space from input string. */ char * copy = truncate_white_space(input); /* Compare input string (case insensitive) to various boolean expressions. */ if (0 == ci_strcmp(copy, "true") 0 == ci_strcmp(copy, "t") 0 == ci_strcmp(copy, "yes") 0 == ci_strcmp(copy, "y")) { *result = 1; } else if (0 == ci_strcmp(copy, "false") 0 == ci_strcmp(copy, "f") 0 == ci_strcmp(copy, "no") 0 == ci_strcmp(copy, "n")) { *result = 0; } else { /* String does not contain a valid bool expression. */ status = eConversionError; } } /* Clean up. */ free(copy); copy = 0; } else { status = eNullPointer; } } else { status = eNullPointer; } return status; }</pre>			
<pre>/* Strictly convert input string to output double, ignoring leading/trailing white space, and flagging every possible error. */ static int string2double(const char * input, double * result) { int status = eOK; if (0 != result) { *result = 0.; if (0 != input) { char * remainder = 0; /* Skip leading white space. */ while (0 != isspace(*input)) ++input;</pre>			

```

if ('\0' == *input || 0 == ci_strcmp(input, "indef") || 0 == ci_strcmp(input, "none") || 0 == ci_strcmp(input, "undef") ||
    0 == ci_strcmp(input, "undefined")) {
    status = eUndefinedValue;
} else if (0 == ci_strcmp(input, "inf") || 0 == ci_strcmp(input, "infinity") || 0 == ci_strcmp(input, "nan")) {
    status = eNan;
} else {
    errno = 0;
    *result = strtod(input, &remainder);
    if (ERANGE == errno) {
        if (0. == *result) status = eUnderflow;
        else status = eOverflow;
    } else if (0 != remainder && '\0' != *remainder) {
        /* Ignore trailing whitespace; the C standard is not specific about whether it's returned or eaten by the conversion. */
        while (0 != isspace(*remainder)) ++remainder;
        if ('\0' != *remainder) status = eStringRemainder;
        /* else conversion succeeded. */
    } else {
        /* Conversion succeeded! */
    }
}
} else {
    status = eNullPointer;
}
} else {
    status = eNullPointer;
}
}
return status;
}

```

```
static int string2long(const char * input, long * result) {
```

```
if (0 != result) {
```

```
if (0 != input) {
```

```
errno = 0;
```

```
while (0 != isspace(*input)) ++input;
```

```
0 == ci_strcmp(input, "undefined")) {
```

```
} else if (0 == ci_strcmp(input, "inf") || 0 == ci_strcmp(input, "infinity") || 0 == ci_strcmp(input, "nan")) {
```

```
} else {
```

```
if (ERANGE == errno) {
```

```
else if (LONG_MAX == *result) status = eOverflow;
```

```
    } else if (0 != remainder && '\0' != *remainder) {
```

```
while (0 != isspace(*remainder)) ++remainder;
```

```
/* else conversion succeeded. */
```

```
/* Conversion succeeded! */
```

}

```
status = eNullPointer;
```

```
else {
```

}

```
return status;  
}
```

Subtasks:

Action # 636: Implement handling of NaN	Closed
Action # 637: Add parameter check and initialization during GPar constructor	Closed
Action # 634: Make Boolean interface compatible with APE.	Closed
Action # 635: Implement handling of INDEF	Closed

History

#1 - 12/14/2012 04:08 PM - Knödseder Jürgen

- Target version set to 00-07-00

#2 - 12/14/2012 04:17 PM - Knödseder Jürgen

- Status changed from New to In Progress

#3 - 12/14/2012 05:51 PM - Knödseder Jürgen

- Status changed from In Progress to Feedback

#4 - 12/21/2012 12:16 AM - Knödseder Jürgen

- Status changed from Feedback to Closed