JSC-61777-models/utils/message

JSC Engineering Orbital Dynamics Message Model

Simulation and Graphics Branch (ER7)
Software, Robotics, and Simulation Division
Engineering Directorate

Package Release JEOD v5.1

Document Revision 1.0
July 2023

National Aeronautics and Space Administration
Lyndon B. Johnson Space Center
Houston, Texas

JSC Engineering Orbital Dynamics
Message Model

Document Revision 1.0
July 2023

Dr. Robert Shelton and Dr. Blair Thompson

Simulation and Graphics Branch (ER7)
Software, Robotics, and Simulation Division
Engineering Directorate

National Aeronautics and Space Administration
Lyndon B. Johnson Space Center
Houston, Texas

Abstract

The MessageHandler class provides a generic framework for routing various informational, warning, error and failure messages to the user. The TrickMessageHandler derived class provides the specific implementation of these capabilities within the Trick environment. The message model is designed in such a way that the user can configure selective suppression of various message types. The user may customize the message format by specifying whether error codes and location information should be included.

This document describes the implementation of the Message Model. It is part of a series of inter-related documents that describe the model requirements, specifications, and architecture of the model. A user guide is also provided to assist with implementing the model in Trick simulations.

Contents

Chapter 1

Introduction

1.1 Model Description

The JEOD Message Model provides a convenient, flexible, consistent facility for users and developers alike to communicate information and program status back to the user through a TTY interface. The Message Model offers parameters which control the format of messages and the level of screening desired, thereby allowing flexibility for users and developers to specify which messages will be sent and how they should be formatted under various conditions.

Most of the JEOD models and simulations make liberal use of the Message Model in order to convey information regarding the state of the simulation, errors, warnings, debug information and failure conditions. The control capabilities allow output to range from maximum debugging verbosity to a clean run of a production sim.

The parent document to this model document is the JEOD Top Level Document [1].

1.2 Document History

Author	Date	Revision	Description
Robert Shelton	Nov 2009	1.0	Initial document
Robert Shelton	July 2023	1.1	Description of connection with sim_interface model

1.3 Document Organization

This document is formatted in accordance with the NASA Software Engineering Requirements Standard [2] and is organized into the following chapters:

Chapter 1: Introduction - This introduction contains three sections: description of model, document history, and organization. The first section provides the introduction to the Message Model and its reason for existence. It contains a brief description of the interconnections with other models, and references to any supporting documents. It lists the document that is

parent to this one. The second section displays the history of this document which includes author, date, and reason for each revision. The final section contains a description of how the document is organized.

Chapter 2: Product Requirements - Describes requirements for the Message Model.

Chapter 3: Product Specification - Describes the underlying theory, architecture, and design of the Message Model in detail. It is organized in three sections: Conceptual Design, Mathematical Formulations, and Detailed Design.

Chapter 4: User Guide - Describes how to use the Message Model in a Trick simulation. It is broken into three sections to represent the JEOD defined user types: Analysts or users of simulations (Analysis), Integrators or developers of simulations (Integration), and Model Extenders (Extension).

Chapter 5: Verification and Validation - Contains Message Model verification and validation procedures and results.

Chapter 2

Product Requirements

This model shall meet the JEOD project requirements specified in the JEOD top-level document [1].

Requirement MH_1: Failure Messages

Requirement:
The Message Model shall be capable of accepting a failure message; conveying it to the user in a console window; and terminating the simulation. Failure messages are never suppressed.

Rationale:
A key function of the Message Model is to convey information to the user regarding fatal errors followed by immediate termination of the simulation.

Verification:
Test

Requirement MH_2: Error Messages

Requirement:
The Message Model shall be capable of accepting an error message and conveying it to the user in a console window. Error messages can be conditionally suppressed by setting the suppression level parameter.

Rationale:
A key function of the Message Model is to convey error messages to the user conditionally depending on a user-settable suppression level parameter.

Verification:
Test

Requirement MH_3: Warning Messages

Requirement:

The Message Model shall be capable of accepting a warning message and conveying it to the user in a console window. Warning messages can be conditionally suppressed by setting the suppression level parameter.

Rationale:

A key function of the Message Model is to convey warning messages to the user conditionally depending on a user-settable suppression level parameter.

Verification:

Test

Requirement MH_4: Informational Messages

Requirement:

The Message Model shall be capable of accepting an informational message and conveying it to the user in a console window. Informational messages can be conditionally suppressed by setting the suppression level parameter.

Rationale:

A key function of the Message Model is to convey informational messages to the user conditionally depending on a user-settable suppression level parameter.

Verification:

Test

Requirement MH_5: Debugging Messages

Requirement:

The Message Model shall be capable of accepting a debugging message and conveying it to the user in a console window. Debugging messages can be conditionally suppressed by setting the suppression level parameter.

Rationale:

A key function of the Message Model is to convey debugging messages to the user conditionally depending on a user-settable suppression level parameter.

Verification:

Test

Requirement MH_6: Generic Messages

Requirement:

The Message Model shall be capable of accepting a generic message and conveying it to the user in a console window. Generic messages can be conditionally suppressed by setting the suppression level parameter.

Rationale:

A key function of the Message Model is to convey generic messages to the user conditionally depending on a user-settable suppression level parameter.

Verification:

Test

Chapter 3

Product Specification

3.1 Conceptual Design

The main purpose of the Message Model is to provide a common interface for display or logging of debugging, informational, warning, error or failure messages. The Message Model allows format and verbosity of messages to be tailored to the needs of the user by means of input variables. MessageHandler is a base class with one pure virtual method (MessageHandler::process_message) which must be implemented in a derived class in order to create a functional Message Model. The TrickMessageHandler is such a derived class and provides an implementation of process_message using the Trick methods exec_terminate for fatal errors and send_hs for all other messages. If other avenues of output were needed, then a different subclass of MessageHandler could be substituted for TrickMessageHandler in order to provide the desired functionality.

3.2 Mathematical Formulations

The Message Model includes no mathematical algorithms.

3.3 Detailed Design

The Message Model includes a protected field, MessageHandler::suppression_level, which the user can set in the input file in order to control which messages will be suppressed. There are also public static integer constants which indicate levels of severity.

Constant	Value	Meaning
MessageHandler::Fail	-1	Prints a message and terminates simulation
MessageHandler::Error	0	Most severe error which does not terminate the simulation
MessageHandler::Warning	9	Indicates results are suspect
MessageHandler::Notice	99	Indicates an informational message
MessageHandler::Debug	999	Intended for messages to inform users of expected events

Table 3.1: Symbolic Constants for Severity Levels

The public methods of the Message Model are:

- Fail – deliver a failure message and terminate the simulation
- Error – deliver an error message of severity MessageHandler::Error
- Warn – Deliver a warning message with severity MessageHandler::Warning
- Inform – Deliver an informational message with severity MessageHandler::Notice
- Debug – Deliver a debugging message with severity MessageHandler::Debug
- Send_message – Deliver a generic message with severity and labeling specified by the user
- Va_send_message – Just like send_message except that an argument list defined by starg.h replaces the variadic arguments of send_message

All of these methods are declared static, thus, there is effectively only one MessageHandler in any given simulation. These static methods are all wrappers for the virtual method process_message, which, due to limitations of C++ cannot be a static method. The programming technique used to circumvent this issue is to include a static field (handler, of type MessageHandler) in the MessageHandler class. The default initialization of handler is NULL, and upon first instantiation of MessageHandler, handler is set to this instance. Subsequent instantiations of the MessageHandler class have no effect on the value of the static field handler. With the introduction of the sim_interface model, the MessageHandeler is constructed automatically. See Chapter 4 for details.

For the user, the implications are as follows:

- There will only be one instance of the MessageHandler, and that one is created automatically with the construction of the JeodSimulationInterface.
- Changes made to instance fields such as suppression_level, suppress_id and suppress_location are only effective if performed on the initial instance of MessageHandler

Additional details of the Message Model design can be found in JEOD Message Model Reference Manual [3].

.

3.4 Inventory

All Message Model files are located in the directory $JEOD_HOME/models/utils/message. Tables **??** to **??** list the configuration-managed files in this directory.

Chapter 4

User Guide

The User Guide is composed of three sections. The Analysis section is intended primarily for users of pre-existing simulations. The Integration section of the user guide is intended for simulation developers. The Extension section of the user guide is intended primarily for developers needing to extend the capability of the Message Model. Such users should have a thorough understanding of how the model is used in the preceding Integration section, and of the model specification (described in Chapter 3).

4.1 Analysis

The primary purpose of the Message Model is to provide analysts (i.e., sim users) useful information regarding the status of a sim while it is running. Some messages are related to the intentional termination of a sim. Other messages may occur without sim termination. Sim users can use the information conveyed by the Message Model to help ensure their sim is running correctly or to report possible problems in the JEOD models to the JEOD development team. When reporting a message be sure to capture the entire message. This will aid with the determination and correction of the problem.

An example of a message from the JEOD Gravity Model is:

```
SIMULATION TERMINATED IN
  PROCESS: 1
  JOB/ROUTINE: 2/gravity_source.cc line 551
DIAGNOSTIC:
Error environment/gravity ord_exceeds_deg_error:
Gravity field order (80) is greater than gravity field degree (70).
```

In this case, the user requested a gravity order which is larger than the gravity degree.

The analyst has three inputs to control the types of messages displayed and the information related to those messages:

suppression_level - Default value: 10 (warnings and non-fatal errors). All messages have an associated severity level, with increasingly positive values indicating warnings of decreasing severity.

Fatal errors have a negative severity level. Messages whose severity equals or exceeds the value of the global message handler's suppression_level are suppressed. Note that fatal errors cannot be suppressed.

suppress_id - Default value: false. This flag indicates whether the message ID is printed for unsuppressed messages. The ID is not printed if this flag is set to true. The message ID is always printed for errors.

suppress_location - Default value: false. This flag indicates whether the message source file and line number are printed for unsuppressed messages. The location is not printed if this flag is set to true. The message location is always printed for errors.

The first Trick sim_object created should contain an instance of JeodSimulationInterface, which will automatically construct a singleton MessageHandler for the sim. Here is a typical definition of such a sim_object.

```
/* ==================== JEOD System Simulation Object ==================== **/
/* ======================================================================== **/
sim_object { /* jeod_sys
-------------------------------------------------*/
   /*=======================================================================
    =  This is the JEOD executive model, this model should be basically
    =  the same for all JEOD-based Trick applications.
=
    ========================================================================*/
   /*==================
    = DATA Structures =
    ==================*/
   utils/sim_interface/include: JeodTrickSimInterface sim_interface;
} jeod_sys;
```

An example of how these three controls might be set in an input file is:

```
jeod_sys.sim_interface.message_handler.suppression_level = MessageHandler::Warning;
jeod_sys.sim_interface.message_handler.suppress_id       = true;
jeod_sys.sim_interface.message_handler.suppress_location = false;
```

In this example case, only errors will be reported; warnings and other messages with a higher severity level will be suppressed. For any message that is not suppressed, the ID number will be suppressed and location of any messages will be displayed.

4.2 Integration

The use of the Message Model by sim integrators is essentially the same as for sim analysts. The severity level of suppressed messages can be changed to suit the needs of the sim integrator. See previous section for more information.

4.3 Extension

Sim developers can add messages to a model. The messages are normally defined in the model class in the **include** directory. For example, messages for the JEOD Gravity Model are defined in **models/environment/gravity/include** in the file gravity_messages.hh as shown below:

```
   // Errors
   static char const * max_deg_error; /* --
      Error issued when requested gravity field degree exceeds maximum */

   static char const * max_ord_error; /* --
      Error issued when requested gravity field order exceeds maximum */

   static char const * ord_exceeds_deg_error; /* --
      Error issued when requested gravity field order is greater than degree */


   // Warnings
   static char const * radial_distance_warning; /* --
      Warning issued when radial distance is less than equatorial radius of
      gravity model */
```

The same messages must also be added to a corresponding file in the **src** directory, such as:

```
// Errors
char const * GravityMessages::max_deg_error =
    PATH " max_deg_error";

char const * GravityMessages::max_ord_error =
    PATH " max_ord_error";

char const * GravityMessages::ord_exceeds_deg_error =
    PATH " ord_exceeds_deg_error";

// Warnings
char const * GravityMessages::radial_distance_warning =
    PATH " radial_distance_warning";
```

Finally, the code that actually triggers the message must be included in the proper source code file. For the Gravity Model example, the message trigger code appears in the gravity_body.cc file. An example of a **failure** and a **warning** trigger are:

```
      // check if maximum order to be used for computations is greater than
      // maximum degree
      if (controls.order > controls.degree) {
```

```
   MessageHandler::fail (
   __FILE__, __LINE__, GravityMessages::ord_exceeds_deg_error,
   "Gravity field order (%i) is greater than gravity field degree (%i).\n",
   controls.order,controls.degree);
   return;

}

// check if radial distance is less than equatorial radius
if (r_mag < radius) {
   MessageHandler::warn (
      __FILE__, __LINE__, GravityMessages::radial_distance_warning,
      "Radial distance (%E) is less than equatorial radius of gravity model (%E).\n",
      r_mag,radius);
}
```

Chapter 5

Verification and Validation

5.1 Verification

A verification sim was built to test the various combinations of message suppression. The sim is located at: **message/verif/SIM_message_handler_verif** This simulation is constructed in such a way that almost all the meaningful output is routed to standard error. There is a bin directory which contains a script (run_test) to run the sim through its tests. The output of the verification sim is shown below. :

```
Tests 1-5: Tests with default suppression settings

Test 1: send_message()
Message utils/message/verif/Message at message_handler_verif_driver.cc line 105:
Test of MessageHandler::send_message()


Test 2: debug()

Test 3: inform()

Test 4: warn()
Warning utils/message/verif/Warning at message_handler_verif_driver.cc line 83:
Test of MessageHandler::warn()


Test 5: error()
Error utils/message/verif/Error at message_handler_verif_driver.cc line 76:
Test of MessageHandler::error()



Tests 6-10: Suppression level set to 9
```

```
Test 6: send_message() (Not suppressed)
Message utils/message/verif/Message at message_handler_verif_driver.cc line 105:
Test of MessageHandler::send_message()


Test 7: debug() (Suppressed)
Test 8: inform() (Suppressed)

Test 9: warn() (Suppressed)

Test 10: error() (Not suppressed)
Error utils/message/verif/Error at message_handler_verif_driver.cc line 76:
Test of MessageHandler::error()



Tests 11-12: Supression level set to 0

Test 11: send_message() (Suppressed)
Test 10: error() (Suppressed)


Tests 13-15: Tests of suppressing ID, location

Test 13: error() (ID suppressed)
message_handler_verif_driver.cc line 76
Test of MessageHandler::error()


Test 14: error() (location suppressed)
Error utils/message/verif/Error:
Test of MessageHandler::error()


Test 15: error() (ID and location suppressed)
Test of MessageHandler::error()


Test 16: fail() (Never suppressed)
```

5.2 Validation

(No validation tests were conducted for the Message Model.)

5.3 Metrics

5.3.1 Code Metrics

Table 5.1 presents coarse metrics on the source files that comprise the model.

Table 5.1: Coarse Metrics

File Name	Number of Lines			
	Blank	Comment	Code	Total
Total	0	0	0	0

Table 5.2 presents the extended cyclomatic complexity (ECC) of the methods defined in the model.

Table 5.2: Cyclomatic Complexity

Method	File	Line	ECC
jeod::MessageHandler:: register_contents ()	include/message_handler.hh	238	1
jeod::MessageHandler:: deregister_contents ()	include/message_handler.hh	245	1
jeod::MessageHandler:: process_add_suppressed_ code (const char * msg_ code JEOD_UNUSED)	include/message_handler.hh	340	1
jeod::MessageHandler:: process_delete_suppressed_ code (const char * msg_ code JEOD_UNUSED)	include/message_handler.hh	355	1
jeod::MessageHandler:: process_clear_suppressed_ codes ()	include/message_handler.hh	370	1
jeod::SuppressedCodeMessage Handler::~SuppressedCode MessageHandler (void)	include/suppressed_code_ message_handler.hh	104	1
jeod::SuppressedCodeMessage Handler::process_add_ suppressed_code (const char * msg_code)	include/suppressed_code_ message_handler.hh	121	1

Continued on next page

Table 5.2: Cyclomatic Complexity (continued)

Method	File	Line	ECC
jeod::SuppressedCodeMessageHandler::process_delete_suppressed_code (const char * msg_code)	include/suppressed_code_message_handler.hh	132	1
jeod::SuppressedCodeMessageHandler::process_clear_suppressed_code (void)	include/suppressed_code_message_handler.hh	143	1
jeod::SuppressedCodeMessageHandler::message_is_to_be_printed (int severity, const char * msg_code)	include/suppressed_code_message_handler.hh	153	3
jeod::MessageHandler::fail (const char * file, unsigned int line, const char * msg_code, const char * format, ...)	src/message_handler.cc	67	2
jeod::MessageHandler::error (const char * file, unsigned int line, const char * msg_code, const char * format, ...)	src/message_handler.cc	110	2
jeod::MessageHandler::warn (const char * file, unsigned int line, const char * msg_code, const char * format, ...)	src/message_handler.cc	151	2
jeod::MessageHandler::inform (const char * file, unsigned int line, const char * msg_code, const char * format, ...)	src/message_handler.cc	187	2
jeod::MessageHandler::debug (const char * file, unsigned int line, const char * msg_code, const char * format, ...)	src/message_handler.cc	221	2

Continued on next page

Table 5.2: Cyclomatic Complexity (continued)

Method	File	Line	ECC
jeod::MessageHandler::send_message (int severity, const char * prefix, const char * file, unsigned int line, const char * msg_code, const char * format, ...)	src/message_handler.cc	256	2
jeod::MessageHandler::va_send_message (int severity, const char * prefix, const char * file, unsigned int line, const char * msg_code, const char * format, va_list args)	src/message_handler.cc	295	2
jeod::MessageHandler::set_suppression_level (unsigned int suppression_level)	src/message_handler.cc	334	2
jeod::MessageHandler::get_suppression_level (void)	src/message_handler.cc	355	2
jeod::MessageHandler::add_suppressed_code (const char * msg_code)	src/message_handler.cc	386	2
jeod::MessageHandler::delete_suppressed_code (const char * msg_code)	src/message_handler.cc	408	2
jeod::MessageHandler::clear_suppressed_codes (void)	src/message_handler.cc	429	2
jeod::MessageHandler::set_suppress_id (bool suppress_id)	src/message_handler.cc	446	2
jeod::MessageHandler::get_suppress_id (void)	src/message_handler.cc	467	2
jeod::MessageHandler::set_suppress_location (bool suppress_location)	src/message_handler.cc	491	2
jeod::MessageHandler::get_suppress_location (void)	src/message_handler.cc	512	2
jeod::MessageHandler::no_handler_error (void)	src/message_handler.cc	537	1

Continued on next page

Table 5.2: Cyclomatic Complexity (continued)

Method	File	Line	ECC
jeod::MessageHandler:: MessageHandler (void)	src/message_handler.cc	562	2
jeod::MessageHandler::~ MessageHandler (void)	src/message_handler.cc	592	2
jeod::SuppressedCodeMessage Handler::register_contents (void)	src/suppressed_code_message_ handler.cc	44	1
jeod::SuppressedCodeMessage Handler::deregister_contents (void)	src/suppressed_code_message_ handler.cc	56	1

Bibliography

[1] Jackson, A., Thebeau, C. JSC Engineering Orbital Dynamics. Technical Report JSC-61777-docs, NASA, Johnson Space Center, Houston, Texas, July 2023.

[2] NASA. NASA Software Engineering Requirements. Technical Report NPR-7150.2, NASA, NASA Headquarters, Washington, D.C., September 2004.

[3] National Aeronautics and Space Administration, Johnson Space Center, Software, Robotics & Simulation Division, Simulation and Graphics Branch, 2101 NASA Parkway, Houston, Texas, 77058. *JEOD Message Model Reference Manual*, July 2023.