# GMAT Software Specification

**This document is now obsolete. GMAT specifications have been broken out by feature and links to each individual spec are located on our wiki here:**

**http://li394-117.members.linode.com:8090/display/GW/Functional+Specifications**

## Table of Contents

|

# Document Overview

## Audience

I have assumed that this document will be used by software engineers to write code, by software testers to test the software, and by technical writers to write end-user documentation. When possible I have attempted to write the specifications so that content can be used with little or no modification in user-documentation and have therefore written the document as if the end-user is the reader.

## Review and Change Control

This is a living document and different parts of this document are at different stages at any given time.  For example, the Spacecraft Orbit State section may be in earroiply draft form, while the Target command is reviewed and ready for testers and documenters.

Each section has a status defined below the section title and the status definitions are as follows:

**Draft**:  First phase of the documentation that fills in what content will be covered, often in bulleted list or sentence fragment form.
**Review**:  Complete draft that is currently being reviewed by developers and testers.
**Complete**:  Review is complete and document is ready to spec test cases and write user docs.
**Modified:**  The document has been modified and re-reviewed and updates/changes are ready to be moved to end-user documentation.  These comments are can be left out or blank in most cases.  The section is only required when making a change to a section that was marked as complete.

Process for making changes to sections that are in **Ready for Test/Doc status.**  This process ensures that changes made to the spec after initial review end up in the user documentation when necessary.  After changes are reviewed, the section is marked as modified, additions are marked with green background like this.  An asterisk is put in the section title so that the document table of contents reflects the fact that there are modifications in the section.  After, changes are made in dependent docs such as the user's guide, the Technical Writer is resposnsible for change the section status back to ReadyForTest/Doc, changing next background from green to white, and removing the asterisk from the section header.

**Only the Spec Owner (currently the PDL) can change sections marked as Complete.**
**If a section is marked as Modified, only the Tech Writer can switch state and then only to Complete.**

# Spec Template and Writers Guide

This section describes how to write a new spec section for a Resource or Command.  The outline contains the required sections with comments describing the content for each section and, in limited cases, describes when a section is not necessary. In general,  there are several spec consumers: programmers responsible for implementing features, testers who will test the feature, and tech writers who write the end user documentation.

**General spec writing guidelines**

- Follow all rules in the GMAT Style Guide.
- Use clear, simple, active voice
- Assume tech-writer/end-user are the primary audience for all material not contained in

message boxes (see below). i.e. (write with end-user quality material in main sections, use whatever is necessary to convey the point in message boxes).

- Read existing specs for example of style before writing
  - See Force-Model for a rigorous Resource Example
  - See BeginFiniteBurn for a rigorous Command example
- If you are a feature lead drafting a spec chapter, emphasize completeness of content. The style does not have to be perfect, it will be reworked by the doc owner.
- If you are the PDL, focus on clarity and consistency of the text.
- Here are special annotation styles

| |
|---|
| Open Issue: Use a red table box like this to point out issues in behavior or functionality that are not resolved. THESE ARE NOT BUGS!! Must be resolved before spec is finished. |

| |
|---|
| Caution: Use a red table box to include potentially confusing or very important information for users. Examples include when a feature only works in the script and not in the GUI, or if a feature can potentially be misused if the user does not understand something critical. |

| |
|---|
| For GUI Tester: Put information intended for a specific document user in a separate grey table box. For example, if a feature may required a unique GUI test type, let the GUI tester know by including the information in a box like this. |

Put script snippets in a grey table box and use monospace font.

```
BeginFiniteBurn aFiniteBurn(aSat)
BeginFiniteBurn aFiniteBurn(aSat)
BeginFiniteBurn aFiniteBurn(aSat)
```

**Chapter Outline**

Status: Draft, Under Review, Complete, or Modified
Owner: Initials
Changes: See section called “Review and Change Control” for this field

## Overview

A one sentence description of the object, without a trailing period

## Script Syntax

Code snippets describing general usage of the object.

THIS SECTION IS ONLY NEEDED FOR COMMANDS.

- Required literal text is **bold**.
- Optional elements are surrounded by [square brackets].
- Placeholders are *italic*.
- Placeholders for user values are called *value*.

## Description

Describe the object at a high level to give the user an intuitive understanding of what the object does. If there are complex interactions discuss briefly (See ForceModel for example) and point to Remarks sections for further clarification.

*See Also*: Put list of resources and commands that interact with the object here.

## Fields

Include reference to field spec. This section is usually empty other than the hyperlink.

## GUI

Include GUI screenshot

Describe behavior of GUI:

- If there are fields that can be inactive or "invisible" describe here.
- If fields have complex interactions describe GUI behavior here.

## Remarks

Describe gory details of the resource.

- Complex field interactions
- Items relevant to user but not described in the Fields section above.

## Examples

Include script examples for primary usages here. Use syntax variations between examples if possible.

Open Issue: This is a real open issue and not a template example! The outline currently doesn't handle unique error messages for the object. Generic error messages are described in a separate chapter. However, many objects often have unique modes of operation that are not allowed and they required unique error messages. The spec doesn't support this yet.

# Inspection Guidelines

## Field Spec

Status: Draft
Owner: SPH

**Feature Owner**

- Are all fields populated with complete information?
- Are all fields populated with correct information?
- Is the format consistent and correct for automated processes?
- Are the resource names and field names spelled correctly?
- Are the field descriptions concise and consistent?

**Developer**

- Are all fields identified? (This requires reading the code)
- Are the field data types correctly identified? (This requires reading the code)
- Are the field couplings correctly identified?
- For enumerated fields, are all available options listed? (This requires reading the code)
- Are the field descriptions accurate?

**GUI Tester**

- Are all items in the GUI identified in the spec.
- Are items not available in the GUI identified?
- Can you write and implement test procedures from the spec?

**All**

- Is the spec complete?
- Is the spec of acceptable quality?
- Is this checklist complete, concise, and useful?

## Feature Spec

Status: Draft
Owner: SPH

**Feature Owner**

- Does the spec contain the required minimum sections?
- Is the object overview in the correct format?

**Developers**

- Is the text accurate?
- Is the spec complete?
    - Are all object couplings identified?
    - Are all command couplings identified?
    - Are the object couplings explained in clear examples?
    - Are error messages explained?

**Testers**

- Is the text specific and clear enough to test from?

**Engineers**

- Is the text accurate?
-

**All**

- Are there any unresolved issues that aren't captured specifically in an "Open Issue" box?
- Is the text written clearly and consistently?
- Is the text written at the level of the user?
- Is the text of finished quality?
- Are basic usages explained in clear examples?
- Are Caution statements used to clarify potentially confusing or soon-to-change behavior?
- Is this checklist complete, concise, and useful?

## Test Procedures

Status: Draft
Owner: SPH

**All**

- Is all core functionality tested?
- Are all input fields tested with non-default values?
- Are all output fields tested with non-default values?
- Are all feature-specific error conditions tested in validation mode?
- Are the individual test summaries specific enough that you could easily tell if they've been satisfied?
- Are all accepted values tested for enumerated fields?
- Is functionality at the limits tested for numerical fields?
- Is this checklist complete, concise, and useful?

## Requirements Spec

Status: Draft
Owner: SPH

**Engineers**

- Are the requirements complete (no missing requirements)?

**Developers**

- Are all requirements specific and unambiguous enough to implement?
- Are all statements worded as actual requirements, not implementation or design details?

**Testers**

- Are all requirements testable?

**All**

- Are the requirements worded clearly and concisely?
- Do requirements conform to the overall style of the document?
- Are the requirements written well (no typos, grammar errors, etc.)?
- Is this checklist complete, concise, and useful?

## Definitions

Active Script: The active script file is the script associated with the mission loaded in the GUI. If you have two scripts A and B open in different script editors, and script A is loaded in the GUI, then script A is the active script.

# General Specifications

## Error Message and Warning Messages

## Philosophy

- Error/Warning messages are for users and should alert the user that there is an mistake in their setup (error) or a potential issue (warning)
- Messages should provide detail of about the error or warning usings user interface syntax/language and not internal code syntax/language.
- Error/Messages and warnings should include the snippets referring to user-names of objects when possible. For example: “`The value of “0" for field "SMA" on Spacecraft “mySat” is not an allowed value”`

## Random comments

- deprecation messages should probably not say this: “Feature X” will be removed from a

future build.  Should say “removed from future version”.  Builds are programmer speak. Version is user speak.
- Avoid including module or method names in part of message relevant to user.  Keep internal module/method names isolated to a particular part of the message reserved for that type of info.
- use object/command type names and when possible use the name defined by user.
- Use the word Error instead of Exception?
- Include invalid numeric values and allowed range in message.

Practical considerations given GMAT’s design.

It appears that validation error messages may have to work differently for issues in initialization (assignment) vs. Mission Sequence (command).  It may be hard to get the line number into error messages from initialization (could be wrong).

In general Provide:
- Line number,
- Exact script line
- Error message

LOJ commented that it would be nice to have extra information in the error messages to aid developers if startup file has test mode = true.

- Need to decide how numeric values should be included (i.e to what precision etc.)
- Need to determine if carriage return should ever be included in messages.  Some have them, some don’t
- Punctuation?

## Undefined Variable

**When Validated**: Build Time or GUI Apply/Ok.
**Message**: `The function or variable "`*`Name`*`" for field "`*`FieldName`*`" does not exist in line: “Insert Script Here”.`

`Open Issue: Here are examples of current error messages.`

**Undefined Variable for While Command**
`The variable "var" for field "LHS" does not exist.`

```
Please create it first from the ParameterSelectDialog or from the
Resource Tree.
```

**Undefined Variable for Propagate Command**

```
The variable "var" for field "Parameter" does not exist.
Please create it first from the ParameterSelectDialog or from the
Resource Tree.
```

## Invalid Data Type

### When Validated:

**Message:** `The value of "arr" for field "LHS" is not an allowed data type. The allowed data types are: [`*`List of Data Types`*`].`

```
Open Issue: Here are examples of current error messages. The
difference is small value -> data type.
```

**Invalid Data Type for While Command**

```
The value of "arr" for field "LHS" is not an allowed value.
The allowed values are: [Variable, Array element, plottable
Parameter].
```

**Invalid Data Type for For Command**

```
Interpreter Exception: Command Exception: The value of
"DefaultSC.SMA" for field "Index Name" on command "For" is not an
allowed value. The allowed values are: [ Variable].
```

```
Open Issue: Some commands allow fields like Sat.X as data members
when the parameter evaluates to a real scalar.  However, not all
object parameters that evaluate to scalar real numbers are
supported.  We need to determine (1) how to exlain to users what
the supported options are, are these Real Parameters, Object
Parameters and (2) how do we modify GMAT so that simple object
parameters, like myDifferentialCorrector.MaximumIterations are
supported just like Sat.X is.
```

### Array Index Out of Bounds

**When Validated**: Run Time
**Message:** `Index exceeds matrix dimensions in line: “Insert Script Here”.`

```
Open Issue: Here are examples of current messages.  Some are close.
Some do now show the actual line that contains the error.  This
test occurs at run-time.
```
**Example: Array Index Out Of Bounds in For Command**
```
Cannot return Real value for array arrw nd column  - exception
thrown: TableTemplate error : index out-of-bounds.
```

**Example: Array Index Out of Bounds in Propagate Command**
```
Command Exception: Cannot return Real value for array arrw d column
- exception thrown: TableTemplate error : index out-of-bounds.
 in Propagate DefaultProp(DefaultSC) {DefaultSC.ElapsedSecs =
arr(9,4)};
```

**Example: Array Index Out of Bounds in Assignment Command**
```
Cannot return Real value for array arrw d column  - exception
thrown: TableTemplate error : index out-of-bounds.
 in
   "GMAT I = arr(9,4)"
```

## Common Resource Object Dialog Boxes

## Common Command Dialog Boxes Elements

# Application Control

## GMAT Startup File

The GMAT startup file is a file used to configure GMAT and is read when the GMAT application is opened. The file is named `gmat_startup_file.txt` and is located in the bin directory of your installation.

You can generate a list of all parameters supported by GMAT by including the following line in the startup file. The list will be printed to the log file.

```
DEBUG_PARAMETERS = ON
```

## Toolbar

### GUI/Script Sync Status box

GMAT allows you to simultaneously edit the Script and the GUI versions of your mission. This allows you to rapidly change editing environments so that you can work in the environment most efficient for current tasks. As a result, the Script and GUI can contain different and at times un-synchronized data. The GUI/Script Sync Status box informs you when the GUI and script have different and possibly un-synchronized data in the GUI and Script versions of your mission. There are four states that can occur and the meaning for each state of the GUI/Script Sync Status field is shown below.

**GUI Modified**: There are unsaved changes in the GUI but there are no changes in the script.

**Script Modified**: There are unsaved changes in the Script but there are no changes in the GUI.

**Synchronized**: There are no unsaved changes in the either the script or GUI and the GUI and Script contain the same mission data.

**UnSynchronized**: There are unsaved changes in the script and GUI and the GUI and Script contain different and possibly inconsistent mission data.

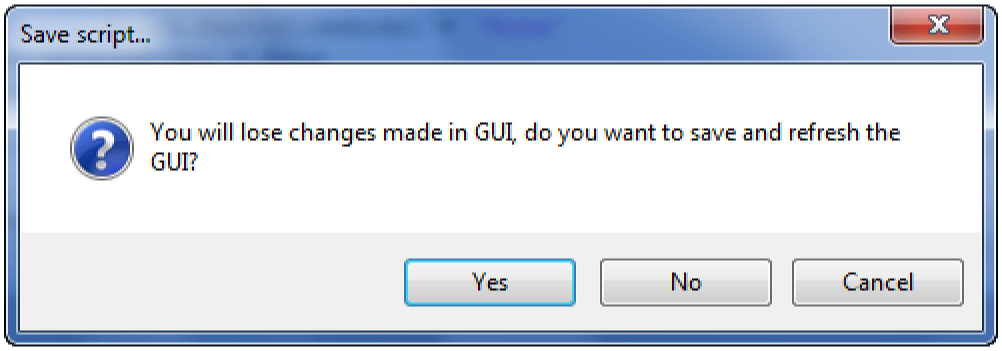
### Save button

The **Save** button allows you to save the mission in the GUI to the disk (in the form of a script file) and if the script is open in a script editor window, to synchronize the script with the GUI. The **Save** button is always active. However, its behavior changes depending upon whether or not the Active Script is loaded in an open script editor and if there are unsaved changes in the script.  Below we describe the behavior for each special case and deviations from the nominal behavior.  The nominal behavior when you click the **Save** button is:

1. Save the last valid mission to the script file.  If a dialog box is open and it contains invalid data, the data from either the time the dialog box was opened, or the last successful validation (OK/Apply) is saved to the file.
2. Refresh the Script Editor loaded with the Active Script if it is open in a Script Editor window.
3. Set **GUI/Script Sync Status** to **Synchronized**.

*Behavior by value of* **GUI/Script Sync Status:**

- **GUI/Script Sync Status** is **Synchronized**: Perform nominal **Save** behavior.
- **GUI/Script Sync Status** is **GUI Modified**: Perform nominal **Save** behavior.
- **GUI/Script Sync Status** is **Script Modified**: Alert the user that they will lose changes in the Script by displaying the dialog box below.  If the user clicks no, do nothing. If the user clicks yes, perform nominal **Save** procedures.
- **GUI/Script Sync Status** is **UnSynchronized**: Alert the user that they will lose changes in the Script by displaying the dialog box below.  If the user clicks no, do nothing. If the user clicks yes, perform nominal **Save** procedures.



You will lose changes made in GUI, do you want to save and refresh the GUI?

Open Issue:

- The error message on the dialog box above should read: “You will lose changes made in the Script if the GUI is saved. Do you want to save the GUI and discard old script changes?”
- There should be no cancel button on the panel.

## Mission Tree

Status: Draft

Owner: TGG

## Parameter Select Dialog Box

```
Open Issue:  The Parameter select dialog box is referenced from the
BeginFiniteBurn command.  This behavior of the box is to copy
objects from left to right, which means an object can appear in
both lists.  DJC has suggested we change the behavior so that
objects are moved from left to right so that an object appears in
only one list.
```

LHS Assignment Mode

- Does not support selection of parameter dependency on central body or coordinate system.
- The following object types are avialable:
  - Implusive Burn
  - Spacecraft
  - Array
  - Variable
- If called from Vary command, parameters must be
  - scalar
  - real number
  - read-write (read only are not available)

Variable Only Mode:
Can only select variable.  This is called from the Index field on the For command.

## Script Syntax and Behavior

### Object Property (Field) Data Types

## Boolean Type

Open Issue:  Should we accept Yes/No or On/Off?
Resolution:  Accept case insensitive On/Off.  Do not accept Yes/No.

### Description

Boolean type can take one of two values: true or false.

### Example

```
MyXYPlot.ShowPlot = true;
DC1.ShowProgress = false;
```

### Accepted RHS Inputs

- true or false without single quotes (case-insensitive)

### Behavior on Output

- true or false without single quotes

## Boolean Array Type

Open Issue:  Should we accept array of Yes/No or On/Off?
Resolution:  Accept case insensitive On/Off.  Do not accept Yes/No.

### Description

Array of true or false Boolean type with square brackets [ ].

### Example

```
MyOrbitView.DrawObject = [ true true ];
```

### Accepted RHS Inputs

- Array of true or false (case-insensitive) with [ ]

**Behavior on Output**

- Array of true or false with  [ ]

## Integer Type

Open Issue:  Should we accept whole float point number without fraction?

**Description**

Integer type represents integer number between –2,147,483,648 and 2,147,483,647

**Example**

```
MyForceModel.GravityField.Earth.Degree = 4;
MyPropagator.InitialStepSize = 60;
```

**Accepted RHS Inputs**

- Integer number

**Behavior on Output**

- Integer number

## Integer Array Type

**Description**

Array of Integer type with square brackets [ ].

**Example**

```
MyOrbitView.OrbitColor = [ 255 32768 ];
```

**Accepted RHS Inputs**

- Array of integer number with [ ]

**Behavior on Output**

- Array of integer number with [ ]

## OnOff Type

Open Issue: Should we change it to Boolean type? It is used in ReportFile and OrbitView.
Resolution: Accept case insensitive true/false. Do not accept Yes/No.

**Description**

OnOff type can take one of two values: On or Off (case-sensitive).

**Example**

```
MyOrbitView.Axes = On;
MyOrbitView.Grid = Off;
MyReport.WriteHeaders = On;
MyReport.LeftJustify = On;
```

**Accepted RHS Inputs**

- On or Off without single quotes (case-sensitive)

**Behavior on Output**

- On or Off without single quotes

## Real Type

**Description**

Real type represents double precision floating point number between 1.7e - 308 and 1.7e+308 (15 digits).

**Example**

```
Sat1.SMA = 7191.938817629013;
MyPropagator.Accuracy = 9.999999999999999e-012;
```

#### Accepted RHS Inputs

- Floating point number

#### Behavior on Output

- Floating point number

### Time Type

On Input:

Open Issue: The first bullet may need modification: GMAT currently accepts Gregorian dates without quotes, at least in assignment mode.
SPH Comment: We probably need to accept Gregorian Dates without quotes for backwards compatibility.

- Single quotes are required if not numeric.
- For numeric, accept with or without single quotes

On Output

- If is not numeric, output with single quotes
- If is numeric, do not output with quotes.

This behavior is the same in all modes: assignment, command, function.

### Enumeration Type

#### Description

An enumeration type is a predefined list string literal.

#### Example

```
MySpacecraft.DisplayStateType = ‘Cartesian’
DC1.DerivativeMethod = ForwardDifference;
```

#### Accepted RHS Inputs

- Literal with single quotes
- Literal without single quotes
- User defined string type in Command mode

**Behavior on Output**

- Literal without single quotes

## Filename Type

Open Issue:  Not all the code using Filename type for filename input. Some code using String type since Filename type was added later.

**Description**

A Filename  type represents a name of a file with or without path.

**Example**

```
MyForceModel.GravityField.Earth.PotentialFile = 'JGM2.cof';
Sat1.ModelFile = 'C:/Gmat/application/data/vehicle/models/aura.3ds';
SolarSystem.DEFilename = '../data/planetary_ephem/de/leDE1941.405';
```

On Input:
- Backslashes or forward slashes are accepted as input
- Absolute path must be resolved by operating system
- Case sensitivity of file names is determined by OS

On Output
- Output the original user input

Relative paths are with respect to several places
- Specified in startup file

Relative paths
Re-initialization when file field is reset.

## String Type

**Description**

A String data type represents a string of text.

**Example**

```
MySat.Id = 'SatId';
MySat.EulerAngleSequence = '321';
MyForceModel.GravityField.Earth.EarthTideModel = 'None';
```

**Accepted RHS Inputs**

- Literal with single quotes
- Literal without single quotes
- User defined string type in Command mode

**Behavior on Output**

- Literal with single quotes

## Object Type

**Description**

Object name of predefined object type.

**Example**

```
MyPropagator.CentralBody = Earth;
MyPropagator.Type = RungeKutta89;
```

**Accepted RHS Inputs**

- Object name with single quotes
- Object name without single quotes
- User defined string type in Command mode

**Behavior on Output**

- Object name without single quotes

## Object Array Type

Open Issue:  I tired with ForceModel, Formation, subscribers and all behaves differently on inputs. See current behavior below.

### Description

List of object names of predefined object type inside curly brackets { } separated by comma or space.

### Example

```
MyForceModel.PrimaryBodies = {Earth};
MyForceModel.PointMasses = {Luna, Mars};
MyReport.Add = {Sat1.X, Sat1.Y, Sat1.Z};
```

### Accepted RHS Inputs

- Object name without single quotes inside { }
- Object name with single quotes inside { } (ForceModel - No)
- User defined String type in Command mode (No)
- Empty curly brackets

Open Issue:  Behavior is not defined list with empty brackets.

### Current Behavior on Inputs

### 1) Object name(s) with single quotes in Object Mode:

```
MyForceModel.PrimaryBodies = {‘Earth’};
```
**** WARNING ****: Utility Exception: 'EARTH'_POT_PATH not in the gmat_startup_file

```
MyForceModel.PointMasses = {‘Luna’, ‘Mars’};
```
1: **** ERROR **** Interpreter Exception: Nonexistent SpacePoint "'Luna'" referenced in the ForceModel "MyForceModel"
2: **** ERROR **** Interpreter Exception: Nonexistent SpacePoint "'Mars'" referenced in the ForceModel "MyForceModel"

```
MyFormation.Add = {'MMS1', 'MMS2'};
```
Parses and runs OK

```
MyXYPlot.YVariables = {'MySat.X', 'MySat.Y'};
```
Parses and runs OK

```
MyOrbitView.Add = {'MySat', 'Earth'};
```
Parses and runs OK

### 2) Single object name without {} in Object Mode:

```
MyForceModel.PrimaryBodies = Earth;
```
1: **** ERROR **** Utility Exception: "Earth" is not enclosed with "{}" in line:

```
MyFormation.Add = MMS1;
MyXYPlot.YVariables = MySat.X;
MyOrbitView.Add = MySat;
```
Parses and runs OK.

### 3) Multiple object names without {} in Object Mode:

```
MyForceModel.PointMasses = Luna, Mars;
```
1: **** ERROR **** Utility Exception: "Luna, Mars" is not enclosed with "{}" in line:

```
MyFormation.Add = MMS1, MMS2;
```
1: **** ERROR **** Interpreter Exception: Nonexistent SpacePoint "MMS1, MMS2" referenced in "MyFormation"

```
MyXYPlot.YVariables = MySat.X MySat.Y;
```
Parses OK but it adds “MySat.X MySat.Y” to the list, so getting run-time message:: *** WARNING *** The XYPlot named "MyXYPlot" will not be shown. The first parameter selected for X Axis or Y Axis is NULL

```
MyOrbitView.Add = MySat, Earth;
```
1: **** ERROR **** Interpreter Exception: Nonexistent SpacePoint "MySat, Earth" referenced in the OrbitView "MyOrbitView"

### 4) Object name(s) with single quotes in Command Mode:

```
MyForceModel.PointMasses = {‘Venus’, ‘Jupiter’}
```
Parses and adds to original MyForceModel, so when Propagator panel is opened, it shows there.

```
MyFormation.Add = {'Sat3'}
MyFormation.Add = {'MMS1', 'MMS2', 'Sat3'}
```
Parses and runs OK.

```
MyXYPlot.YVariables = {‘MySat.VX’, ‘MySat.VY’};
MyOrbitView.Add = {'Luna', 'Sat2'};
```
Parses and runs without adding new objects.

### 5) Single object name without {} in Command Mode:

```
MyForceModel.PointMasses = Venus;
```
1: **** ERROR **** Utility Exception: "Venus" is not enclosed with "{}" in line:

```
MyFormation.Add = MMS1;
```
Parses and runs OK.

```
MyXYPlot.YVariables = MySat.X;
MyOrbitView.Add = MySat;
```
Parses and runs without adding new objects.

### 6) Multiple object names without {} in Command Mode:

```
MyForceModel.PointMasses = Venus, Jupiter;
```
1: **** ERROR **** Utility Exception: "Venus, Jupiter" is not enclosed with "{}" in line:

```
MyFormation.Add = 'MMS1', 'MMS2', 'Sat3';
MyFormation.Add = MMS1, MMS2, Sat3;
Parses and runs OK.
```

```
MyXYPlot.YVariables = MySat.VX MySat.VY;
MyXYPlot.YVariables = ‘MySat.VX’ ‘MySat.VY’;
MyOrbitView.Add = Luna,Sat2;
MyOrbitView.Add = 'Luna', 'Sat2'
```
Parses and runs without adding new objects.

### 7) Setting as String type in Command Mode:

```
MyForceModel.PointMasses = PointMassesStr;
```
1: **** ERROR **** Utility Exception: "PointMassesStr" is not enclosed with "{}" in line:
" 152: MyForceModel.PointMasses = PointMassesStr;"

```
MyXYPlot.YVariables = Sat1VXVYStr;
```
Runtime error - Subscriber Exception: The Y parameter: Sat1VXVYStr of MyXYPlot is not plottable.

```
MyOrbitView.Add = LunaSat2Str;
```
Runtime error - GmatBase Exception Thrown: SetRefObject(*obj, Parameter, LunaSat2Str) not defined for OrbitView named "MyOrbitView"

### Behavior on Output

- Object name/s without single quotes inside curly brackets

## Unsigned Integer Type

Open Issue: Behavior is not defined list with empty brackets.

### Real Number Fields

### Integer Fields

## Built-in Data Types

String

Variable

Array

### Search Path Order

## Resources

# Eclipse Locator

Status:  Brainstorm/early draft
Owner: JJKP

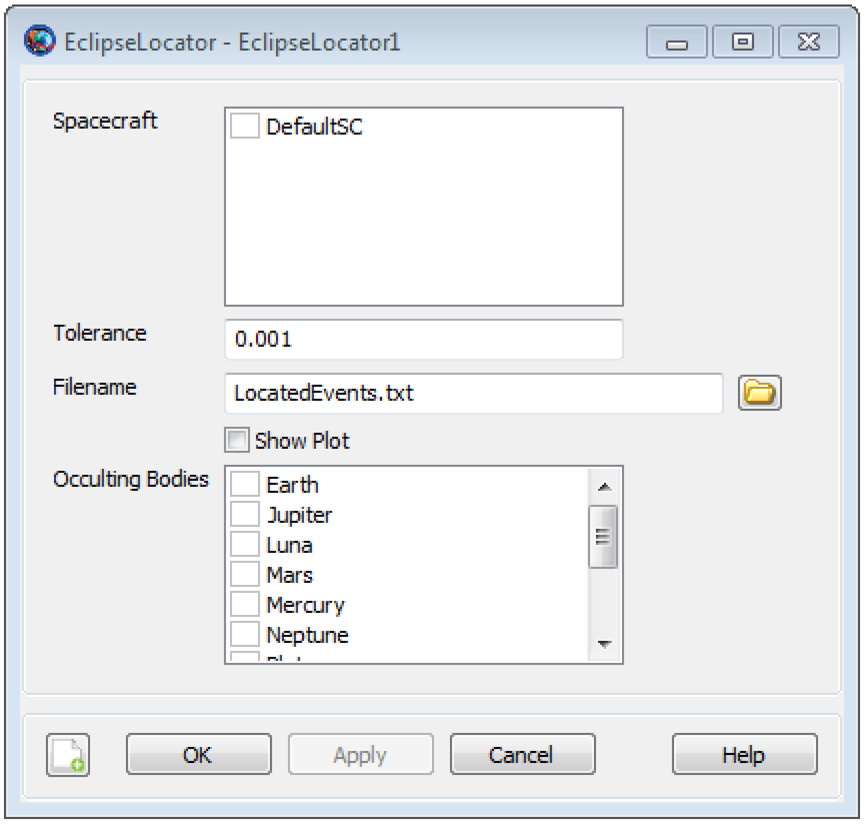
## Overview

Locate eclipse entry and exit times when spacecraft are in the shadow of a celestial body.

## Description

The Eclipse Locator object allows you to select a set of occulting bodies and a set of spacecraft and locates Umbra, Penumbra, and Antumbra eclipse types for all combinations of spacecraft and occulting Body.  GMAT integrates the event function derivative  adaptive stepping of event location roots.  Complex feature see Remarks section for more detailed description and behavior.

## GUI


EclipseLocator - EclipseLocator1

Spacecraft: DefaultSC

Tolerance: 0.001

Filename: LocatedEvents.txt

Show Plot

Occulting Bodies: Earth, Jupiter, Luna, Mars, Mercury, Neptune

Open Issue:  When you create an event locator there are no ojbects selected by default.  This is inconstent with other objects. For example, if you add a GroundTrackPlot, an existing spacecraft is selected for plotting.  Perhaps the issue is with GroundTrackPlot, but we need to discuss desired behavior and consistency.

## Fields

## Remarks

Caution:  To avoid missing events such as short eclipses, GMAT integrates the event function derivative along with the orbital equations of motion.  This can cause performance problems with many of the default integrator Accuracy settings.  If you are experiencing performance problems with event location, try using a larger value for the Accuracy field on the numberic integrator that still provides the required accuracy for your application.

### Event Report Location and Format

Event reports are written to the output folder located in the GMAT root directory.  The report

```
Type          Participants   Duration (sec)   UTC Start Time            UTC End Time
--------      ------------   --------------   ------------------------
------------------------
Penumbra      aSat - Earth   2106.326569   01 Jan 2000 12:10:05.356   01 Jan 2000
12:45:11.682
Umbra         aSat - Earth   2085.680368   01 Jan 2000 12:10:15.736   01 Jan 2000
12:45:01.416
Penumbra      aSat - Earth   2105.016963   01 Jan 2000 13:51:02.821   01 Jan 2000
14:26:07.838
Umbra         aSat - Earth   2085.143270   01 Jan 2000 13:51:12.536   01 Jan 2000
14:25:57.679
Penumbra      aSat - Earth   2105.166245   01 Jan 2000 15:31:58.893   01 Jan 2000
16:07:04.059
Umbra         aSat - Earth   2084.464079   01 Jan 2000 15:32:09.441   01 Jan 2000
16:06:53.905

Event Report Summary
--------------------
  Max Penumbra Duration             :      2106.327 s (aSat - Earth)
  Max Umbra Duration                :      2085.680 s (aSat - Earth)

  Number of Penumbra Events         : 3
  Number of Umbra Events            : 3
```

When GMAT fails to locate events, the event report file is not written to disk. If you attempt to open the file in the Output Tree, the editor contains the text: *** No events were found ***
If you stop a GMAT run by clicking the Stop button in the toolbar, then GMAT adds the following text at the top of the event report
*** NOTICE: Execution was interrupted; the event list may be incomplete ***

Behavior when reversing propagation direction.

Maximizing performance - if an occulting body is not significant, don't include it in the list.

Behavior when mission is stopped by the user

**Behavior when no events are located.**

**Behavior when Assignments  commands are applied to Spacecraft**

It is possible to create physical discontinuities in the orbital motion by using an assignment command to modify a spacecraft. In fact, if the state of a spacecraft is modified by any command other than the Maneuver or Propagate command, discontinuities in the trajectory are likely. When an assignment command is encountered in a mission, and a spacecraft is in shadow before the assignment command, GMAT resets the event locator and will not attempt to match eclipse exit with eclipse entry in part because the assignment command could potentially remove the spacecraft from the shadow region resulting in a mismatched event.

Below is an illustrative example. After the initial Propagate command, spacecraft aSat is in shadow. During the second propagate command, the spacecraft exits the shadow region. However, because an assignment command is issued, GMAT does not attempt to match the shadow entry with shadow exit or compute the duration of the shadow event as illustrated in the resulting event report.

```
Create Spacecraft aSat aSat2
Create EclipseLocator aLocator
aLocator.OccultingBodies = {Earth};
aLocator.Spacecraft   = {aSat};
Create Propagator aPropagator

BeginMissionSequence

Propagate aPropagator(aSat,{aSat.ElapsedDays = .03})
aSat.X  = aSat.X + .0001;
Propagate aPropagator(aSat,{aSat.ElapsedDays = .01})
```

Open Issue:  Nail-down menu options that occur when you left-click in an event report. Currently there are many UniCode options in the menu that need to be hidden.  Right-to-Left reordering moves data around and should probably be removed from menu or made inactive.

Open Issue:  Can setting minimum step on an integrator to be greater than zero cause events to be missed because adpative step algorithm isn't controlling the step size?  If so, should we should throw a warning or just put caution statement in the user's guide.   I lean towards a warning. -SPH

Discuss behavior when locating events in iterative processes.

Behavior when using control flow.

Caution:  In future versions of GMAT, the event location tolerance  will be changed to a tolerance on the event time.  Currently the tolerance applies to event function value.

Caution:  The adaptive stepping to avoid skipping events is not supported when using SPK propagation.  Events longer than the StepSize value on the propagator may be missed.

## Examples

## EphemerisFile

Status: Brainstorm/early draft
Owner: Not Yet Assigned

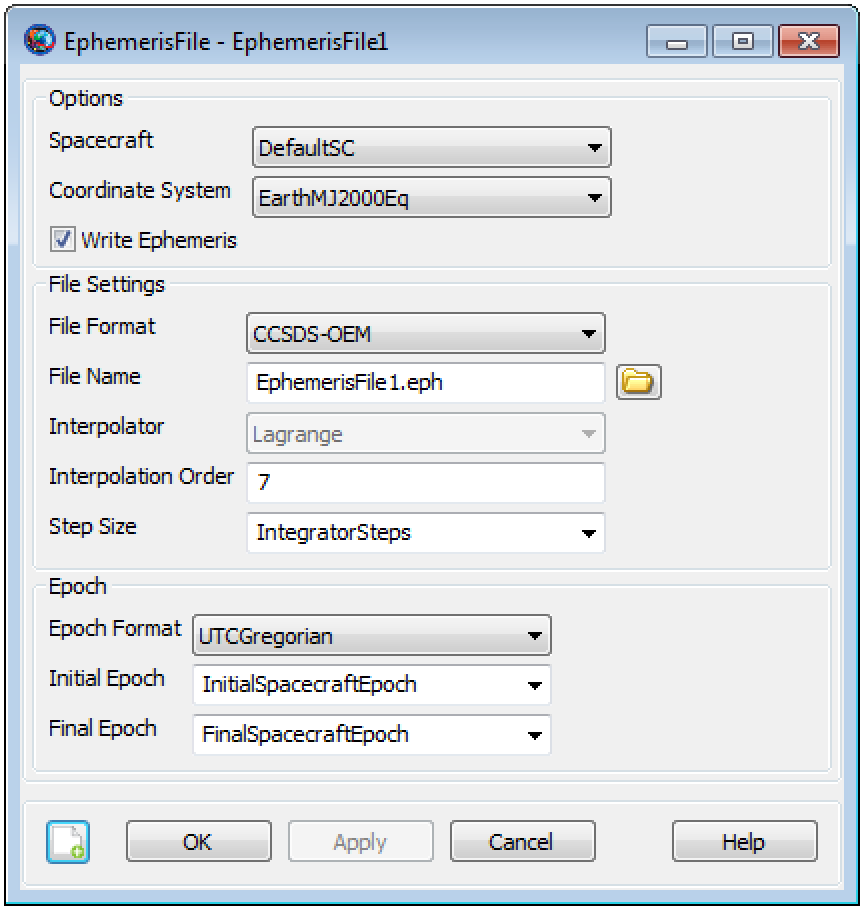
### Overview

The EphemerisFile object allows you to create files that contain the time history of a spacecraft's position and velocity. Currently, two formats are supported: CCSDS OEM (V1.0) and SPK. The CCSDS-OEM format is a textual ephemeris file while the SPICE format is binary. The behavior of the two ephemeris types is different because they are independent standards and the differences are discussed in detail in the Remarks section below.

Using the EphemerisFile object you can select the spacecraft and coordinate system for your ephemeris data (see below for limitations when using of SPICE) and the time span for the data you would like written to the file. You can also specify whether the the file should contain raw integrator time steps or interpolated data and in the case interpolation is selected you can choose the order of the interpolation.

### Script Syntax

### GUI


EphemerisFile - EphemerisFile1

Options

Spacecraft: DefaultSC

Coordinate System: EarthMJ2000Eq

☑ Write Ephemeris

File Settings

File Format: CCSDS-OEM

File Name: EphemerisFile1.eph

Interpolator: Lagrange

Interpolation Order: 7

Step Size: IntegratorSteps

Epoch

Epoch Format: UTCGregorian

Initial Epoch: InitialSpacecraftEpoch

Final Epoch: FinalSpacecraftEpoch

```
Open Issue:  If coordinate system is ignored for SPK ephem, then
where does the central body come from?
```

```
Open Issue:  Toggle command is currently not working for ephemeris
file but ephem files do show up in GUI on Toggle command panel.
Should we implement the connection or remove from GUI and document?
Should talk to LOJ about effort for each of these options.
```

## Remarks

CCSDS-OEM Behavior

When using the CCSDS-OEM file format, GMAT currently requires that Lagrange interpolation is selected.

## SPK Behavior

SPICE kernels and the SPICE utilities perform their own interpolation and GMAT uses the Hermite option. When using SPICE, if you set the step size to a numeric value (other than the keywork “IntegratorSteps”) then GMAT interpolates the data to the desired value and then sends the data to the SPICE ephemeris file. The result is that two interpolations occur and it is recommended that when using SPICE ephemeris format you set StepSize to IntegratorSteps and the SPICE routines will handle the interpolation. If a file exists with the name provided in the filename field, GMAT renames the existing file so that the data is not lost and writes the new ephemeris to the requested filename. RESOLVE THE COORDINATE SYSTEM ISSUE.

## Other Behavior

non-monotonic data
discontinuities and ephemeris segments.

The system shall throw an error message and stop interpolation if there is not enough data available to support the requested interpolation type and order.

The system shall only write the ephemeris for the solution of iterative processes such as differential correction, optimization, and estimation.

The system shall throw an error message and stop interpolation if the data points provided to the interpolator are not monotonic .

The system shall allow the user to apply the Toggle command described in FRC-16 to the Ephemeris File object.

The system shall write the last data point to an ephemeris segment using the exact ephemeris values when interpolating at a requested time step and the ephemeris interval is not an integer multiple of the requested interpolation step.

The system shall separately interpolate ephemeris segments bounded by the following discrete events:

1) Impulsive maneuver
2) Dynamics model change ( change in propagator or finite burn)
3) Assignment command

```
Open Issue:  CCSDS does not require monotonic ephemeris file.
Currently if you forward and then backprop, CCSDS has all data. SPK
throws an error message but GMAT proceeds: "  Message received from
CSPICE is: At least 4 states are required to define a Hermite
polynomial of degree 7...."  I lean towards documenting this and
moving on rather than changing GMAT to throw an exception in this
case. -SPH
```

## Examples

Create a CCSDS-OEM file with 2 minute time steps using 9th order Lagrange interpolation. This script snippet creates an ephemeris file for the entire propagation time span.

```
Create Spacecraft aSat
Create Propagator aPropagator

Create EphemerisFile anEphem
anEphem.Spacecraft = aSat
anEphem.Filename = 'myEphemFile.eph';
anEphem.InitialEpoch = InitialSpacecraftEpoch;
anEphem.FinalEpoch = FinalSpacecraftEpoch;
GMAT anEphem.InterpolationOrder = 9;
anEphem.StepSize = 120;

BeginMissionSequence

Propagate aPropagator(aSat,{aSat.Apoapsis})
```

Create an SPK ephemeris file with 2 minute time steps using 5th order Hermite interpolation.

```
Create Spacecraft aSat
aSat.NAIFId = 123456
Create Propagator aPropagator

Create EphemerisFile anEphem
anEphem.Spacecraft = aSat
GMAT anEphem.FileFormat = SPK;
anEphem.Filename = 'myEphemFile.bsp';
GMAT anEphem.InterpolationOrder = 5;

BeginMissionSequence

Propagate aPropagator(aSat,{aSat.Apoapsis})
```

If an error occurs when writing to an SPK file, the file does not close and if you try to write to the file GMAT can't save it to a new name.  I see this message: Utility Exception: Unknown system error occurred when attempting to rename existing SPK file "../output/myEphemFile8.bsp".