SimulationInterfaceMacro

5.0

Generated by Doxygen 1.8.14

Contents

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Module Documentation

6.1 Models

Modules

- Utils

6.1.1 Detailed Description

6.2 Utils

Modules

- SimInterface

6.2.1 Detailed Description

6.3 SimInterface

Files

- file checkpoint_input_manager.hh

  Define class CheckPointInputManager and related classes.
- file checkpoint_output_manager.hh

  Define class CheckPointOutputManager and related classes.
- file class_declarations.hh

  Forward declarations of classes defined in the utils/sim_interface model.
- file config.hh

  Configure JEOD for use by some simulation engine.
- file config_test_harness.hh

  Configure JEOD for use in standalone test mode.
- file config_trick10.hh

  Configure JEOD for use in a Trick10 environment.
- file jeod_class.hh

  Define the JEOD class declaration macros JEOD_MAKE_SIM_INTERFACES and and JEOD_DECLARE_SIM_IN↩ TERFACES.
- file jeod_integrator_interface.hh

  Define the interface for accessing / updating elements of a simulation engine's integrator object.
- file jeod_trick_integrator.hh

  Define the interface for accessing / updating elements of a Trick simulation integrator object.
- file memory_attributes.hh

  Define JEOD memory interface macros.
- file memory_interface.hh

  Define the MemoryInterface class, which abstractly defines the interface between the memory manager and the simulation engine.
- file sim_interface_messages.hh

  Define the class SimInterfaceMessages, the class that specifies the message IDs used in the sim interface model.
- file simulation_interface.hh

  Define the abstract class JeodSimulationInterface.
- file trick10_memory_interface.hh

  Define the interface for registering / deregistering memory with Trick.
- file trick_dynbody_integ_loop.hh

  Define the class IntegrationGroupIntegLoopScheduler, which replaces the base Trick integration loop for multi-rate JEOD-based simulations.
- file trick_memory_interface.hh

  Define the interface for registering / deregistering memory with Trick.
- file trick_message_handler.hh

  Define the class TrickMessageHandler, the message handler designed for use in Trick-based simulations.
- file trick_sim_interface.hh

  Define the class JeodTrickSimInterface.
- file checkpoint_input_manager.cc

  Define CheckPointInputManager member functions and of related classes.
- file checkpoint_output_manager.cc

  Define CheckPointOutputManager member functions and of related classes.
- file memory_interface.cc

  Implement the MemoryInterface class.
- file sim_interface_messages.cc

  Implement the class SimInterfaceMessages.

- file simulation_interface.cc

  Implement SimulationInterface methods.
- file trick10_memory_interface.cc

  Define JeodTrickMemoryInterface methods.
- file trick_dynbody_integ_loop.cc

  Define JeodDynbodyIntegrationLoop methods.
- file trick_memory_interface.cc

  Define JeodTrickMemoryInterface methods.
- file trick_memory_interface_alloc.cc

  Define JeodTrickMemoryInterface methods related to allocation/deallocation.
- file trick_memory_interface_attrib.cc

  Define JeodTrickMemoryInterface methods related to attributes.
- file trick_memory_interface_chkpnt.cc

  Define JeodTrick10MemoryInterface methods related to checkpoint/restart.
- file trick_memory_interface_xlate.cc

  Define JeodTrickMemoryInterface methods related to name translation.
- file trick_message_handler.cc

  Define member functions for the class TrickMessageHandler.
- file trick_sim_interface.cc

  Implement TrickSimInterface methods.

Namespaces

- jeod

  Namespace jeod.
- Trick

  Namespace Trick furnishes several standard functions for use in the Trick environment.
- er7_utils

  Namespace er7_utils contains the state integration models used by JEOD.

Macros

- #define JEOD_UNUSED
- #define ER7_UTILS_UNUSED
- #define ER7_UTILS_RESTRICT
- #define ER7_UTILS_ALWAYS_INLINE
- #define JEOD_ATTRIBUTES_TYPE int
- #define JEOD_ATTRIBUTES_POINTER_TYPE void *
- #define JEOD_SIM_INTEGRATOR_POINTER_TYPE void *
- #define JEOD_SIZE_T size_t
- #define JEOD_PTRDIFF_T long int
- #define JEOD_INTPTR_T long int
- #define JEOD_UINTPTR_T unsigned long int
- #define JEOD_CLASS_ESTABLISH_FRIENDS(class_name)
- #define JEOD_ATTRIBUTES_SIM_ENGINE_HEADER "sim_services/MemoryManager/include/attributes.h"
- #define JEOD_ATTRIBUTES_TYPE struct ATTRIBUTES_tag
- #define JEOD_ATTRIBUTES_POINTER_TYPE JEOD_ATTRIBUTES_TYPE *
- #define JEOD_SIM_INTEGRATOR_SIM_ENGINE_HEADER "sim_services/Integrator/include/Integrator.hh"
- #define JEOD_SIM_INTEGRATOR_FORWARD namespace Trick { class Integrator; }
- #define JEOD_SIM_INTEGRATOR_POINTER_TYPE Trick::Integrator *

- #define JEOD_SIM_INTEGRATOR_ENUM Integrator_type
- #define JEOD_MAKE_SIM_INTERFACES(class_name) JEOD_CLASS_ESTABLISH_FRIENDS(class_↩ name)

  JEOD_MAKE_SIM_INTERFACES(class_name) Defines friends of the given class.
- #define JEOD_DECLARE_SIM_INTERFACES(class_name)

  JEOD_DECLARE_SIM_INTERFACES(class_name) Forward declare classes and external functions needed to make the JEOD_MAKE_SIM_INTERFACES(class_name) expansion compilable.
- #define PATH "utils/sim_interface/"
- #define CLASS SimInterfaceMessages
- #define MAKE_MESSAGE_CODE(id) char const * CLASS::id = PATH #id
- #define MAX_MSG_SIZE 4096

Variables

- Trick::MemoryManager * trick_MM
- Trick::Integrator * trick_curr_integ
- Trick::MemoryManager * trick_MM
- Trick::MemoryManager * trick_MM
- Trick::MemoryManager * trick_MM

6.3.1 Detailed Description

6.3.2 Macro Definition Documentation

6.3.2.1 CLASS

```
#define CLASS SimInterfaceMessages
```

Definition at line 38 of file sim_interface_messages.cc.

6.3.2.2 ER7_UTILS_ALWAYS_INLINE

```
#define ER7_UTILS_ALWAYS_INLINE
```

Definition at line 113 of file config.hh.

6.3.2.3 ER7_UTILS_RESTRICT

```
#define ER7_UTILS_RESTRICT
```

Definition at line 108 of file config.hh.

6.3.2.4 ER7_UTILS_UNUSED

```
#define ER7_UTILS_UNUSED
```

Definition at line 103 of file config.hh.

6.3.2.5 JEOD_ATTRIBUTES_POINTER_TYPE [1/2]

```
#define JEOD_ATTRIBUTES_POINTER_TYPE void *
```

Definition at line 80 of file config_test_harness.hh.

6.3.2.6 JEOD_ATTRIBUTES_POINTER_TYPE [2/2]

```
#define JEOD_ATTRIBUTES_POINTER_TYPE JEOD_ATTRIBUTES_TYPE *
```

Definition at line 93 of file config_trick10.hh.

6.3.2.7 JEOD_ATTRIBUTES_SIM_ENGINE_HEADER

```
#define JEOD_ATTRIBUTES_SIM_ENGINE_HEADER "sim_services/MemoryManager/include/attributes.h"
```

Definition at line 90 of file config_trick10.hh.

6.3.2.8 JEOD_ATTRIBUTES_TYPE [1/2]

```
#define JEOD_ATTRIBUTES_TYPE int
```

Definition at line 79 of file config_test_harness.hh.

6.3.2.9 JEOD_ATTRIBUTES_TYPE [2/2]

```
#define JEOD_ATTRIBUTES_TYPE struct ATTRIBUTES_tag
```

Definition at line 92 of file config_trick10.hh.

6.3.2.10 JEOD_CLASS_ESTABLISH_FRIENDS

```
#define JEOD_CLASS_ESTABLISH_FRIENDS(
              class_name )
```

Value:

```
friend class InputProcessor; \
   friend void init_attrjeod__ ## class_name();
```

Definition at line 77 of file config_trick10.hh.

6.3.2.11 JEOD_DECLARE_SIM_INTERFACES

```
#define JEOD_DECLARE_SIM_INTERFACES(
              class_name )
```

JEOD_DECLARE_SIM_INTERFACES(class_name) Forward declare classes and external functions needed to make the JEOD_MAKE_SIM_INTERFACES(class_name) expansion compilable.

All JEOD files that use JEOD_MAKE_SIM_INTERFACES within classes (will) make a parallel call to this macro at file scope in the global namespace.

Parameters

class_name	Name of the class defined later in the header in question.

Definition at line 138 of file jeod_class.hh.

6.3.2.12 JEOD_INTPTR_T

```
#define JEOD_INTPTR_T long int
```

Definition at line 68 of file config_trick10.hh.

6.3.2.13 JEOD_MAKE_SIM_INTERFACES

```
#define JEOD_MAKE_SIM_INTERFACES(
              class_name ) JEOD_CLASS_ESTABLISH_FRIENDS(class_name)
```

JEOD_MAKE_SIM_INTERFACES(class_name) Defines friends of the given class.

This macro is to be invoked in the body of all JEOD classes. The intent is to make all parts of the class visible to the designated simulation engine classes and functions.

Parameters

class_name	Name of the class being defined.

Definition at line 102 of file jeod_class.hh.

6.3.2.14 JEOD_PTRDIFF_T

```
#define JEOD_PTRDIFF_T long int
```

Definition at line 67 of file config_trick10.hh.

6.3.2.15 JEOD_SIM_INTEGRATOR_ENUM

```
#define JEOD_SIM_INTEGRATOR_ENUM Integrator_type
```

Definition at line 108 of file config_trick10.hh.

6.3.2.16 JEOD_SIM_INTEGRATOR_FORWARD

```
#define JEOD_SIM_INTEGRATOR_FORWARD namespace Trick { class Integrator; }
```

Definition at line 105 of file config_trick10.hh.

6.3.2.17 JEOD_SIM_INTEGRATOR_POINTER_TYPE [1/2]

```
#define JEOD_SIM_INTEGRATOR_POINTER_TYPE void *
```

Definition at line 90 of file config_test_harness.hh.

6.3.2.18 JEOD_SIM_INTEGRATOR_POINTER_TYPE [2/2]

```
#define JEOD_SIM_INTEGRATOR_POINTER_TYPE Trick::Integrator *
```

Definition at line 107 of file config_trick10.hh.

6.3.2.19 JEOD_SIM_INTEGRATOR_SIM_ENGINE_HEADER

```
#define JEOD_SIM_INTEGRATOR_SIM_ENGINE_HEADER "sim_services/Integrator/include/Integrator.hh"
```

Definition at line 103 of file config_trick10.hh.

6.3.2.20 JEOD_SIZE_T

```
#define JEOD_SIZE_T size_t
```

Definition at line 66 of file config_trick10.hh.

6.3.2.21 JEOD_UINTPTR_T

```
#define JEOD_UINTPTR_T unsigned long int
```

Definition at line 69 of file config_trick10.hh.

6.3.2.22 JEOD_UNUSED

```
#define JEOD_UNUSED
```

Definition at line 98 of file config.hh.

6.3.2.23 MAKE_MESSAGE_CODE

```
#define MAKE_MESSAGE_CODE(
              id ) char const * CLASS::id = PATH #id
```

Definition at line 39 of file sim_interface_messages.cc.

6.3.2.24 MAX_MSG_SIZE

```
#define MAX_MSG_SIZE 4096
```

Definition at line 45 of file trick_message_handler.cc.

Referenced by jeod::TrickMessageHandler::process_message().

6.3.2.25 PATH

```
#define PATH "utils/sim_interface/"
```

Definition at line 37 of file sim_interface_messages.cc.

6.3.3 Variable Documentation

6.3.3.1 trick_curr_integ

```
Trick::Integrator* trick_curr_integ
```

Referenced by jeod::JeodDynbodyIntegrationLoop::integrate_dt().

6.3.3.2 trick_MM [1/4]

```
Trick::MemoryManager* trick_MM
```

6.3.3.3 trick_MM [2/4]

```
Trick::MemoryManager* trick_MM
```

Referenced by jeod::JeodTrickMemoryInterface::deregister_allocation(), and jeod::JeodTrickMemoryInterface↩ ::register_allocation().

6.3.3.4 trick_MM [3/4]

```
Trick::MemoryManager* trick_MM
```

6.3.3.5 trick_MM [4/4]

```
Trick::MemoryManager* trick_MM
```

Referenced by jeod::JeodTrick10MemoryInterface::JeodTrick10MemoryInterface().

Chapter 7

Namespace Documentation

7.1 er7_utils Namespace Reference

Namespace er7_utils contains the state integration models used by JEOD.

7.1.1 Detailed Description

Namespace er7_utils contains the state integration models used by JEOD.

7.2 jeod Namespace Reference

Namespace jeod.

Data Structures

- class BasicJeodTrickSimInterface

  The BasicJeodTrickSimInterface implements the required capabilities of the generic JeodSimulationInterface in a Trick simulation environment.
- class CheckPointInputManager

  A CheckPointInputManager provides tools for reading a checkpoint file.
- class CheckPointOutputManager

  A CheckPointOutputManager provides the basic tools for writing a checkpoint file.
- class JeodDynbodyIntegrationLoop

  A Trick::IntegLoopScheduler that provides the ability to integrate a collection of Trick::SimObject instances over time, with the sim objects capable of being moved from one integration loop to another during run time.
- class JeodIntegratorInterface

  A JeodIntegratorInterface extends the ER7 IntegratorInterface with the concept of a pointer to the simulation engine's integration object.
- class JeodMemoryInterface

  Abstract interface between the JEOD memory manager and the simulation engine.
- class JeodSimulationInterface

This abstract class defines the basis for the interface between JEOD and a simulation engine.

- class JeodSimulationInterfaceInit

  Define configuration data needed to configure the dynamically-created message handler and memory manager.
- class JeodTrick10MemoryInterface

  A TrickMemoryInterface implements the two required methods needed to register and deregister memory with the simulation engine, Trick in this case.
- class JeodTrickIntegrator

  A JeodTrickIntegrator specializes the JeodIntegratorInterface for use with Trick as the simulation engine.
- class JeodTrickMemoryInterface

  A TrickMemoryInterface implements the two required methods needed to register and deregister memory with the simulation engine, Trick in this case.
- class JeodTrickSimInterface

  A JEODTrickSimInterface implements the required capabilities of the generic JeodSimulationInterface in a Trick simulation environment.
- class SectionedInputBuffer

  A SectionedInputBuffer is a std::streambuf that reads from a section in a checkpoint file.
- class SectionedInputStream

  A SectionedInputStream is a std::istream that reads from a section in a checkpoint file.
- class SectionedOutputBuffer

  A SectionedOutputBuffer is a std::streambuf that writes a section of a checkpoint file.
- class SectionedOutputStream

  A SectionedOutputStream is a std::ostream that writes a section of a checkpoint file.
- class SimInterfaceMessages

  Specifies the message IDs used in the sim_interface model.
- class TrickJeodIntegrator

  A TrickJeodIntegrator specializes the Trick::Integrator to provide the Trick side of the integration interface between Trick and JEOD.
- class TrickMessageHandler

  The MessageHandler class for designed for use in Trick-based simulations.
- class TrickMessageHandlerMixin

  The TrickMessageHandlerMixin implements the required capabilities of the generic JeodSimulationInterface in a Trick simulation environment.

7.2.1 Detailed Description

Namespace jeod.

7.3 Trick Namespace Reference

Namespace Trick furnishes several standard functions for use in the Trick environment.

7.3.1 Detailed Description

Namespace Trick furnishes several standard functions for use in the Trick environment.

Chapter 8

Data Structure Documentation

8.1 jeod::JeodTrickMemoryInterface::AllocationMapEntry Struct Reference

Describes a chunk of JEOD-allocated memory.

`#include <trick_memory_interface.hh>`

Public Member Functions

- AllocationMapEntry (const std::type_info &type_info, uint32_t nelem, bool arrayp)
  Construct an AllocationMapEntry object.

Data Fields

- const std::type_info & typeid_info
  Descriptor of the data type.
- uint32_t nelements
  The number of elements in the allocated chunk of memory.
- bool is_array
  Is the item an array or a single object?

8.1.1 Detailed Description

Describes a chunk of JEOD-allocated memory.

Definition at line 295 of file trick_memory_interface.hh.

8.1.2 Constructor & Destructor Documentation

8.1.2.1 AllocationMapEntry()

```
jeod::JeodTrickMemoryInterface::AllocationMapEntry::AllocationMapEntry (
            const std::type_info & type_info,
            uint32_t nelem,
            bool arrayp )  [inline]
```

Construct an AllocationMapEntry object.

Parameters

type_info	Type info
nelem	Array size
arrayp	Is item an array?

Definition at line 318 of file trick_memory_interface.hh.

8.1.3 Field Documentation

8.1.3.1 is_array

`bool jeod::JeodTrickMemoryInterface::AllocationMapEntry::is_array`

Is the item an array or a single object?

trick_units(–)

Definition at line 310 of file trick_memory_interface.hh.

Referenced by jeod::JeodTrick10MemoryInterface::checkpoint_allocations().

8.1.3.2 nelements

`uint32_t jeod::JeodTrickMemoryInterface::AllocationMapEntry::nelements`

The number of elements in the allocated chunk of memory.

trick_units(–)

Definition at line 305 of file trick_memory_interface.hh.

Referenced by jeod::JeodTrick10MemoryInterface::checkpoint_allocations().

8.1.3.3 typeid_info

`const std::type_info& jeod::JeodTrickMemoryInterface::AllocationMapEntry::typeid_info`

Descriptor of the data type.

trick_units(–)

Definition at line 300 of file trick_memory_interface.hh.

Referenced by jeod::JeodTrick10MemoryInterface::checkpoint_allocations().

The documentation for this struct was generated from the following file:

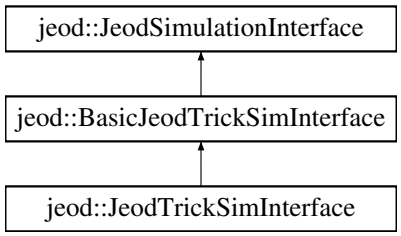
- trick_memory_interface.hh

8.2 jeod::BasicJeodTrickSimInterface Class Reference

The BasicJeodTrickSimInterface implements the required capabilities of the generic JeodSimulationInterface in a Trick simulation environment.

```
#include <trick_sim_interface.hh>
```

Inheritance diagram for jeod::BasicJeodTrickSimInterface:



Public Member Functions

- BasicJeodTrickSimInterface (MessageHandler &message_handler)

  Construct a BasicJeodTrickSimInterface object.
- virtual ~BasicJeodTrickSimInterface ()

  Destroy a BasicJeodTrickSimInterface object.
- void set_checkpoint_file_name (std::string name)

  Set the checkpoint file name.
- std::string get_checkpoint_file_name () const

  Get the checkpoint file name.
- virtual void set_mode (JeodSimulationInterface::Mode new_mode)

  Set the mode.
- void checkpoint_allocations (void)

  Dump the allocation information to the checkpoint file.
- void restore_allocations (void)

  Restore the allocated data per the checkpoint file.
- void checkpoint_containers (void)

  Dump the container objects to the checkpoint file.
- void restore_containers (void)

  Restore the container objects from the checkpoint file.
- void open_checkpoint_file (void)

  Open the checkpoint output file.
- void close_checkpoint_file (void)

  Close the checkpoint output file.
- void open_restart_file (void)

  Open the checkpoint input file.
- void close_restart_file (void)

  Close the checkpoint input file.

Protected Member Functions

- virtual JeodIntegratorInterface * create_integrator_internal (void)

  Create an integration interface object.
- virtual double get_job_cycle_internal (void)

  Get the current job's cycle time.
- virtual JeodMemoryInterface & get_memory_interface_internal (void)

  Get the memory interface.
- virtual SectionedInputStream get_checkpoint_reader_internal (const std::string &section_id)

  Get a reader to a section of the currently open checkpoint file.
- virtual SectionedOutputStream get_checkpoint_writer_internal (const std::string &section_id)

  Get a writer to a section of the currently open checkpoint file.

Protected Attributes

- MessageHandler & generic_message_handler

  The global MessageHandler.
- JeodTrick10MemoryInterface trick_memory_interface

  The interface between JEOD and Trick's memory management schemes.
- JeodMemoryManager memory_manager

  The global JEOD memory manager.
- std::string checkpoint_file_name

  The name of the segmented checkpoint file used for the next checkpoint / restart action.
- std::string section_start

  String indicating the start of a checkpoint file section.
- std::string section_end

  String indicating the end of a checkpoint file section.
- CheckPointInputManager * checkpoint_reader

  The object that manages reading from a checkpoint file.
- CheckPointOutputManager * checkpoint_writer

  The object that manages writing to a checkpoint file.

Private Member Functions

- BasicJeodTrickSimInterface (const BasicJeodTrickSimInterface &)

  Not implemented.
- BasicJeodTrickSimInterface & operator= (const BasicJeodTrickSimInterface &)

  Not implemented.

Friends

- class InputProcessor
- void init_attrjeod__BasicJeodTrickSimInterface ()

Additional Inherited Members

8.2.1 Detailed Description

The BasicJeodTrickSimInterface implements the required capabilities of the generic JeodSimulationInterface in a Trick simulation environment.

By virtue of member data ownership, the class creates the requisite MessageHandler and MemoryManager and does so in the correct order.

Definition at line 89 of file trick_sim_interface.hh.

8.2.2 Constructor & Destructor Documentation

8.2.2.1 BasicJeodTrickSimInterface() [1/2]

```
jeod::BasicJeodTrickSimInterface::BasicJeodTrickSimInterface (
            MessageHandler & message_handler )  [explicit]
```

Construct a BasicJeodTrickSimInterface object.

Parameters

in,out	*message_handler*	handler Units: Message

Definition at line 62 of file trick_sim_interface.cc.

References generic_message_handler, section_end, and section_start.

8.2.2.2 ~BasicJeodTrickSimInterface()

```
jeod::BasicJeodTrickSimInterface::~BasicJeodTrickSimInterface (
            void  )  [virtual]
```

Destroy a BasicJeodTrickSimInterface object.

Definition at line 98 of file trick_sim_interface.cc.

References checkpoint_reader, and checkpoint_writer.

8.2.2.3 BasicJeodTrickSimInterface() [2/2]

```
jeod::BasicJeodTrickSimInterface::BasicJeodTrickSimInterface (
            const BasicJeodTrickSimInterface &  )  [private]
```

Not implemented.

8.2.3 Member Function Documentation

8.2.3.1 checkpoint_allocations()

```
void jeod::BasicJeodTrickSimInterface::checkpoint_allocations (
            void  )
```

Dump the allocation information to the checkpoint file.

Definition at line 321 of file trick_sim_interface.cc.

References jeod::JeodTrick10MemoryInterface::checkpoint_allocations(), jeod::JeodTrick10MemoryInterface::is_↩checkpoint_restart_supported(), and trick_memory_interface.

8.2.3.2 checkpoint_containers()

```
void jeod::BasicJeodTrickSimInterface::checkpoint_containers (
            void  )
```

Dump the container objects to the checkpoint file.

Definition at line 347 of file trick_sim_interface.cc.

References jeod::JeodTrick10MemoryInterface::checkpoint_containers(), jeod::JeodTrick10MemoryInterface::is_↩checkpoint_restart_supported(), and trick_memory_interface.

8.2.3.3 close_checkpoint_file()

```
void jeod::BasicJeodTrickSimInterface::close_checkpoint_file (
            void  )
```

Close the checkpoint output file.

Definition at line 234 of file trick_sim_interface.cc.

References checkpoint_writer.

8.2.3.4 close_restart_file()

```
void jeod::BasicJeodTrickSimInterface::close_restart_file (
            void  )
```

Close the checkpoint input file.

Definition at line 307 of file trick_sim_interface.cc.

References checkpoint_reader.

8.2.3.5 create_integrator_internal()

```
JeodIntegratorInterface * jeod::BasicJeodTrickSimInterface::create_integrator_internal (
            void  )  [protected], [virtual]
```

Create an integration interface object.

Returns

Integrator interface that encapsulates an sim engine integrator.

Implements jeod::JeodSimulationInterface.

Definition at line 140 of file trick_sim_interface.cc.

8.2.3.6 get_checkpoint_file_name()

```
std::string jeod::BasicJeodTrickSimInterface::get_checkpoint_file_name ( ) const  [inline]
```

Get the checkpoint file name.

Definition at line 110 of file trick_sim_interface.hh.

References checkpoint_file_name.

8.2.3.7 get_checkpoint_reader_internal()

```
SectionedInputStream jeod::BasicJeodTrickSimInterface::get_checkpoint_reader_internal (
            const std::string & section_id )  [protected], [virtual]
```

Get a reader to a section of the currently open checkpoint file.

Returns

Checkpoint reader

Parameters

in	*section_id*	Section name

Implements jeod::JeodSimulationInterface.

Definition at line 288 of file trick_sim_interface.cc.

References checkpoint_reader, jeod::CheckPointInputManager::create_section_reader(), and jeod::SimInterfaceMessages::phasing_error.

8.2.3.8 get_checkpoint_writer_internal()

```
SectionedOutputStream jeod::BasicJeodTrickSimInterface::get_checkpoint_writer_internal (
            const std::string & section_id )  [protected], [virtual]
```

Get a writer to a section of the currently open checkpoint file.

Returns

Checkpoint writer

Parameters

in	*section_id*	Section name

Implements jeod::JeodSimulationInterface.

Definition at line 215 of file trick_sim_interface.cc.

References checkpoint_writer, jeod::CheckPointOutputManager::create_section_writer(), and jeod::SimInterfaceMessages::phasing_error.

8.2.3.9 get_job_cycle_internal()

```
double jeod::BasicJeodTrickSimInterface::get_job_cycle_internal (
            void  )  [protected], [virtual]
```

Get the current job's cycle time.

Returns

Current job's cycle time
Units: s

Implements jeod::JeodSimulationInterface.

Definition at line 152 of file trick_sim_interface.cc.

8.2.3.10 get_memory_interface_internal()

```
JeodMemoryInterface & jeod::BasicJeodTrickSimInterface::get_memory_interface_internal (
            void ) [protected], [virtual]
```

Get the memory interface.

Returns

Memory interface

Implements jeod::JeodSimulationInterface.

Definition at line 164 of file trick_sim_interface.cc.

References trick_memory_interface.

8.2.3.11 open_checkpoint_file()

```
void jeod::BasicJeodTrickSimInterface::open_checkpoint_file (
            void )
```

Open the checkpoint output file.

Definition at line 175 of file trick_sim_interface.cc.

References checkpoint_file_name, checkpoint_writer, jeod::JeodTrick10MemoryInterface::get_trick_checkpoint↩_file(), jeod::JeodTrick10MemoryInterface::is_checkpoint_restart_supported(), section_end, section_start, and trick_memory_interface.

8.2.3.12 open_restart_file()

```
void jeod::BasicJeodTrickSimInterface::open_restart_file (
            void )
```

Open the checkpoint input file.

Definition at line 248 of file trick_sim_interface.cc.

References checkpoint_file_name, checkpoint_reader, jeod::JeodTrick10MemoryInterface::get_trick_checkpoint↩_file(), jeod::JeodTrick10MemoryInterface::is_checkpoint_restart_supported(), section_end, section_start, and trick_memory_interface.

8.2.3.13 operator=()

```
BasicJeodTrickSimInterface& jeod::BasicJeodTrickSimInterface::operator= (
            const BasicJeodTrickSimInterface &  )  [private]
```

Not implemented.

8.2.3.14 restore_allocations()

```
void jeod::BasicJeodTrickSimInterface::restore_allocations (
            void  )
```

Restore the allocated data per the checkpoint file.

Definition at line 334 of file trick_sim_interface.cc.

References jeod::JeodTrick10MemoryInterface::is_checkpoint_restart_supported(), memory_manager, jeod::↩ JeodTrick10MemoryInterface::restore_allocations(), and trick_memory_interface.

8.2.3.15 restore_containers()

```
void jeod::BasicJeodTrickSimInterface::restore_containers (
            void  )
```

Restore the container objects from the checkpoint file.

Definition at line 360 of file trick_sim_interface.cc.

References jeod::JeodTrick10MemoryInterface::is_checkpoint_restart_supported(), jeod::JeodTrick10Memory↩ Interface::restore_containers(), and trick_memory_interface.

8.2.3.16 set_checkpoint_file_name()

```
void jeod::BasicJeodTrickSimInterface::set_checkpoint_file_name (
            std::string name )  [inline]
```

Set the checkpoint file name.

Definition at line 104 of file trick_sim_interface.hh.

References checkpoint_file_name.

8.2.3.17 set_mode()

```
void jeod::BasicJeodTrickSimInterface::set_mode (
            JeodSimulationInterface::Mode new_mode )  [virtual]
```

Set the mode.

Assumptions and Limitations

- See SimulationInterface::set_mode.

Parameters

in	*new_mode*	New mode.

Reimplemented from jeod::JeodSimulationInterface.

Definition at line 121 of file trick_sim_interface.cc.

References jeod::JeodSimulationInterface::get_mode(), memory_manager, jeod::JeodTrickMemoryInterface::set_mode(), jeod::JeodSimulationInterface::set_mode(), and trick_memory_interface.

8.2.4 Friends And Related Function Documentation

8.2.4.1 init_attrjeod__BasicJeodTrickSimInterface

```
void init_attrjeod__BasicJeodTrickSimInterface ( )  [friend]
```

8.2.4.2 InputProcessor

```
friend class InputProcessor  [friend]
```

Definition at line 90 of file trick_sim_interface.hh.

8.2.5 Field Documentation

8.2.5.1 checkpoint_file_name

```
std::string jeod::BasicJeodTrickSimInterface::checkpoint_file_name  [protected]
```

The name of the segmented checkpoint file used for the next checkpoint / restart action.

If the name is the empty string (default), the checkpoint / restart mechanisms will attempt to construct a name from the corresponding Trick checkpoint file name.trick_units(–)

Definition at line 200 of file trick_sim_interface.hh.

Referenced by get_checkpoint_file_name(), open_checkpoint_file(), open_restart_file(), and set_checkpoint_file_name().

8.2.5.2 checkpoint_reader

```
CheckPointInputManager* jeod::BasicJeodTrickSimInterface::checkpoint_reader  [protected]
```

The object that manages reading from a checkpoint file.

trick_io(**)

Definition at line 215 of file trick_sim_interface.hh.

Referenced by close_restart_file(), get_checkpoint_reader_internal(), open_restart_file(), and ~BasicJeodTrick↩SimInterface().

8.2.5.3 checkpoint_writer

```
CheckPointOutputManager* jeod::BasicJeodTrickSimInterface::checkpoint_writer  [protected]
```

The object that manages writing to a checkpoint file.

trick_io(**)

Definition at line 220 of file trick_sim_interface.hh.

Referenced by close_checkpoint_file(), get_checkpoint_writer_internal(), open_checkpoint_file(), and ~Basic↩JeodTrickSimInterface().

8.2.5.4 generic_message_handler

```
MessageHandler& jeod::BasicJeodTrickSimInterface::generic_message_handler  [protected]
```

The global MessageHandler.

trick_units(–)

Definition at line 181 of file trick_sim_interface.hh.

Referenced by BasicJeodTrickSimInterface().

8.2.5.5 memory_manager

```
JeodMemoryManager jeod::BasicJeodTrickSimInterface::memory_manager  [protected]
```

The global JEOD memory manager.

trick_units(–)

Definition at line 191 of file trick_sim_interface.hh.

Referenced by restore_allocations(), and set_mode().

8.2.5.6 section_end

```
std::string jeod::BasicJeodTrickSimInterface::section_end  [protected]
```

String indicating the end of a checkpoint file section.

trick_io(*o) trick_units(–)

Definition at line 210 of file trick_sim_interface.hh.

Referenced by BasicJeodTrickSimInterface(), open_checkpoint_file(), and open_restart_file().

8.2.5.7 section_start

```
std::string jeod::BasicJeodTrickSimInterface::section_start  [protected]
```

String indicating the start of a checkpoint file section.

trick_io(*o) trick_units(–)

Definition at line 205 of file trick_sim_interface.hh.

Referenced by BasicJeodTrickSimInterface(), open_checkpoint_file(), and open_restart_file().

8.2.5.8 trick_memory_interface

```
JeodTrick10MemoryInterface jeod::BasicJeodTrickSimInterface::trick_memory_interface  [protected]
```

The interface between JEOD and Trick's memory management schemes.

trick_units(–)

Definition at line 186 of file trick_sim_interface.hh.

Referenced by checkpoint_allocations(), checkpoint_containers(), get_memory_interface_internal(), open_checkpoint_file(), open_restart_file(), restore_allocations(), restore_containers(), and set_mode().

The documentation for this class was generated from the following files:

- trick_sim_interface.hh
- trick_sim_interface.cc

8.3 jeod::CheckPointInputManager Class Reference

A CheckPointInputManager provides tools for reading a checkpoint file.

```
#include <checkpoint_input_manager.hh>
```

Data Structures

- struct SectionInfo

  A SectionInfo contains the start and end positions of a checkpoint file section.

Public Member Functions

- CheckPointInputManager (const std::string &fname, const std::string &start_marker, const std::string &end_marker)

  Construct a CheckPointInputManager object.
- SectionedInputStream create_section_reader (const std::string &tag)

  Create a C++ input stream that reads from a checkpoint file section.
- bool operator! () const

  Conversion to boolean.
- bool have_active_reader () const

  Is there an active checkpoint section reader?
- bool register_reader (SectionedInputStream *reader)

  Register the supplied section reader as the currently-active reader.
- bool deregister_reader (SectionedInputStream *reader)

  Deregister the supplied section reader as the currently-active reader.

Private Member Functions

- void initialize (void)

  Determine the locations of the various sections that comprise the file.
- SectionedInputStream create_section_reader (bool trick, const std::string &tag)

  Create a C++ input stream that reads from a checkpoint file section.
- SectionedInputStream create_trick_section_reader ()

  Create a C++ input stream that reads from the Trick checkpoint file section.
- CheckPointInputManager (const CheckPointInputManager &)

  Not implemented.
- CheckPointInputManager & operator= (const CheckPointInputManager &)

  Not implemented.

Private Attributes

- std::map< std::string, SectionInfo > sections

  Maps section names to section start/end positions.
- std::ifstream stream

  The C++ file stream that reads the checkpoint file.
- SectionedInputStream * current_reader

  The reader that currently is active.
- const std::string filename

  The name of the checkpoint file.
- const std::string & section_start

  The string that indicates the start of a checkpoint file section.
- const std::string & section_end

  The string that indicates the start of a checkpoint file section.
- bool is_open

  Is the checkpoint file open?

8.3.1 Detailed Description

A CheckPointInputManager provides tools for reading a checkpoint file.

A Trick 10 checkpoint file comprises multiple sections delineated by section markers. This class recognizes those markers and generates C++ input streams that other objects can use to read the contents of one of those checkpoint file sections. The interpretation of the contents of a checkpoint file section is the responsibility of those other objects.

Definition at line 419 of file checkpoint_input_manager.hh.

8.3.2 Constructor & Destructor Documentation

8.3.2.1 CheckPointInputManager() [1/2]

```
jeod::CheckPointInputManager::CheckPointInputManager (
            const std::string & fname,
            const std::string & start_marker,
            const std::string & end_marker )
```

Construct a CheckPointInputManager object.

Parameters

in	*fname*	Name of file to be opened
in	*start_marker*	Start of section marker
in	*end_marker*	End of section marker

Definition at line 317 of file checkpoint_input_manager.cc.

References filename, jeod::SimInterfaceMessages::implementation_error, initialize(), is_open, and stream.

8.3.2.2 CheckPointInputManager() [2/2]

```
jeod::CheckPointInputManager::CheckPointInputManager (
            const CheckPointInputManager &  )  [private]
```

Not implemented.

8.3.3 Member Function Documentation

8.3.3.1 create_section_reader() [1/2]

```
SectionedInputStream jeod::CheckPointInputManager::create_section_reader (
            const std::string & tag )  [inline]
```

Create a C++ input stream that reads from a checkpoint file section.

Error handling

A null SectionedInputStream is created when the CheckPointInputManager itself is invalid or when the designated section is not present in the checkpoint file.

Parameters

tag	Tag that identifies the section to be read.

Returns

A SectionedInputStream object, which must be used to initialize a local SectionedInputStream variable.

Definition at line 438 of file checkpoint_input_manager.hh.

Referenced by create_trick_section_reader(), and jeod::BasicJeodTrickSimInterface::get_checkpoint_reader_internal().

8.3.3.2 create_section_reader() [2/2]

```
SectionedInputStream jeod::CheckPointInputManager::create_section_reader (
            bool trick,
            const std::string & tag )  [private]
```

Create a C++ input stream that reads from a checkpoint file section.

Usage

Use this function as the initializer of a section reader variable.

Error handling

A null SectionedInputStream is created when the CheckPointInputManager itself is invalid or when the designated section is not present in the checkpoint file.

Returns

A SectionedInputStream object.

Parameters

in	*trick*	OK to create the Trick section reader?
in	*tag*	Tag identifying the section to be read.

Definition at line 427 of file checkpoint_input_manager.cc.

References jeod::CheckPointInputManager::SectionInfo::end_pos, filename, jeod::SimInterfaceMessages↩ ::implementation_error, is_open, sections, jeod::CheckPointInputManager::SectionInfo::start_pos, and stream.

8.3.3.3 create_trick_section_reader()

```
SectionedInputStream jeod::CheckPointInputManager::create_trick_section_reader (
            void ) [private]
```

Create a C++ input stream that reads from the Trick checkpoint file section.

Returns

Trick SectionedInputStream object.

Definition at line 470 of file checkpoint_input_manager.cc.

References create_section_reader(), current_reader, and jeod::SectionedInputStream::deactivate().

8.3.3.4 deregister_reader()

```
bool jeod::CheckPointInputManager::deregister_reader (
            SectionedInputStream * reader )
```

Deregister the supplied section reader as the currently-active reader.

Returns

True => success.

Parameters

in	*reader*	Reader to be deregistered

Definition at line 507 of file checkpoint_input_manager.cc.

References current_reader.

Referenced by jeod::SectionedInputStream::deactivate(), and jeod::SectionedInputStream::~SectionedInput↩ Stream().

8.3.3.5 have_active_reader()

```
bool jeod::CheckPointInputManager::have_active_reader ( ) const  [inline]
```

Is there an active checkpoint section reader?

Returns

True if there is an active reader, false otherwise.

Definition at line 454 of file checkpoint_input_manager.hh.

References current_reader.

Referenced by jeod::SectionedInputStream::is_activatable().

8.3.3.6 initialize()

```
void jeod::CheckPointInputManager::initialize (
            void  )  [private]
```

Determine the locations of the various sections that comprise the file.

Definition at line 347 of file checkpoint_input_manager.cc.

References filename, jeod::SimInterfaceMessages::implementation_error, section_end, section_start, sections, and stream.

Referenced by CheckPointInputManager().

8.3.3.7 operator"!()

```
bool jeod::CheckPointInputManager::operator!  ( ) const  [inline]
```

Conversion to boolean.

Returns

False if object is OK.

Definition at line 446 of file checkpoint_input_manager.hh.

References is_open, and stream.

8.3.3.8 operator=()

```
CheckPointInputManager& jeod::CheckPointInputManager::operator= (
            const CheckPointInputManager &  )  [private]
```

Not implemented.

8.3.3.9 register_reader()

```
bool jeod::CheckPointInputManager::register_reader (
            SectionedInputStream * reader )
```

Register the supplied section reader as the currently-active reader.

Returns

True => success.

Parameters

in	*reader*	Reader to be registered

Definition at line 488 of file checkpoint_input_manager.cc.

References current_reader.

Referenced by jeod::SectionedInputStream::activate().

8.3.4 Field Documentation

8.3.4.1 current_reader

```
SectionedInputStream* jeod::CheckPointInputManager::current_reader  [private]
```

The reader that currently is active.

trick_io(**)

Definition at line 524 of file checkpoint_input_manager.hh.

Referenced by create_trick_section_reader(), deregister_reader(), have_active_reader(), and register_reader().

8.3.4.2 filename

```
const std::string jeod::CheckPointInputManager::filename  [private]
```

The name of the checkpoint file.

Definition at line 529 of file checkpoint_input_manager.hh.

Referenced by CheckPointInputManager(), create_section_reader(), and initialize().

8.3.4.3 is_open

```
bool jeod::CheckPointInputManager::is_open  [private]
```

Is the checkpoint file open?

trick_io(**)

Definition at line 544 of file checkpoint_input_manager.hh.

Referenced by CheckPointInputManager(), create_section_reader(), and operator!().

8.3.4.4 section_end

```
const std::string& jeod::CheckPointInputManager::section_end  [private]
```

The string that indicates the start of a checkpoint file section.

Definition at line 539 of file checkpoint_input_manager.hh.

Referenced by initialize().

8.3.4.5 section_start

```
const std::string& jeod::CheckPointInputManager::section_start  [private]
```

The string that indicates the start of a checkpoint file section.

Definition at line 534 of file checkpoint_input_manager.hh.

Referenced by initialize().

8.3.4.6 sections

```
std::map<std::string, SectionInfo> jeod::CheckPointInputManager::sections  [private]
```

Maps section names to section start/end positions.

trick_io(**)

Definition at line 514 of file checkpoint_input_manager.hh.

Referenced by create_section_reader(), and initialize().

8.3.4.7 stream

```
std::ifstream jeod::CheckPointInputManager::stream  [private]
```

The C++ file stream that reads the checkpoint file.

trick_io(**)

Definition at line 519 of file checkpoint_input_manager.hh.

Referenced by CheckPointInputManager(), create_section_reader(), initialize(), and operator!().

The documentation for this class was generated from the following files:

- checkpoint_input_manager.hh
- checkpoint_input_manager.cc

8.4 jeod::CheckPointOutputManager Class Reference

A CheckPointOutputManager provides the basic tools for writing a checkpoint file.

```
#include <checkpoint_output_manager.hh>
```

Public Member Functions

- CheckPointOutputManager (const std::string &fname, const std::string &start_marker, const std::string &end_marker)

  Construct a CheckPointOutputManager object.
- SectionedOutputStream create_section_writer (const std::string &tag)

  Create a C++ output stream that writes a checkpoint file section.
- bool operator! () const

  Conversion to boolean.
- bool have_active_writer () const

  Is there an active checkpoint section writer?
- bool register_writer (SectionedOutputStream *writer)

  Register the supplied section writer as the currently-active writer.
- bool deregister_writer (SectionedOutputStream *writer)

  Deregister the supplied section writer as the currently-active writer.

Private Member Functions

- SectionedOutputStream create_section_writer (bool trick, const std::string &tag)

  Create a C++ output stream that writes to a checkpoint file section.
- SectionedOutputStream create_trick_section_writer ()

  Create a C++ output stream that writes a Trick checkpoint file section.
- CheckPointOutputManager (const CheckPointOutputManager &)

  Not implemented.
- CheckPointOutputManager & operator= (const CheckPointOutputManager &)

  Not implemented.

Private Attributes

- std::ofstream stream

  The C++ file stream that writes to the checkpoint file.
- SectionedOutputStream * current_writer

  The writer that currently is active.
- const std::string filename

  The name of the checkpoint file.
- const std::string & section_start

  The string that indicates the start of a checkpoint file section.
- const std::string & section_end

  The string that indicates the start of a checkpoint file section.
- bool is_open

  Is the checkpoint file open?

Friends

- class MemoryManagerWrapper

8.4.1 Detailed Description

A CheckPointOutputManager provides the basic tools for writing a checkpoint file.

Section markers split a Trick 10 checkpoint file into multiple parts. This class generates C++ output streams that write the section markers and that other objects can use to write checkpoint file section data.

Definition at line 279 of file checkpoint_output_manager.hh.

8.4.2 Constructor & Destructor Documentation

8.4.2.1 CheckPointOutputManager() [1/2]

```
jeod::CheckPointOutputManager::CheckPointOutputManager (
            const std::string & fname,
            const std::string & start_marker,
            const std::string & end_marker )
```

Construct a CheckPointOutputManager object.

Parameters

in	*fname*	Name of file to be opened
in	*start_marker*	Start of section marker
in	*end_marker*	End of section marker

Definition at line 321 of file checkpoint_output_manager.cc.

References filename, jeod::SimInterfaceMessages::implementation_error, is_open, and stream.

8.4.2.2 CheckPointOutputManager() [2/2]

```
jeod::CheckPointOutputManager::CheckPointOutputManager (
            const CheckPointOutputManager &  )  [private]
```

Not implemented.

8.4.3 Member Function Documentation

8.4.3.1 create_section_writer() [1/2]

```
SectionedOutputStream jeod::CheckPointOutputManager::create_section_writer (
            const std::string & tag )  [inline]
```

Create a C++ output stream that writes a checkpoint file section.

Returns

Constructed SectionedOutputStream.

Definition at line 293 of file checkpoint_output_manager.hh.

Referenced by create_trick_section_writer(), and jeod::BasicJeodTrickSimInterface::get_checkpoint_writer_internal().

8.4.3.2 create_section_writer() [2/2]

```
SectionedOutputStream jeod::CheckPointOutputManager::create_section_writer (
            bool trick,
            const std::string & tag )  [private]
```

Create a C++ output stream that writes to a checkpoint file section.

Usage

Use this function as the initializer of a section writer variable.

Error handling

A null SectionedOutputStream is created when the CheckPointOutputManager itself is invalid or the designated section is invalid.

Returns

A SectionedOutputStream object.

Parameters

in	*trick*	OK to create the Trick section writer?
in	*tag*	Tag identifying the section to be written.

Definition at line 355 of file checkpoint_output_manager.cc.

References filename, jeod::SimInterfaceMessages::implementation_error, is_open, section_end, section_start, and stream.

8.4.3.3 create_trick_section_writer()

```
SectionedOutputStream jeod::CheckPointOutputManager::create_trick_section_writer (
            void  )  [private]
```

Create a C++ output stream that writes a Trick checkpoint file section.

Create a C++ output stream that writes to the Trick checkpoint file section.

Returns

A SectionedOutputStream object, which must be used to initialize a local SectionedOutputStream variable. Trick SectionedOutputStream object.

Definition at line 388 of file checkpoint_output_manager.cc.

References create_section_writer(), current_writer, and jeod::SectionedOutputStream::deactivate().

8.4.3.4 deregister_writer()

```
bool jeod::CheckPointOutputManager::deregister_writer (
            SectionedOutputStream * writer )
```

Deregister the supplied section writer as the currently-active writer.

Returns

True => success.

Parameters

in	*writer*	Writer to be deregistered

Definition at line 425 of file checkpoint_output_manager.cc.

References current_writer.

Referenced by jeod::SectionedOutputStream::deactivate().

8.4.3.5 have_active_writer()

```
bool jeod::CheckPointOutputManager::have_active_writer ( ) const  [inline]
```

Is there an active checkpoint section writer?

Returns

True if there is an active writer, false otherwise.

Definition at line 309 of file checkpoint_output_manager.hh.

References current_writer.

Referenced by jeod::SectionedOutputStream::is_activatable().

8.4.3.6 operator"!()

```
bool jeod::CheckPointOutputManager::operator!  ( ) const  [inline]
```

Conversion to boolean.

Returns

False if object is OK.

Definition at line 301 of file checkpoint_output_manager.hh.

References is_open, and stream.

8.4.3.7 operator=()

```
CheckPointOutputManager& jeod::CheckPointOutputManager::operator= (
            const CheckPointOutputManager &  )  [private]
```

Not implemented.

8.4.3.8 register_writer()

```
bool jeod::CheckPointOutputManager::register_writer (
            SectionedOutputStream * writer )
```

Register the supplied section writer as the currently-active writer.

Returns

True => success.

Parameters

in	*writer*	Writer to be Registered

Definition at line 406 of file checkpoint_output_manager.cc.

References current_writer.

Referenced by jeod::SectionedOutputStream::activate().

8.4.4 Friends And Related Function Documentation

8.4.4.1 MemoryManagerWrapper

```
friend class MemoryManagerWrapper  [friend]
```

Definition at line 280 of file checkpoint_output_manager.hh.

8.4.5 Field Documentation

8.4.5.1 current_writer

```
SectionedOutputStream* jeod::CheckPointOutputManager::current_writer  [private]
```

The writer that currently is active.

trick_io(**)

Definition at line 345 of file checkpoint_output_manager.hh.

Referenced by create_trick_section_writer(), deregister_writer(), have_active_writer(), and register_writer().

8.4.5.2 filename

```
const std::string jeod::CheckPointOutputManager::filename  [private]
```

The name of the checkpoint file.

Definition at line 350 of file checkpoint_output_manager.hh.

Referenced by CheckPointOutputManager(), and create_section_writer().

8.4.5.3 is_open

```
bool jeod::CheckPointOutputManager::is_open  [private]
```

Is the checkpoint file open?

trick_io(**)

Definition at line 365 of file checkpoint_output_manager.hh.

Referenced by CheckPointOutputManager(), create_section_writer(), and operator!().

8.4.5.4 section_end

```
const std::string& jeod::CheckPointOutputManager::section_end  [private]
```

The string that indicates the start of a checkpoint file section.

Definition at line 360 of file checkpoint_output_manager.hh.

Referenced by create_section_writer().

8.4.5.5 section_start

```
const std::string& jeod::CheckPointOutputManager::section_start  [private]
```

The string that indicates the start of a checkpoint file section.

Definition at line 355 of file checkpoint_output_manager.hh.

Referenced by create_section_writer().

8.4.5.6 stream

```
std::ofstream jeod::CheckPointOutputManager::stream  [private]
```

The C++ file stream that writes to the checkpoint file.

trick_io(**)

Definition at line 340 of file checkpoint_output_manager.hh.

Referenced by CheckPointOutputManager(), create_section_writer(), and operator!().

The documentation for this class was generated from the following files:

- checkpoint_output_manager.hh
- checkpoint_output_manager.cc

8.5 jeod::JeodTrickMemoryInterface::ContainerListEntry Struct Reference

Describes a Checkpointable object.

```
#include <trick_memory_interface.hh>
```

Public Member Functions

- ContainerListEntry (const void *parent, const JeodMemoryTypeDescriptor &tdesc, const std::string &sub_id, JeodCheckpointable &obj)

  Construct an ContainerListEntry object.

Data Fields

- const void * owner

  The object that contains the container.
- const JeodMemoryTypeDescriptor & owner_type

  Type description of the object that contains the container.
- std::string elem_name

  The name of the element of the container in the owning object.
- JeodCheckpointable & container

  The container itself.

8.5.1 Detailed Description

Describes a Checkpointable object.

Definition at line 248 of file trick_memory_interface.hh.

8.5.2 Constructor & Destructor Documentation

8.5.2.1 ContainerListEntry()

```
jeod::JeodTrickMemoryInterface::ContainerListEntry::ContainerListEntry (
            const void * parent,
            const JeodMemoryTypeDescriptor & tdesc,
            const std::string & sub_id,
            JeodCheckpointable & obj )  [inline]
```

Construct an ContainerListEntry object.

Parameters

parent	Parent object
tdesc	Type descriptor
sub_id	Parent element
obj	Checkpointable itself

Definition at line 278 of file trick_memory_interface.hh.

8.5.3 Field Documentation

8.5.3.1 container

```
JeodCheckpointable& jeod::JeodTrickMemoryInterface::ContainerListEntry::container
```

The container itself.

trick_units(–)

Definition at line 268 of file trick_memory_interface.hh.

Referenced by jeod::JeodTrick10MemoryInterface::restore_containers().

8.5.3.2 elem_name

```
std::string jeod::JeodTrickMemoryInterface::ContainerListEntry::elem_name
```

The name of the element of the container in the owning object.

trick_units(–)

Definition at line 263 of file trick_memory_interface.hh.

Referenced by jeod::JeodTrick10MemoryInterface::deregister_container(), jeod::JeodTrick10MemoryInterface::get_container_id(), and jeod::JeodTrick10MemoryInterface::register_container().

8.5.3.3 owner

```
const void* jeod::JeodTrickMemoryInterface::ContainerListEntry::owner
```

The object that contains the container.

trick_units(–)

Definition at line 253 of file trick_memory_interface.hh.

Referenced by jeod::JeodTrick10MemoryInterface::deregister_container(), jeod::JeodTrick10MemoryInterface::get_container_id(), and jeod::JeodTrick10MemoryInterface::register_container().

8.5.3.4 owner_type

```
const JeodMemoryTypeDescriptor& jeod::JeodTrickMemoryInterface::ContainerListEntry::owner_type
```

Type description of the object that contains the container.

trick_units(–)

Definition at line 258 of file trick_memory_interface.hh.

Referenced by jeod::JeodTrick10MemoryInterface::deregister_container(), jeod::JeodTrick10MemoryInterface::get_container_id(), and jeod::JeodTrick10MemoryInterface::register_container().

The documentation for this struct was generated from the following file:

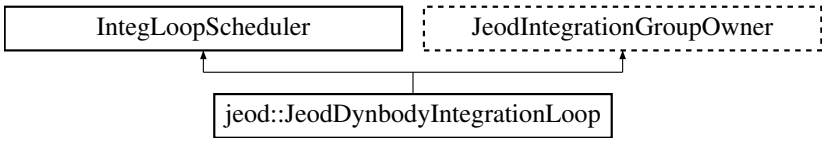
- trick_memory_interface.hh

8.6 jeod::JeodDynbodyIntegrationLoop Class Reference

A Trick::IntegLoopScheduler that provides the ability to integrate a collection of Trick::SimObject instances over time, with the sim objects capable of being moved from one integration loop to another during run time.

`#include <trick_dynbody_integ_loop.hh>`

Inheritance diagram for jeod::JeodDynbodyIntegrationLoop:



Public Member Functions

- JeodDynbodyIntegrationLoop ()

  JeodDynbodyIntegrationLoop default constructor.
- JeodDynbodyIntegrationLoop (double cycle, Trick::SimObject &sim_object_in, TimeManager &time_↩ manager_in, DynManager &dyn_manager_in, GravityManager &grav_manager_in, er7_utils::Integrator↩ Constructor ∗&integ_cotr_in, DynamicsIntegrationGroup &integ_group_factory)

  JeodDynbodyIntegrationLoop non-default constructor.
- virtual ∼JeodDynbodyIntegrationLoop ()

  JeodDynbodyIntegrationLoop destructor.
- void initialize_integ_loop (void)

  S_define-level function to initialize the integration loop.
- void set_time_to_loop_start ()

  S_define-level function to reset JEOD time to the time at the start of the current integration loop.
- virtual void update_integration_group (JeodIntegrationGroup &group)

  Update the provided integration group, which must be the integration group contained within this integration loop object.
- virtual int add_sim_object (Trick::SimObject &sim_obj)

  Add a sim object to the set of objects to be integrated by this integration loop object.
- virtual void add_integrable_object (er7_utils::IntegrableObject &integrable_object)

  Add the specified integrable object, which should not be a DynBody, to the integration group's set of integrable objects.
- virtual int remove_sim_object (Trick::SimObject &sim_obj)

  Remove a sim object from the set of objects to be integrated by this integration loop object.
- virtual void remove_integrable_object (er7_utils::IntegrableObject &integrable_object)

  Remove the specified integrable object from the integration group's set of integrable objects.
- virtual void gravitation (void)

  Compute the gravitational accelerations of each dynamic body that is integrated by this integration loop.
- virtual void collect_derivatives (void)

  Collect the derivatives for each dynamic body that is integrated by this integration loop.
- virtual void set_deriv_ephem_update (bool val)

  Set the deriv_ephem_update flag for the integration group.

Protected Member Functions

- Trick::SimObject * find_containing_sim_object (er7_utils::IntegrableObject &integrable_object)

  Find the sim object that contains the specified integrable object.
- virtual void add_sim_object_bodies (Trick::SimObject &sim_obj)

  Add the DynBody objects contained in the specified sim object to the set of DynBody objects integrated by this integration loop.
- virtual void add_sim_object_bodies (void)

  Add the dyn bodies contained in all the sim objects integrated by this integration loop to the loop's integration group.
- virtual void remove_sim_object_bodies (Trick::SimObject &sim_obj)

  Remove the DynBody objects contained in the specified sim object from the set of DynBody objects integrated by this integration loop.
- virtual int integrate_dt (double beg_sim_time, double del_sim_time)

  Integrate sim objects over the specified time span.

Protected Attributes

- Trick::SimObject * loop_sim_object

  The simulation object that contains this integration loop object.
- DynManager * dyn_manager

  The JEOD dynamics manager.
- TimeManager * time_manager

  The JEOD time manager.
- GravityManager * gravity_manager

  The gravity model manager.
- JeodTrickIntegrator integ_interface

  Dummy integration interface; needed by the integ_group.
- er7_utils::IntegratorConstructor ** integ_constructor

  Handle to the integration constructor used to create integrators.
- const DynamicsIntegrationGroup * integ_group_factory

  The externally-supplied integration group used as a template for creating this integration loop's integration group.
- DynamicsIntegrationGroup * integ_group

  The integration group that performs the integration.
- bool deriv_ephem_update

  If set, ephemerides will be updated at the derivative rate.

Private Member Functions

- JeodDynbodyIntegrationLoop (const JeodDynbodyIntegrationLoop &)

  < Deleted.
- JeodDynbodyIntegrationLoop & operator= (const JeodDynbodyIntegrationLoop &)

Friends

- class InputProcessor
- void init_attrjeod__JeodDynbodyIntegrationLoop ()

8.6.1 Detailed Description

A Trick::IntegLoopScheduler that provides the ability to integrate a collection of Trick::SimObject instances over time, with the sim objects capable of being moved from one integration loop to another during run time.

A JeodDynbodyIntegrationLoop augments this capability in a number of regards:

- All DynBody objects contained in the sim objects integrated by a JeodDynbodyIntegrationLoop object are integrated using JEOD integration.
- The DynBody objects to be integrated by a JeodDynbodyIntegrationLoop object are automatically collected as a member of the DynamicsIntegrationGroup object contained within a JeodDynbodyIntegrationLoop object.
- Non-DynBody integrable objects can also be integrated using JEOD integration.
- Non-DynBody integrable objects that are elsewhere identified as being associated with a DynBody object are automatically collected along with the DynBody objects with which they are associated.
- The DynBody and associated integrable objects are integrated using the DynamicsIntegrationGroup object contained in the loop object.

Users of this class are strongly encouraged to do so via a JeodIntegLoopSimObject. See $JEOD_HOME/lib/jeod/↩ JEOD_S_modules/integ_loop.sm.

Definition at line 133 of file trick_dynbody_integ_loop.hh.

8.6.2 Constructor & Destructor Documentation

8.6.2.1 JeodDynbodyIntegrationLoop() [1/3]

```
jeod::JeodDynbodyIntegrationLoop::JeodDynbodyIntegrationLoop ( )
```

JeodDynbodyIntegrationLoop default constructor.

Note

This exists only for the purpose of automated checkpoint/restart.

Warning

Do not use the default constructor outside of this context.

Definition at line 55 of file trick_dynbody_integ_loop.cc.

8.6.2.2 JeodDynbodyIntegrationLoop() [2/3]

```
jeod::JeodDynbodyIntegrationLoop::JeodDynbodyIntegrationLoop (
            double cycle,
            Trick::SimObject & sim_object_in,
            TimeManager & time_manager_in,
            DynManager & dyn_manager_in,
            GravityManager & grav_manager_in,
            er7_utils::IntegratorConstructor *& integ_cotr_in,
            DynamicsIntegrationGroup & integ_group_factory )
```

JeodDynbodyIntegrationLoop non-default constructor.

This is the constructor that should be used in the S_define file. The SimObject that contains this JeodDynbodyIntegrationLoop instance must register an "integ_loop" class job that calls the loop's integrate method.

Parameters

cycle	The integration interval in simulation seconds. This must be the same interval as specified in the integ_loop job specification.
sim_object_in	The SimObject that contains this JeodDynbodyIntegrationLoop instance.
time_manager_in	The simulation's time manager object.
dyn_manager_in	The simulation's dynamics manager object.
grav_manager_in	The simulation's gravity manager object.
integ_cotr_in	The integrator constructor used to create integration artifacts.
integ_group_factory	The integration group object used to create this loop's integ group.

Definition at line 71 of file trick_dynbody_integ_loop.cc.

References add_sim_object(), jeod::SimInterfaceMessages::integration_error, and loop_sim_object.

8.6.2.3 ~JeodDynbodyIntegrationLoop()

```
jeod::JeodDynbodyIntegrationLoop::~JeodDynbodyIntegrationLoop (
            void  )  [virtual]
```

JeodDynbodyIntegrationLoop destructor.

Definition at line 106 of file trick_dynbody_integ_loop.cc.

References integ_group.

8.6.2.4 JeodDynbodyIntegrationLoop() [3/3]

```
jeod::JeodDynbodyIntegrationLoop::JeodDynbodyIntegrationLoop (
            const JeodDynbodyIntegrationLoop &  )  [private]
```

< Deleted.

Deleted.

8.6.3 Member Function Documentation

8.6.3.1 add_integrable_object()

```
void jeod::JeodDynbodyIntegrationLoop::add_integrable_object (
            er7_utils::IntegrableObject & integrable_object )  [virtual]
```

Add the specified integrable object, which should not be a DynBody, to the integration group's set of integrable objects.

Parameters

integrable_object	Object to be added.

Definition at line 184 of file trick_dynbody_integ_loop.cc.

References integ_group, and jeod::SimInterfaceMessages::integration_error.

8.6.3.2 add_sim_object()

```
int jeod::JeodDynbodyIntegrationLoop::add_sim_object (
             Trick::SimObject & sim_obj )  [virtual]
```

Add a sim object to the set of objects to be integrated by this integration loop object.

The job queues for this loop are rebuilt after adding the sim object.

Parameters

sim_obj	The SimObject to be added to this loop object.

Returns

Zero => success, non-zero => error.

Definition at line 285 of file trick_dynbody_integ_loop.cc.

References add_sim_object_bodies(), and dyn_manager.

Referenced by JeodDynbodyIntegrationLoop().

8.6.3.3 add_sim_object_bodies() [1/2]

```
void jeod::JeodDynbodyIntegrationLoop::add_sim_object_bodies (
             Trick::SimObject & sim_obj )  [protected], [virtual]
```

Add the DynBody objects contained in the specified sim object to the set of DynBody objects integrated by this integration loop.

Parameters

sim_obj	The SimObject being added to this loop object.

Definition at line 329 of file trick_dynbody_integ_loop.cc.

References dyn_manager, find_containing_sim_object(), and integ_group.

8.6.3.4 add_sim_object_bodies() [2/2]

```
void jeod::JeodDynbodyIntegrationLoop::add_sim_object_bodies (
            void  )  [protected], [virtual]
```

Add the dyn bodies contained in all the sim objects integrated by this integration loop to the loop's integration group.

Definition at line 348 of file trick_dynbody_integ_loop.cc.

References dyn_manager, find_containing_sim_object(), and integ_group.

Referenced by add_sim_object(), and update_integration_group().

8.6.3.5 collect_derivatives()

```
virtual void jeod::JeodDynbodyIntegrationLoop::collect_derivatives (
            void  )  [inline], [virtual]
```

Collect the derivatives for each dynamic body that is integrated by this integration loop.

Definition at line 284 of file trick_dynbody_integ_loop.hh.

References integ_group.

8.6.3.6 find_containing_sim_object()

```
Trick::SimObject * jeod::JeodDynbodyIntegrationLoop::find_containing_sim_object (
            er7_utils::IntegrableObject & integrable_object )  [protected]
```

Find the sim object that contains the specified integrable object.

Parameters

integrable_object	Object to be found.

Returns

Sim object that contains the specified object, or null if none.

Definition at line 146 of file trick_dynbody_integ_loop.cc.

References jeod::JeodSimulationInterface::get_address_at_name(), and jeod::JeodSimulationInterface::get_↩ name_at_address().

Referenced by add_sim_object_bodies(), and remove_sim_object_bodies().

8.6.3.7 gravitation()

```
virtual void jeod::JeodDynbodyIntegrationLoop::gravitation (
            void  )  [inline], [virtual]
```

Compute the gravitational accelerations of each dynamic body that is integrated by this integration loop.

Definition at line 274 of file trick_dynbody_integ_loop.hh.

References dyn_manager, gravity_manager, and integ_group.

8.6.3.8 initialize_integ_loop()

```
void jeod::JeodDynbodyIntegrationLoop::initialize_integ_loop (
            void  )
```

S_define-level function to initialize the integration loop.

This function should be called as a very low phase integration class job.

Definition at line 117 of file trick_dynbody_integ_loop.cc.

References deriv_ephem_update, dyn_manager, integ_constructor, integ_group, integ_group_factory, integ_↩ interface, jeod::SimInterfaceMessages::integration_error, and time_manager.

8.6.3.9 integrate_dt()

```
int jeod::JeodDynbodyIntegrationLoop::integrate_dt (
            double beg_sim_time,
            double del_sim_time )  [protected], [virtual]
```

Integrate sim objects over the specified time span.

This is an overridable internal integration function and is called by the externally-visible integrate method and by call_dynamic_event_jobs.

Returns

Zero/non-zero success indicator. Out-of-sync integrators cause a non-zero return.

Parameters

beg_sim_time	The time at the start of the integration interval.
del_sim_time	The time span of the integration interval.

Definition at line 222 of file trick_dynbody_integ_loop.cc.

References integ_group, jeod::SimInterfaceMessages::integration_error, and trick_curr_integ.

8.6.3.10 operator=()

```
JeodDynbodyIntegrationLoop& jeod::JeodDynbodyIntegrationLoop::operator= (
            const JeodDynbodyIntegrationLoop &  )  [private]
```

8.6.3.11 remove_integrable_object()

```
void jeod::JeodDynbodyIntegrationLoop::remove_integrable_object (
            er7_utils::IntegrableObject & integrable_object )  [virtual]
```

Remove the specified integrable object from the integration group's set of integrable objects.

Parameters

integrable_object	Object to be removed.

Definition at line 205 of file trick_dynbody_integ_loop.cc.

References integ_group.

8.6.3.12 remove_sim_object()

```
int jeod::JeodDynbodyIntegrationLoop::remove_sim_object (
            Trick::SimObject & sim_obj )  [virtual]
```

Remove a sim object from the set of objects to be integrated by this integration loop object.

The job queues for this loop are rebuilt after removing the sim object.

Parameters

sim_obj	The SimObject to be removed from this loop object.

Returns

Zero => success, non-zero => error.

Definition at line 307 of file trick_dynbody_integ_loop.cc.

References dyn_manager, and remove_sim_object_bodies().

8.6.3.13 remove_sim_object_bodies()

```
void jeod::JeodDynbodyIntegrationLoop::remove_sim_object_bodies (
            Trick::SimObject & sim_obj )  [protected], [virtual]
```

Remove the DynBody objects contained in the specified sim object from the set of DynBody objects integrated by this integration loop.

Parameters

sim_obj	The SimObject being removed from this loop object.

Definition at line 369 of file trick_dynbody_integ_loop.cc.

References dyn_manager, find_containing_sim_object(), and integ_group.

Referenced by remove_sim_object().

8.6.3.14 set_deriv_ephem_update()

```
virtual void jeod::JeodDynbodyIntegrationLoop::set_deriv_ephem_update (
            bool val )  [inline], [virtual]
```

Set the deriv_ephem_update flag for the integration group.

Parameters

val	New value for deriv_ephem_update.

Definition at line 294 of file trick_dynbody_integ_loop.hh.

References deriv_ephem_update, and integ_group.

8.6.3.15 set_time_to_loop_start()

```
void jeod::JeodDynbodyIntegrationLoop::set_time_to_loop_start ( )
```

S_define-level function to reset JEOD time to the time at the start of the current integration loop.

This function should be called as a very low phase pre-integration class job in simulations that have multiple integration loops.

Definition at line 214 of file trick_dynbody_integ_loop.cc.

References time_manager.

8.6.3.16 update_integration_group()

```
void jeod::JeodDynbodyIntegrationLoop::update_integration_group (
             JeodIntegrationGroup & group )  [virtual]
```

Update the provided integration group, which must be the integration group contained within this integration loop object.

Note

This function is public because it is called (indirectly) from DynManager::initialize_simulation. It should otherwise be viewed as a protected or private function.

Parameters

group	The IntegrationGroup to be updated, which must be the integration loop's integration group object.

Definition at line 388 of file trick_dynbody_integ_loop.cc.

References add_sim_object_bodies(), integ_group, and jeod::SimInterfaceMessages::integration_error.

8.6.4 Friends And Related Function Documentation

8.6.4.1 init_attrjeod__JeodDynbodyIntegrationLoop

```
void init_attrjeod__JeodDynbodyIntegrationLoop ( )  [friend]
```

8.6.4.2 InputProcessor

```
friend class InputProcessor  [friend]
```

Definition at line 138 of file trick_dynbody_integ_loop.hh.

8.6.5 Field Documentation

8.6.5.1 deriv_ephem_update

```
bool jeod::JeodDynbodyIntegrationLoop::deriv_ephem_update  [protected]
```

If set, ephemerides will be updated at the derivative rate.

If clear, ephemerides will not be updated at the derivative rate by the ephemerides manager. Derivative-rate updates can still be attained by explicitly calling the various ephemerides model's update functions as derivative class jobs.↩ trick_units(–)

Definition at line 404 of file trick_dynbody_integ_loop.hh.

Referenced by initialize_integ_loop(), and set_deriv_ephem_update().

8.6.5.2 dyn_manager

```
DynManager* jeod::JeodDynbodyIntegrationLoop::dyn_manager  [protected]
```

The JEOD dynamics manager.

trick_units(–)

Definition at line 364 of file trick_dynbody_integ_loop.hh.

Referenced by add_sim_object(), add_sim_object_bodies(), gravitation(), initialize_integ_loop(), remove_sim_↩ object(), and remove_sim_object_bodies().

8.6.5.3 gravity_manager

```
GravityManager* jeod::JeodDynbodyIntegrationLoop::gravity_manager  [protected]
```

The gravity model manager.

trick_units(–)

Definition at line 374 of file trick_dynbody_integ_loop.hh.

Referenced by gravitation().

8.6.5.4 integ_constructor

```
er7_utils::IntegratorConstructor** jeod::JeodDynbodyIntegrationLoop::integ_constructor  [protected]
```

Handle to the integration constructor used to create integrators.

trick_units(–)

Definition at line 384 of file trick_dynbody_integ_loop.hh.

Referenced by initialize_integ_loop().

8.6.5.5 integ_group

```
DynamicsIntegrationGroup* jeod::JeodDynbodyIntegrationLoop::integ_group  [protected]
```

The integration group that performs the integration.

trick_units(–)

Definition at line 395 of file trick_dynbody_integ_loop.hh.

Referenced by add_integrable_object(), add_sim_object_bodies(), collect_derivatives(), gravitation(), initialize_integ_loop(), integrate_dt(), remove_integrable_object(), remove_sim_object_bodies(), set_deriv_ephem_update(), update_integration_group(), and ~JeodDynbodyIntegrationLoop().

8.6.5.6 integ_group_factory

```
const DynamicsIntegrationGroup* jeod::JeodDynbodyIntegrationLoop::integ_group_factory  [protected]
```

The externally-supplied integration group used as a template for creating this integration loop's integration group.

trick_units(–)

Definition at line 390 of file trick_dynbody_integ_loop.hh.

Referenced by initialize_integ_loop().

8.6.5.7 integ_interface

```
JeodTrickIntegrator jeod::JeodDynbodyIntegrationLoop::integ_interface  [protected]
```

Dummy integration interface; needed by the integ_group.

trick_units(–)

Definition at line 379 of file trick_dynbody_integ_loop.hh.

Referenced by initialize_integ_loop().

8.6.5.8 loop_sim_object

```
Trick::SimObject* jeod::JeodDynbodyIntegrationLoop::loop_sim_object  [protected]
```

The simulation object that contains this integration loop object.

trick_units(–)

Definition at line 359 of file trick_dynbody_integ_loop.hh.

Referenced by JeodDynbodyIntegrationLoop().

8.6.5.9 time_manager

```
TimeManager* jeod::JeodDynbodyIntegrationLoop::time_manager  [protected]
```

The JEOD time manager.

trick_units(–)

Definition at line 369 of file trick_dynbody_integ_loop.hh.

Referenced by initialize_integ_loop(), and set_time_to_loop_start().

The documentation for this class was generated from the following files:

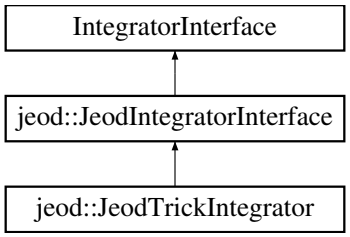
- trick_dynbody_integ_loop.hh
- trick_dynbody_integ_loop.cc

8.7 jeod::JeodIntegratorInterface Class Reference

A JeodIntegratorInterface extends the ER7 IntegratorInterface with the concept of a pointer to the simulation engine's integration object.

```
#include <jeod_integrator_interface.hh>
```

Inheritance diagram for jeod::JeodIntegratorInterface:



Public Member Functions

- virtual ~JeodIntegratorInterface ()

  Destructor.
- virtual er7_utils::Integration::Technique interpret_integration_type (int) const =0

  Interpret the integration technique.
- virtual Trick::Integrator * get_integrator ()=0

  Get the simulation engine's integrator.

Friends

- class InputProcessor
- void init_attrjeod__JeodIntegratorInterface ()

8.7.1 Detailed Description

A JeodIntegratorInterface extends the ER7 IntegratorInterface with the concept of a pointer to the simulation engine's integration object.

Definition at line 86 of file jeod_integrator_interface.hh.

8.7.2 Constructor & Destructor Documentation

8.7.2.1 ~JeodIntegratorInterface()

```
virtual jeod::JeodIntegratorInterface::~JeodIntegratorInterface ( )  [inline], [virtual]
```

Destructor.

Definition at line 98 of file jeod_integrator_interface.hh.

8.7.3 Member Function Documentation

8.7.3.1 get_integrator()

```
virtual Trick::Integrator* jeod::JeodIntegratorInterface::get_integrator ( )  [pure virtual]
```

Get the simulation engine's integrator.

Returns

Pointer to the simulation engine's integrator.

Implemented in jeod::JeodTrickIntegrator.

8.7.3.2 interpret_integration_type()

```
virtual er7_utils::Integration::Technique jeod::JeodIntegratorInterface::interpret_integration↩
_type (
            int  ) const  [pure virtual]
```

Interpret the integration technique.

Implemented in jeod::JeodTrickIntegrator.

8.7.4 Friends And Related Function Documentation

8.7.4.1 init_attrjeod__JeodIntegratorInterface

```
void init_attrjeod__JeodIntegratorInterface ( )  [friend]
```

8.7.4.2 InputProcessor

```
friend class InputProcessor  [friend]
```

Definition at line 87 of file jeod_integrator_interface.hh.

The documentation for this class was generated from the following file:

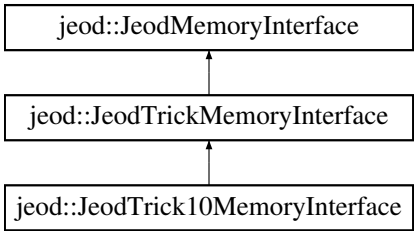
- jeod_integrator_interface.hh

8.8 jeod::JeodMemoryInterface Class Reference

Abstract interface between the JEOD memory manager and the simulation engine.

```
#include <memory_interface.hh>
```

Inheritance diagram for jeod::JeodMemoryInterface:



Public Member Functions

- JeodMemoryInterface ()

  Default constructor.
- virtual ~JeodMemoryInterface ()

  Destructor.
- JeodMemoryInterface (const JeodMemoryInterface &)

  Copy constructor.
- JeodMemoryInterface & operator= (const JeodMemoryInterface &)

  Assignment operator.
- virtual const struct ATTRIBUTES_tag * find_attributes (const std::string &type_name) const =0

  Find the attributes for a given class name.
- virtual const struct ATTRIBUTES_tag * find_attributes (const std::type_info &data_type) const =0

  Find the attributes for a given class.
- virtual struct ATTRIBUTES_tag primitive_attributes (const std::type_info &data_type) const =0

  Create an attributes structure that represents a primitive type.
- virtual struct ATTRIBUTES_tag pointer_attributes (const struct ATTRIBUTES_tag &pointed_to_attr) const =0

  Create an attributes structure that represents a pointer type.
- virtual struct ATTRIBUTES_tag void_pointer_attributes (void) const =0

  Create a simulation engine description of void.*
- virtual struct ATTRIBUTES_tag structure_attributes (const struct ATTRIBUTES_tag *target_attr, std::size_t target_size) const =0

  Create an attributes structure that represents a structured type.
- virtual bool register_allocation (const void *addr, const JeodMemoryItem &item, const JeodMemoryType↩ Descriptor &tdesc, const char *file, unsigned int line)=0

  Register allocated memory with the simulation engine.
- virtual void deregister_allocation (const void *addr, const JeodMemoryItem &item, const JeodMemoryType↩ Descriptor &tdesc, const char *file, unsigned int line)=0

  Revoke registation of memory that is about to be deleted.
- virtual void register_container (const void *container, const JeodMemoryTypeDescriptor &container_type, const char *elem_name, JeodCheckpointable &checkpointable)=0

  Register a JeodCheckpointable object with the simulation engine.
- virtual void deregister_container (const void *container, const JeodMemoryTypeDescriptor &container_type, const char *elem_name, JeodCheckpointable &checkpointable)=0

  Deregister a JeodCheckpointable object with the simulation engine.
- virtual bool is_checkpoint_restart_supported (void) const =0

  Indicates whether the checkpoint/restart methods are viable.
- virtual const std::string get_name_at_address (const void *addr, const JeodMemoryTypeDescriptor *tdesc) const =0

  Get the simulation engine's name (if any) of the address.
- virtual void * get_address_at_name (const std::string &name) const =0

  Get the address (if any) identified by the given name.

Friends

- class InputProcessor
- void init_attrjeod__JeodMemoryInterface ()

8.8.1 Detailed Description

Abstract interface between the JEOD memory manager and the simulation engine.

Definition at line 90 of file memory_interface.hh.

8.8.2 Constructor & Destructor Documentation

8.8.2.1 JeodMemoryInterface() [1/2]

```
jeod::JeodMemoryInterface::JeodMemoryInterface (
            void  )
```

Default constructor.

Definition at line 41 of file memory_interface.cc.

8.8.2.2 ~JeodMemoryInterface()

```
jeod::JeodMemoryInterface::~JeodMemoryInterface (
            void  )  [virtual]
```

Destructor.

Definition at line 51 of file memory_interface.cc.

8.8.2.3 JeodMemoryInterface() [2/2]

```
jeod::JeodMemoryInterface::JeodMemoryInterface (
            const JeodMemoryInterface & src )  [explicit]
```

Copy constructor.

Parameters

in	*src*	Item to be copied

Definition at line 62 of file memory_interface.cc.

8.8.3 Member Function Documentation

8.8.3.1 deregister_allocation()

```
virtual void jeod::JeodMemoryInterface::deregister_allocation (
            const void * addr,
```

```
            const JeodMemoryItem & item,
            const JeodMemoryTypeDescriptor & tdesc,
            const char * file,
            unsigned int line )  [pure virtual]
```

Revoke registation of memory that is about to be deleted.

Parameters

in	*addr*	Address of allocated memory to be de-registered.
in	*item*	JEOD descriptor of the memory
in	*tdesc*	JEOD descriptor of the type of the allocated memory
in	*file*	File in which allocation was performed
in	*line*	Line number in that file

Implemented in jeod::JeodTrickMemoryInterface.

8.8.3.2 deregister_container()

```
virtual void jeod::JeodMemoryInterface::deregister_container (
            const void * container,
            const JeodMemoryTypeDescriptor & container_type,
            const char * elem_name,
            JeodCheckpointable & checkpointable )  [pure virtual]
```

Deregister a JeodCheckpointable object with the simulation engine.

Parameters

in	*container*	Object that contains the checkpointable
in	*container_type*	Checkpointable container type info
in	*elem_name*	Element name of checkpointable object
in,out	*checkpointable*	The checkpointable object itself

Implemented in jeod::JeodTrickMemoryInterface, and jeod::JeodTrick10MemoryInterface.

8.8.3.3 find_attributes() [1/2]

```
virtual const struct ATTRIBUTES_tag* jeod::JeodMemoryInterface::find_attributes (
            const std::string & type_name ) const  [pure virtual]
```

Find the attributes for a given class name.

Parameters

in	*type_name*	Name of the class.

Returns

Attributes pointer. Note: This is not an allocated pointer.

Implemented in jeod::JeodTrickMemoryInterface.

8.8.3.4 find_attributes() [2/2]

```
virtual const struct ATTRIBUTES_tag* jeod::JeodMemoryInterface::find_attributes (
            const std::type_info & data_type ) const  [pure virtual]
```

Find the attributes for a given class.

Parameters

in	*data_type*	RTTI descriptor of the type.

Returns

Attributes pointer. Note: This is not an allocated pointer.

Implemented in jeod::JeodTrickMemoryInterface.

8.8.3.5 get_address_at_name()

```
virtual void* jeod::JeodMemoryInterface::get_address_at_name (
            const std::string & name ) const  [pure virtual]
```

Get the address (if any) identified by the given name.

Note

An implementation that does not support name translation will return the null pointer.
A stubbed implementation should have its is_checkpoint_restart_supported method return false.

Returns

Address corresponding to the given name, if any

Parameters

in	*name*	Value previously constructed by get_name_at_address()

Implemented in jeod::JeodTrickMemoryInterface, and jeod::JeodTrick10MemoryInterface.

8.8.3.6 get_name_at_address()

```
virtual const std::string jeod::JeodMemoryInterface::get_name_at_address (
            const void * addr,
            const JeodMemoryTypeDescriptor * tdesc ) const  [pure virtual]
```

Get the simulation engine's name (if any) of the address.

A derived class associated with a simulation engine that does not support this translation should return an empty string for all calls. When the underlying simulation engine does support this translation, the implementation should return values as follows:

- The string "NULL" if the input address is the null pointer.
- The empty string to indicate an invalid input address or an input address that is unknown to the simulation engine.
- A non-empty, non-"NULL" string to indicate a valid address. Applying the get_address_at_name method to this result must yield the input address.

Note

A stubbed implementation should have its is_checkpoint_restart_supported method return false.

Returns

Name of the address, if any

Parameters

in	*addr*	Address of memory to identified by name
in	*tdesc*	Type context in which to interpret the address

Implemented in jeod::JeodTrickMemoryInterface, and jeod::JeodTrick10MemoryInterface.

8.8.3.7 is_checkpoint_restart_supported()

```
virtual bool jeod::JeodMemoryInterface::is_checkpoint_restart_supported (
            void  ) const  [pure virtual]
```

Indicates whether the checkpoint/restart methods are viable.

Checkpoint/restart can be used only in an environment that provides viable checkpoint/restart methods.

Returns

True if the checkpoint / restart is supported, false otherwise.

Implemented in jeod::JeodTrickMemoryInterface, and jeod::JeodTrick10MemoryInterface.

8.8.3.8 operator=()

```
JeodMemoryInterface & jeod::JeodMemoryInterface::operator= (
            const JeodMemoryInterface & src )
```

Assignment operator.

Returns

*this

Parameters

in	*src*	Item to be copied

Definition at line 75 of file memory_interface.cc.

8.8.3.9 pointer_attributes()

```
virtual struct ATTRIBUTES_tag jeod::JeodMemoryInterface::pointer_attributes (
            const struct ATTRIBUTES_tag & pointed_to_attr ) const  [pure virtual]
```

Create an attributes structure that represents a pointer type.

Parameters

in	*pointed_to_attr*	Attributes of the pointed-to type.

Returns

Attribute structure.

Implemented in jeod::JeodTrickMemoryInterface.

8.8.3.10 primitive_attributes()

```
virtual struct ATTRIBUTES_tag jeod::JeodMemoryInterface::primitive_attributes (
            const std::type_info & data_type ) const  [pure virtual]
```

Create an attributes structure that represents a primitive type.

Parameters

in	*data_type*	RTTI descriptor of the type.

Returns

Attributes structure.

Implemented in jeod::JeodTrickMemoryInterface.

8.8.3.11 register_allocation()

```
virtual bool jeod::JeodMemoryInterface::register_allocation (
            const void * addr,
            const JeodMemoryItem & item,
            const JeodMemoryTypeDescriptor & tdesc,
            const char * file,
            unsigned int line ) [pure virtual]
```

Register allocated memory with the simulation engine.

Parameters

in	*addr*	Address of allocated memory to be registered.
in	*item*	JEOD descriptor of the allocated memory
in	*tdesc*	JEOD descriptor of the type of the allocated memory
in	*file*	File in which allocation was performed
in	*line*	Line number in that file

Implemented in jeod::JeodTrickMemoryInterface.

8.8.3.12 register_container()

```
virtual void jeod::JeodMemoryInterface::register_container (
            const void * container,
            const JeodMemoryTypeDescriptor & container_type,
            const char * elem_name,
            JeodCheckpointable & checkpointable ) [pure virtual]
```

Register a JeodCheckpointable object with the simulation engine.

Parameters

in	*container*	Object that contains the checkpointable
in	*container_type*	Checkpointable container type info
in	*elem_name*	Element name of checkpointable object
in,out	*checkpointable*	The checkpointable object itself

Implemented in jeod::JeodTrickMemoryInterface, and jeod::JeodTrick10MemoryInterface.

8.8.3.13 structure_attributes()

```
virtual struct ATTRIBUTES_tag jeod::JeodMemoryInterface::structure_attributes (
            const struct ATTRIBUTES_tag * target_attr,
            std::size_t target_size ) const  [pure virtual]
```

Create an attributes structure that represents a structured type.

Parameters

in	*target_attr*	Attributes from find_attributes
in	*target_size*	Size of the underlying type

Returns

Attribute structure.

Implemented in jeod::JeodTrickMemoryInterface.

8.8.3.14 void_pointer_attributes()

```
virtual struct ATTRIBUTES_tag jeod::JeodMemoryInterface::void_pointer_attributes (
            void  ) const  [pure virtual]
```

Create a simulation engine description of void*.

Returns

Attribute structure.

Implemented in jeod::JeodTrickMemoryInterface.

8.8.4 Friends And Related Function Documentation

8.8.4.1 init_attrjeod__JeodMemoryInterface

```
void init_attrjeod__JeodMemoryInterface ( )  [friend]
```

8.8.4.2 InputProcessor

```
friend class InputProcessor  [friend]
```

Definition at line 92 of file memory_interface.hh.

The documentation for this class was generated from the following files:

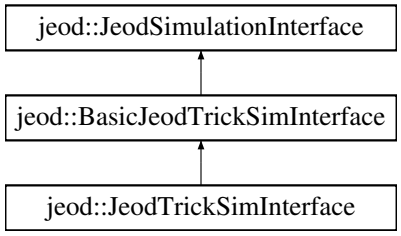
- memory_interface.hh
- memory_interface.cc

8.9 jeod::JeodSimulationInterface Class Reference

This abstract class defines the basis for the interface between JEOD and a simulation engine.

```
#include <simulation_interface.hh>
```

Inheritance diagram for jeod::JeodSimulationInterface:



Public Types

- enum Mode {
  Construction = 0, PreCheckpoint = 1, Checkpoint = 2, PostCheckpoint = 3,
  Restart = 4, Restore = 5, Initialization = 6, Operational = 7,
  Shutdown = 8, Dead = 9, NumModes = 10 }

  Defines the states of the JeodSimulationInterface state machine.

Public Member Functions

- JeodSimulationInterface ()

  Construct a JeodSimulationInterface object.
- virtual ~JeodSimulationInterface ()

  Destruct a JeodSimulationInterface object.
- virtual void configure (const JeodSimulationInterfaceInit &config)

  Configure a JeodSimulationInterface object.
- Mode get_mode (void) const

  Get the current mode.
- virtual void set_mode (Mode new_mode)

  Set the mode, but only if allowed per the mode state transition diagram.

Static Public Member Functions

- static JeodIntegratorInterface * create_integrator_interface (void)

  Create a simulation integrator interface object.
- static double get_job_cycle (void)

  Get the cycle time of the currently executing job.
- static std::string get_name_at_address (const void *addr, const JeodMemoryTypeDescriptor *tdesc)

  Translate the given address to a symbolic name.
- static void * get_address_at_name (const std::string &name)

  Translate the given symbolic name to an address.
- static JeodMemoryInterface & get_memory_interface (void)

  Get the interface with the simulation memory model.
- static SectionedInputStream get_checkpoint_reader (const std::string &section_id)

  Get a reader of a section of the currently open checkpoint file.
- static SectionedOutputStream get_checkpoint_writer (const std::string &section_id)

  Get a writer to a section of the currently open checkpoint file.

Protected Member Functions

- virtual JeodIntegratorInterface * create_integrator_internal (void)=0

  Create an integration interface object.
- virtual double get_job_cycle_internal (void)=0

  Get the simulation cycle time of the currently executing function.
- virtual JeodMemoryInterface & get_memory_interface_internal (void)=0

  Get the interface with the simulation memory manager.
- virtual SectionedInputStream get_checkpoint_reader_internal (const std::string &section_id)=0

  Get a checkpoint section reader.
- virtual SectionedOutputStream get_checkpoint_writer_internal (const std::string &section_id)=0

  Get a checkpoint section writer.

Protected Attributes

- Mode mode

  The mode in which the simulation interface is operating.
- Mode saved_mode

  The mode prior to a checkpoint or restart process.

Static Protected Attributes

- static JeodSimulationInterface * sim_interface = NULL

  The singleton instance of a SimulationInterface object that must be created by a conforming JEOD simulation before any call can be made to one of the three static methods declared above.

Private Member Functions

- JeodSimulationInterface (const JeodSimulationInterface &)

  Not implemented.
- JeodSimulationInterface & operator= (const JeodSimulationInterface &)

  Not implemented.

Friends

- class InputProcessor
- void init_attrjeod__JeodSimulationInterface ()

8.9.1 Detailed Description

This abstract class defines the basis for the interface between JEOD and a simulation engine.

A compliant derived class must contain one instance each of a class that derives from MessageHandler and a class that derives from JeodMemoryManager. The MessageHandler object must be constructed before the Jeod↩ MemoryManager object; destruction must be performed in reverse order.

Definition at line 138 of file simulation_interface.hh.

8.9.2 Member Enumeration Documentation

8.9.2.1 Mode

`enum jeod::JeodSimulationInterface::Mode`

Defines the states of the JeodSimulationInterface state machine.

Enumerator

Construction	
PreCheckpoint	
Checkpoint	
PostCheckpoint	
Restart	
Restore	
Initialization	
Operational	
Shutdown	
Dead	
NumModes	

Definition at line 148 of file simulation_interface.hh.

8.9.3 Constructor & Destructor Documentation

8.9.3.1 JeodSimulationInterface() [1/2]

```
jeod::JeodSimulationInterface::JeodSimulationInterface (
            void  )
```

Construct a JeodSimulationInterface object.

Definition at line 73 of file simulation_interface.cc.

References sim_interface, and jeod::SimInterfaceMessages::singleton_error.

8.9.3.2 ~JeodSimulationInterface()

```
jeod::JeodSimulationInterface::~JeodSimulationInterface (
            void  ) [virtual]
```

Destruct a JeodSimulationInterface object.

Definition at line 93 of file simulation_interface.cc.

References sim_interface.

8.9.3.3 JeodSimulationInterface() [2/2]

```
jeod::JeodSimulationInterface::JeodSimulationInterface (
            const JeodSimulationInterface &  ) [private]
```

Not implemented.

8.9.4 Member Function Documentation

8.9.4.1 configure()

```
void jeod::JeodSimulationInterface::configure (
            const JeodSimulationInterfaceInit & config ) [virtual]
```

Configure a JeodSimulationInterface object.

Parameters

in	*config*	Configuration spec

Definition at line 107 of file simulation_interface.cc.

References jeod::JeodSimulationInterfaceInit::memory_debug_level, jeod::JeodSimulationInterfaceInit::message_suppress_id, jeod::JeodSimulationInterfaceInit::message_suppress_location, and jeod::JeodSimulationInterfaceInit::message_suppression_level.

8.9.4.2 create_integrator_interface()

```
JeodIntegratorInterface * jeod::JeodSimulationInterface::create_integrator_interface (
            void ) [static]
```

Create a simulation integrator interface object.

Returns

Constructed IntegratorInterface object.

Definition at line 129 of file simulation_interface.cc.

References create_integrator_internal(), sim_interface, and jeod::SimInterfaceMessages::singleton_error.

8.9.4.3 create_integrator_internal()

```
virtual JeodIntegratorInterface* jeod::JeodSimulationInterface::create_integrator_internal (
            void ) [protected], [pure virtual]
```

Create an integration interface object.

The calling object is responsible for destroying the created object.

Returns

Created integration interface object.

Implemented in jeod::BasicJeodTrickSimInterface.

Referenced by create_integrator_interface().

8.9.4.4 get_address_at_name()

```
void * jeod::JeodSimulationInterface::get_address_at_name (
            const std::string & name ) [static]
```

Translate the given symbolic name to an address.

Returns

Address

Parameters

in	*name*	Symbolic name

Definition at line 223 of file simulation_interface.cc.

References get_memory_interface_internal(), sim_interface, and jeod::SimInterfaceMessages::singleton_error.

Referenced by jeod::JeodDynbodyIntegrationLoop::find_containing_sim_object().

8.9.4.5 get_checkpoint_reader()

```
SectionedInputStream jeod::JeodSimulationInterface::get_checkpoint_reader (
            const std::string & section_id )  [static]
```

Get a reader of a section of the currently open checkpoint file.

Returns

Checkpoint reader

Parameters

in	*section_id*	Section ID

Definition at line 248 of file simulation_interface.cc.

References get_checkpoint_reader_internal(), sim_interface, and jeod::SimInterfaceMessages::singleton_error.

Referenced by jeod::JeodTrick10MemoryInterface::restore_allocations(), and jeod::JeodTrick10MemoryInterface::restore_containers().

8.9.4.6 get_checkpoint_reader_internal()

```
virtual SectionedInputStream jeod::JeodSimulationInterface::get_checkpoint_reader_internal (
            const std::string & section_id )  [protected], [pure virtual]
```

Get a checkpoint section reader.

Implemented in jeod::BasicJeodTrickSimInterface.

Referenced by get_checkpoint_reader().

8.9.4.7 get_checkpoint_writer()

```
SectionedOutputStream jeod::JeodSimulationInterface::get_checkpoint_writer (
            const std::string & section_id )  [static]
```

Get a writer to a section of the currently open checkpoint file.

Returns

Checkpoint writer

Parameters

in	*section_id*	Section ID

Definition at line 269 of file simulation_interface.cc.

References get_checkpoint_writer_internal(), sim_interface, and jeod::SimInterfaceMessages::singleton_error.

Referenced by jeod::JeodTrick10MemoryInterface::checkpoint_allocations(), and jeod::JeodTrick10MemoryInterface::checkpoint_containers().

8.9.4.8 get_checkpoint_writer_internal()

```
virtual SectionedOutputStream jeod::JeodSimulationInterface::get_checkpoint_writer_internal (
            const std::string & section_id )  [protected], [pure virtual]
```

Get a checkpoint section writer.

Implemented in jeod::BasicJeodTrickSimInterface.

Referenced by get_checkpoint_writer().

8.9.4.9 get_job_cycle()

```
double jeod::JeodSimulationInterface::get_job_cycle (
            void  )  [static]
```

Get the cycle time of the currently executing job.

Returns

Cycle time in simulation engine seconds of the currently executing job.
Units: s

Definition at line 152 of file simulation_interface.cc.

References get_job_cycle_internal(), sim_interface, and jeod::SimInterfaceMessages::singleton_error.

8.9.4.10 get_job_cycle_internal()

```
virtual double jeod::JeodSimulationInterface::get_job_cycle_internal (
            void  )  [protected], [pure virtual]
```

Get the simulation cycle time of the currently executing function.

Returns

Cycle time in simulation engine seconds

Implemented in jeod::BasicJeodTrickSimInterface.

Referenced by get_job_cycle().

8.9.4.11 get_memory_interface()

```
JeodMemoryInterface & jeod::JeodSimulationInterface::get_memory_interface (
            void  )  [static]
```

Get the interface with the simulation memory model.

Returns

Memory interface

Definition at line 175 of file simulation_interface.cc.

References get_memory_interface_internal(), sim_interface, and jeod::SimInterfaceMessages::singleton_error.

8.9.4.12 get_memory_interface_internal()

```
virtual JeodMemoryInterface& jeod::JeodSimulationInterface::get_memory_interface_internal (
            void  )  [protected], [pure virtual]
```

Get the interface with the simulation memory manager.

Returns

JEOD/simulation engine memory interface.

Implemented in jeod::BasicJeodTrickSimInterface.

Referenced by get_address_at_name(), get_memory_interface(), and get_name_at_address().

8.9.4.13 get_mode()

```
Mode jeod::JeodSimulationInterface::get_mode (
            void  ) const  [inline]
```

Get the current mode.

Definition at line 250 of file simulation_interface.hh.

References mode.

Referenced by jeod::BasicJeodTrickSimInterface::set_mode().

8.9.4.14 get_name_at_address()

```
std::string jeod::JeodSimulationInterface::get_name_at_address (
            const void * addr,
            const JeodMemoryTypeDescriptor * tdesc )  [static]
```

Translate the given address to a symbolic name.

Returns

Symbolic name

Parameters

in	*addr*	Address
in	*tdesc*	Descriptor

Definition at line 197 of file simulation_interface.cc.

References get_memory_interface_internal(), sim_interface, and jeod::SimInterfaceMessages::singleton_error.

Referenced by jeod::JeodDynbodyIntegrationLoop::find_containing_sim_object().

8.9.4.15 operator=()

```
JeodSimulationInterface& jeod::JeodSimulationInterface::operator= (
            const JeodSimulationInterface &  )  [private]
```

Not implemented.

8.9.4.16 set_mode()

```
void jeod::JeodSimulationInterface::set_mode (
            Mode new_mode )  [virtual]
```

Set the mode, but only if allowed per the mode state transition diagram.

Assumptions and Limitations

- The standard JEODSys Trick sim object follows the correct state transition diagram. A similar sequence must be implemented when JEOD is used outside of the Trick environment. In a Trick environment, *nobody* should call this function except the Trick scheduler, and these calls must conform with the sequence in the standard JEODSys Trick sim object.

Parameters

in	*new_mode*	New mode

Reimplemented in jeod::BasicJeodTrickSimInterface.

Definition at line 296 of file simulation_interface.cc.

References Checkpoint, Construction, Dead, jeod::SimInterfaceMessages::implementation_error, Initialization, mode, NumModes, Operational, jeod::SimInterfaceMessages::phasing_error, PostCheckpoint, PreCheckpoint, Restart, Restore, saved_mode, and Shutdown.

Referenced by jeod::BasicJeodTrickSimInterface::set_mode().

8.9.5 Friends And Related Function Documentation

8.9.5.1 init_attrjeod__JeodSimulationInterface

```
void init_attrjeod__JeodSimulationInterface ( )  [friend]
```

8.9.5.2 InputProcessor

```
friend class InputProcessor  [friend]
```

Definition at line 139 of file simulation_interface.hh.

8.9.6 Field Documentation

8.9.6.1 mode

```
Mode jeod::JeodSimulationInterface::mode  [protected]
```

The mode in which the simulation interface is operating.

trick_units(–)

Definition at line 313 of file simulation_interface.hh.

Referenced by get_mode(), and set_mode().

8.9.6.2 saved_mode

```
Mode jeod::JeodSimulationInterface::saved_mode  [protected]
```

The mode prior to a checkpoint or restart process.

set_mode(Restore) restores the mode to this saved value.trick_units(–)

Definition at line 319 of file simulation_interface.hh.

Referenced by set_mode().

8.9.6.3 sim_interface

```
JeodSimulationInterface * jeod::JeodSimulationInterface::sim_interface = NULL  [static], [protected]
```

The singleton instance of a SimulationInterface object that must be created by a conforming JEOD simulation before any call can be made to one of the three static methods declared above.

The first created instance of a class that derives from this base class becomes **the** SimulationInterface object used during the course of the simulation. Creation of more than one SimulationInteface objects is a non-fatal error. Attempts to allocate memory or generate a message prior creating a SimulationInteface object is a fatal error.trick↩
_io(∗o) trick_units(–)

Definition at line 270 of file simulation_interface.hh.

Referenced by create_integrator_interface(), get_address_at_name(), get_checkpoint_reader(), get_checkpoint↩
_writer(), get_job_cycle(), get_memory_interface(), get_name_at_address(), JeodSimulationInterface(), and ∼↩
JeodSimulationInterface().

The documentation for this class was generated from the following files:

- simulation_interface.hh
- simulation_interface.cc

8.10 jeod::JeodSimulationInterfaceInit Class Reference

Define configuration data needed to configure the dynamically-created message handler and memory manager.

`#include <simulation_interface.hh>`

Public Member Functions

- JeodSimulationInterfaceInit ()
  Construct a JeodSimulationInterfaceInit object.

Data Fields

- unsigned int message_suppression_level
  Specifies the message handler's message suppression level; see MessageHandler::suppression_level for details.
- bool message_suppress_id
  Specifies the message handler's suppress_id flag; see MessageHandler::suppression_id for details.
- bool message_suppress_location
  Specifies the message handler's suppress_location flag; see MessageHandler::suppression_location for details.
- unsigned int memory_debug_level
  Specifies the memory manager's debug level; see JeodMemoryManager::debug_level for details.

8.10.1 Detailed Description

Define configuration data needed to configure the dynamically-created message handler and memory manager.

Definition at line 87 of file simulation_interface.hh.

8.10.2 Constructor & Destructor Documentation

8.10.2.1 JeodSimulationInterfaceInit()

```
jeod::JeodSimulationInterfaceInit::JeodSimulationInterfaceInit (
            void  )
```

Construct a JeodSimulationInterfaceInit object.

Definition at line 51 of file simulation_interface.cc.

References memory_debug_level, and message_suppression_level.

8.10.3 Field Documentation

8.10.3.1 memory_debug_level

```
unsigned int jeod::JeodSimulationInterfaceInit::memory_debug_level
```

Specifies the memory manager's debug level; see JeodMemoryManager::debug_level for details.

Default value: 0.trick_units(–)

Definition at line 127 of file simulation_interface.hh.

Referenced by jeod::JeodSimulationInterface::configure(), and JeodSimulationInterfaceInit().

8.10.3.2 message_suppress_id

```
bool jeod::JeodSimulationInterfaceInit::message_suppress_id
```

Specifies the message handler's suppress_id flag; see MessageHandler::suppression_id for details.

Default value: false.trick_units(–)

Definition at line 113 of file simulation_interface.hh.

Referenced by jeod::JeodSimulationInterface::configure().

8.10.3.3 message_suppress_location

```
bool jeod::JeodSimulationInterfaceInit::message_suppress_location
```

Specifies the message handler's suppress_location flag; see MessageHandler::suppression_location for details.

Default value: false.trick_units(–)

Definition at line 120 of file simulation_interface.hh.

Referenced by jeod::JeodSimulationInterface::configure().

8.10.3.4 message_suppression_level

```
unsigned int jeod::JeodSimulationInterfaceInit::message_suppression_level
```

Specifies the message handler's message suppression level; see MessageHandler::suppression_level for details.

Default value: MessageHandler::Warning (warnings and non-fatal errors).trick_units(–)

Definition at line 106 of file simulation_interface.hh.

Referenced by jeod::JeodSimulationInterface::configure(), and JeodSimulationInterfaceInit().

The documentation for this class was generated from the following files:

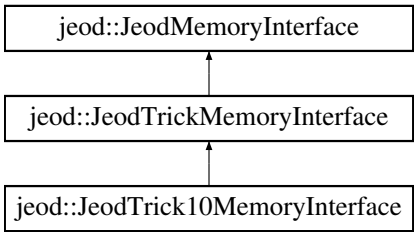
- simulation_interface.hh
- simulation_interface.cc

8.11 jeod::JeodTrick10MemoryInterface Class Reference

A TrickMemoryInterface implements the two required methods needed to register and deregister memory with the simulation engine, Trick in this case.

`#include <trick10_memory_interface.hh>`

Inheritance diagram for jeod::JeodTrick10MemoryInterface:



Public Member Functions

- JeodTrick10MemoryInterface ()

  Construct a JeodTrick10MemoryInterface object.
- virtual ~JeodTrick10MemoryInterface ()

  Destruct a JeodTrick10MemoryInterface object.
- virtual void register_container (const void *owner, const JeodMemoryTypeDescriptor &owner_type, const char *elem_name, JeodCheckpointable &container)

  Register the checkpointable object with Trick.
- virtual void deregister_container (const void *owner, const JeodMemoryTypeDescriptor &owner_type, const char *elem_name, JeodCheckpointable &container)

  Revoke the registrations performed by register_container.
- virtual const std::string get_name_at_address (const void *addr, const JeodMemoryTypeDescriptor *tdesc) const

  Get the simulation name, if any, associated with the address.
- virtual void * get_address_at_name (const std::string &name) const

  Get the address, if any, that corresponds to the given name.
- virtual bool is_checkpoint_restart_supported (void) const

  The Trick10 memory interface supports checkpoint/restart.
- virtual const std::string get_trick_checkpoint_file (bool checkpoint)

  Get the name of the current Trick checkpoint file.
- virtual void checkpoint_containers (void)

  Dump the checkpointable objects to the checkpoint file.
- virtual void restore_containers (void)

  Restore the checkpointable objects from the checkpoint file.
- virtual void checkpoint_allocations (void)

  Dump the allocation information to the checkpoint file.
- virtual void restore_allocations (JeodMemoryManager &memory_manager)

  Restore the allocated data per the checkpoint file.

Protected Member Functions

- std::string get_container_id (const ContainerListEntry &entry) const

  Construct the identifier for a checkpointable object.
- std::string translate_addr_to_name (const void *addr, const ATTRIBUTES *attr) const

  Translate the given address to an address specification string, with the address interpreted in the context of the supplied attributes.
- void * translate_name_to_addr (const std::string &spec) const

  Translate the given address specification string to an address.

Protected Attributes

- Trick::ClassicCheckPointAgent * trick_checkpoint_agent

  Trick checkpoint agent.

Private Member Functions

- JeodTrick10MemoryInterface (const JeodTrick10MemoryInterface &)

  Not implemented.
- JeodTrick10MemoryInterface & operator= (const JeodTrick10MemoryInterface &)

  Not implemented.

Friends

- class InputProcessor
- void init_attrjeod__JeodTrick10MemoryInterface ()

Additional Inherited Members

8.11.1 Detailed Description

A TrickMemoryInterface implements the two required methods needed to register and deregister memory with the simulation engine, Trick in this case.

Definition at line 109 of file trick10_memory_interface.hh.

8.11.2 Constructor & Destructor Documentation

8.11.2.1 JeodTrick10MemoryInterface() [1/2]

```
jeod::JeodTrick10MemoryInterface::JeodTrick10MemoryInterface (
            void  )
```

Construct a JeodTrick10MemoryInterface object.

Definition at line 62 of file trick10_memory_interface.cc.

References jeod::SimInterfaceMessages::interface_error, trick_checkpoint_agent, and trick_MM.

8.11.2.2 ~JeodTrick10MemoryInterface()

```
jeod::JeodTrick10MemoryInterface::~JeodTrick10MemoryInterface (
            void  )  [virtual]
```

Destruct a JeodTrick10MemoryInterface object.

Definition at line 84 of file trick10_memory_interface.cc.

8.11.2.3 JeodTrick10MemoryInterface() [2/2]

```
jeod::JeodTrick10MemoryInterface::JeodTrick10MemoryInterface (
            const JeodTrick10MemoryInterface &  )  [private]
```

Not implemented.

8.11.3 Member Function Documentation

8.11.3.1 checkpoint_allocations()

```
void jeod::JeodTrick10MemoryInterface::checkpoint_allocations (
            void  )  [virtual]
```

Dump the allocation information to the checkpoint file.

Reimplemented from jeod::JeodTrickMemoryInterface.

Definition at line 431 of file trick_memory_interface_chkpnt.cc.

References jeod::SectionedOutputStream::activate(), jeod::JeodTrickMemoryInterface::allocation_map, jeod::JeodTrickMemoryInterface::construct_identifier(), jeod::JeodTrickMemoryInterface::container_list, jeod::JeodSimulationInterface::get_checkpoint_writer(), jeod::SimInterfaceMessages::interface_error, jeod::JeodTrickMemoryInterface::AllocationMapEntry::is_array, jeod::JeodTrickMemoryInterface::AllocationMapEntry::nelements, and jeod::JeodTrickMemoryInterface::AllocationMapEntry::typeid_info.

Referenced by jeod::BasicJeodTrickSimInterface::checkpoint_allocations().

8.11.3.2 checkpoint_containers()

```
void jeod::JeodTrick10MemoryInterface::checkpoint_containers (
            void ) [virtual]
```

Dump the checkpointable objects to the checkpoint file.

Reimplemented from jeod::JeodTrickMemoryInterface.

Definition at line 243 of file trick_memory_interface_chkpnt.cc.

References jeod::SectionedOutputStream::activate(), jeod::JeodTrickMemoryInterface::container_list, jeod::SectionedOutputStream::deactivate(), jeod::JeodSimulationInterface::get_checkpoint_writer(), get_container_id(), and jeod::SimInterfaceMessages::interface_error.

Referenced by jeod::BasicJeodTrickSimInterface::checkpoint_containers().

8.11.3.3 deregister_container()

```
void jeod::JeodTrick10MemoryInterface::deregister_container (
            const void * owner,
            const JeodMemoryTypeDescriptor & owner_type,
            const char * elem_name,
            JeodCheckpointable & container ) [virtual]
```

Revoke the registrations performed by register_container.

This function is typically called at destruction time via JEOD_DEREGISTER_CHECKPOINTABLE.

Assumptions and Limitations

- The following unenforced assumptions are made:
  - A corresponding register_container was previously made.
  - Trick has been pre-initialized.

  Enforcement of the above is the responsibility the simulation developer, the JEOD memory manager, and the simulation interface.

Parameters

in	*owner*	Owner of the container
in	*owner_type*	Owner type descriptor
in	*elem_name*	Container element
in,out	*container*	The container

Reimplemented from jeod::JeodTrickMemoryInterface.

Definition at line 145 of file trick_memory_interface_chkpnt.cc.

References jeod::JeodTrickMemoryInterface::container_list, jeod::JeodTrickMemoryInterface::ContainerListEntry::elem_name, jeod::SimInterfaceMessages::interface_error, jeod::JeodTrickMemoryInterface::ContainerListEntry::owner, and jeod::JeodTrickMemoryInterface::ContainerListEntry::owner_type.

8.11.3.4 get_address_at_name()

```
void * jeod::JeodTrick10MemoryInterface::get_address_at_name (
            const std::string & name ) const  [virtual]
```

Get the address, if any, that corresponds to the given name.

Returns

Name of the address, if any

Parameters

in	*name*	of an address Units: Name

Reimplemented from jeod::JeodTrickMemoryInterface.

Definition at line 124 of file trick_memory_interface_xlate.cc.

References translate_name_to_addr().

8.11.3.5 get_container_id()

```
std::string jeod::JeodTrick10MemoryInterface::get_container_id (
            const ContainerListEntry & entry ) const  [protected]
```

Construct the identifier for a checkpointable object.

Returns

Container ID

Parameters

in	*entry*	Container list entry

Definition at line 200 of file trick_memory_interface_chkpnt.cc.

References jeod::JeodTrickMemoryInterface::ContainerListEntry::elem_name, jeod::JeodTrickMemoryInterface::ContainerListEntry::owner, jeod::JeodTrickMemoryInterface::ContainerListEntry::owner_type, translate_addr_to_name(), and translate_name_to_addr().

Referenced by checkpoint_containers(), register_container(), and restore_containers().

8.11.3.6 get_name_at_address()

```
const std::string jeod::JeodTrick10MemoryInterface::get_name_at_address (
            const void * addr,
            const JeodMemoryTypeDescriptor * tdesc ) const  [virtual]
```

Get the simulation name, if any, associated with the address.

Returns

Name of the address, if any

Parameters

in	*addr*	Address of memory whose name is to be found
in	*tdesc*	How to interpret address

Reimplemented from jeod::JeodTrickMemoryInterface.

Definition at line 88 of file trick_memory_interface_xlate.cc.

References translate_addr_to_name(), and translate_name_to_addr().

8.11.3.7 get_trick_checkpoint_file()

```
const std::string jeod::JeodTrick10MemoryInterface::get_trick_checkpoint_file (
            bool checkpoint )  [virtual]
```

Get the name of the current Trick checkpoint file.

Returns

Name of the current Trick checkpoint file.
Units:

Parameters

in	*checkpoint*	True for checkpoint, false for restart

Reimplemented from jeod::JeodTrickMemoryInterface.

Definition at line 67 of file trick_memory_interface_xlate.cc.

Referenced by jeod::BasicJeodTrickSimInterface::open_checkpoint_file(), and jeod::BasicJeodTrickSimInterface::open_restart_file().

8.11.3.8 is_checkpoint_restart_supported()

```
virtual bool jeod::JeodTrick10MemoryInterface::is_checkpoint_restart_supported (
            void  ) const  [inline], [virtual]
```

The Trick10 memory interface supports checkpoint/restart.

Reimplemented from jeod::JeodTrickMemoryInterface.

Definition at line 149 of file trick10_memory_interface.hh.

Referenced by jeod::BasicJeodTrickSimInterface::checkpoint_allocations(), jeod::BasicJeodTrickSimInterface::checkpoint_containers(), jeod::BasicJeodTrickSimInterface::open_checkpoint_file(), jeod::BasicJeodTrickSimInterface::open_restart_file(), jeod::BasicJeodTrickSimInterface::restore_allocations(), and jeod::BasicJeodTrickSimInterface::restore_containers().

8.11.3.9 operator=()

```
JeodTrick10MemoryInterface& jeod::JeodTrick10MemoryInterface::operator= (
            const JeodTrick10MemoryInterface &  )  [private]
```

Not implemented.

8.11.3.10 register_container()

```
void jeod::JeodTrick10MemoryInterface::register_container (
            const void * owner,
            const JeodMemoryTypeDescriptor & owner_type,
            const char * elem_name,
            JeodCheckpointable & container )  [virtual]
```

Register the checkpointable object with Trick.

This function is typically called at construction or initialization time via JEOD_REGISTER_CHECKPOINTABLE.

Assumptions and Limitations

- The following unenforced assumptions are made:
  - Sim objects have been constructed and registered with Trick.
  - Checkpointable objects are unique.
  - Trick has been pre-initialized.
  - Not in shutdown mode.

  Enforcement of the above is the responsibility the simulation developer, the JEOD memory manager, and the simulation interface.

Parameters

in	*owner*	Owner of the container
in	*owner_type*	Owner type descriptor
in	*elem_name*	Container element
in,out	*container*	The container

Reimplemented from jeod::JeodTrickMemoryInterface.

Definition at line 82 of file trick_memory_interface_chkpnt.cc.

References jeod::JeodTrickMemoryInterface::container_list, jeod::JeodTrickMemoryInterface::ContainerListEntry::elem_name, get_container_id(), jeod::SimInterfaceMessages::interface_error, jeod::JeodTrickMemoryInterface::ContainerListEntry::owner, and jeod::JeodTrickMemoryInterface::ContainerListEntry::owner_type.

8.11.3.11 restore_allocations()

```
void jeod::JeodTrick10MemoryInterface::restore_allocations (
            JeodMemoryManager & memory_manager )  [virtual]
```

Restore the allocated data per the checkpoint file.

Parameters

in,out	*memory_manager*	JEOD memory manager

Reimplemented from jeod::JeodTrickMemoryInterface.

Definition at line 501 of file trick_memory_interface_chkpnt.cc.

References jeod::SectionedInputStream::activate(), jeod::JeodTrickMemoryInterface::container_list, jeod::JeodSimulationInterface::get_checkpoint_reader(), and jeod::SimInterfaceMessages::interface_error.

Referenced by jeod::BasicJeodTrickSimInterface::restore_allocations().

8.11.3.12 restore_containers()

```
void jeod::JeodTrick10MemoryInterface::restore_containers (
            void )  [virtual]
```

Restore the checkpointable objects from the checkpoint file.

Reimplemented from jeod::JeodTrickMemoryInterface.

Definition at line 312 of file trick_memory_interface_chkpnt.cc.

References jeod::SectionedInputStream::activate(), jeod::JeodTrickMemoryInterface::ContainerListEntry::container, jeod::JeodTrickMemoryInterface::container_list, jeod::SectionedInputStream::deactivate(), jeod::JeodSimulationInterface::get_checkpoint_reader(), get_container_id(), and jeod::SimInterfaceMessages::interface_error.

Referenced by jeod::BasicJeodTrickSimInterface::restore_containers().

8.11.3.13 translate_addr_to_name()

```
std::string jeod::JeodTrick10MemoryInterface::translate_addr_to_name (
            const void * addr,
            const ATTRIBUTES * attr ) const  [protected]
```

Translate the given address to an address specification string, with the address interpreted in the context of the supplied attributes.

It is the attributes structure that resolves the A versus A.B versus A.B.C ambiguity.

Note

The attributes structure must be that of a pointer type.

Parameters

addr	The address to be translated.
attr	The context in which to interpret the address.

Returns

Address specification string, e.g., &foo.bar.baz[42]

Definition at line 156 of file trick_memory_interface_xlate.cc.

References jeod::SimInterfaceMessages::interface_error, jeod::JeodTrickMemoryInterface::pointer_attributes(), and trick_checkpoint_agent.

Referenced by get_container_id(), and get_name_at_address().

8.11.3.14 translate_name_to_addr()

```
void * jeod::JeodTrick10MemoryInterface::translate_name_to_addr (
            const std::string & spec ) const  [protected]
```

Translate the given address specification string to an address.

This is the inverse of translate_addr_to_name.

Parameters

spec	The address specification to be interpreted.

Returns

Address corresponding to the address specification.

Definition at line 196 of file trick_memory_interface_xlate.cc.

References jeod::SimInterfaceMessages::interface_error.

Referenced by get_address_at_name(), get_container_id(), and get_name_at_address().

8.11.4 Friends And Related Function Documentation

8.11.4.1 init_attrjeod__JeodTrick10MemoryInterface

```
void init_attrjeod__JeodTrick10MemoryInterface ( )  [friend]
```

8.11.4.2 InputProcessor

```
friend class InputProcessor  [friend]
```

Definition at line 111 of file trick10_memory_interface.hh.

8.11.5 Field Documentation

8.11.5.1 trick_checkpoint_agent

```
Trick::ClassicCheckPointAgent* jeod::JeodTrick10MemoryInterface::trick_checkpoint_agent  [protected]
```

Trick checkpoint agent.

trick_io(**)

Definition at line 183 of file trick10_memory_interface.hh.

Referenced by JeodTrick10MemoryInterface(), and translate_addr_to_name().

The documentation for this class was generated from the following files:

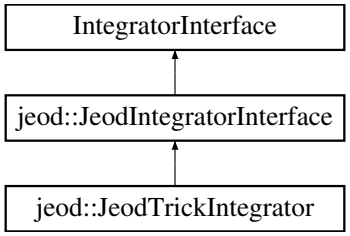
- trick10_memory_interface.hh
- trick10_memory_interface.cc
- trick_memory_interface_chkpnt.cc
- trick_memory_interface_xlate.cc

8.12 jeod::JeodTrickIntegrator Class Reference

A JeodTrickIntegrator specializes the JeodIntegratorInterface for use with Trick as the simulation engine.

`#include <jeod_trick_integrator.hh>`

Inheritance diagram for jeod::JeodTrickIntegrator:



Public Member Functions

- JeodTrickIntegrator ()

  Default constructor.
- virtual ~JeodTrickIntegrator ()

  Destructor.
- virtual er7_utils::Integration::Technique interpret_integration_type (int integ_technique) const

  Interpret the integration technique.
- virtual ::Trick::Integrator * get_integrator ()

  Get the simulation engine's integrator.
- virtual double get_dt () const

  Get the integration cycle time step.
- virtual bool get_first_step_derivs_flag () const

  Get the flag that tells the simulation engine to compute derivatives on the initial step of each integration cycle.
- virtual void set_first_step_derivs_flag (bool value)

  Set the flag that tells the simulation engine to compute derivatives on the initial step of each integration cycle.
- virtual void reset_first_step_derivs_flag ()

  Reset the flag that tells the simulation engine to compute derivatives on the initial step of each integration cycle.
- virtual void restore_first_step_derivs_flag ()

  Restore the flag that tells the simulation engine to compute derivatives on the initial step of each integration cycle to it's value prior to the most recent call to reset_first_step_derivs_flag.
- virtual void set_step_number (unsigned int stepno)

  Set the step number within an integration cycle.
- virtual void set_time (double sim_time)

  Update the time model given the simulation time.

Private Member Functions

- JeodTrickIntegrator (const JeodTrickIntegrator &)

  Not implemented.
- JeodTrickIntegrator & operator= (const JeodTrickIntegrator &)

  Not implemented.

Private Attributes

- TrickJeodIntegrator trick_integrator

  Trick integration structure.
- bool default_first_step_deriv

  Default value of trick_integrator.first_step_deriv.

Friends

- class InputProcessor
- void init_attrjeod__JeodTrickIntegrator ()

8.12.1 Detailed Description

A JeodTrickIntegrator specializes the JeodIntegratorInterface for use with Trick as the simulation engine.

Definition at line 119 of file jeod_trick_integrator.hh.

8.12.2 Constructor & Destructor Documentation

8.12.2.1 JeodTrickIntegrator() [1/2]

```
jeod::JeodTrickIntegrator::JeodTrickIntegrator ( )  [inline]
```

Default constructor.

Definition at line 129 of file jeod_trick_integrator.hh.

8.12.2.2 ~JeodTrickIntegrator()

```
virtual jeod::JeodTrickIntegrator::~JeodTrickIntegrator ( )  [inline], [virtual]
```

Destructor.

Definition at line 139 of file jeod_trick_integrator.hh.

8.12.2.3 JeodTrickIntegrator() [2/2]

```
jeod::JeodTrickIntegrator::JeodTrickIntegrator (
            const JeodTrickIntegrator &  )  [private]
```

Not implemented.

8.12.3 Member Function Documentation

8.12.3.1 get_dt()

```
virtual double jeod::JeodTrickIntegrator::get_dt ( ) const  [inline], [virtual]
```

Get the integration cycle time step.

Returns

Simulation time delta t, in seconds

Definition at line 165 of file jeod_trick_integrator.hh.

References trick_integrator.

8.12.3.2 get_first_step_derivs_flag()

```
virtual bool jeod::JeodTrickIntegrator::get_first_step_derivs_flag ( ) const  [inline], [virtual]
```

Get the flag that tells the simulation engine to compute derivatives on the initial step of each integration cycle.

Returns

Value of the first step derivatives flag

Definition at line 175 of file jeod_trick_integrator.hh.

References trick_integrator.

8.12.3.3 get_integrator()

```
virtual ::Trick::Integrator* jeod::JeodTrickIntegrator::get_integrator ( )  [inline], [virtual]
```

Get the simulation engine's integrator.

Returns

Pointer to the simulation engine's integrator.

Implements jeod::JeodIntegratorInterface.

Definition at line 156 of file jeod_trick_integrator.hh.

References trick_integrator.

8.12.3.4 interpret_integration_type()

```
virtual er7_utils::Integration::Technique jeod::JeodTrickIntegrator::interpret_integration_↩
type (
            int integ_technique ) const  [inline], [virtual]
```

Interpret the integration technique.

Implements jeod::JeodIntegratorInterface.

Definition at line 144 of file jeod_trick_integrator.hh.

8.12.3.5 operator=()

```
JeodTrickIntegrator& jeod::JeodTrickIntegrator::operator= (
            const JeodTrickIntegrator &  )  [private]
```

Not implemented.

8.12.3.6 reset_first_step_derivs_flag()

```
virtual void jeod::JeodTrickIntegrator::reset_first_step_derivs_flag ( )  [inline], [virtual]
```

Reset the flag that tells the simulation engine to compute derivatives on the initial step of each integration cycle.

Derivatives are always needed just after a reset. The behavior should revert to nominal after the reset has been performed.

Definition at line 196 of file jeod_trick_integrator.hh.

References default_first_step_deriv, and trick_integrator.

8.12.3.7 restore_first_step_derivs_flag()

```
virtual void jeod::JeodTrickIntegrator::restore_first_step_derivs_flag ( )  [inline], [virtual]
```

Restore the flag that tells the simulation engine to compute derivatives on the initial step of each integration cycle to it's value prior to the most recent call to reset_first_step_derivs_flag.

Definition at line 207 of file jeod_trick_integrator.hh.

References default_first_step_deriv, and trick_integrator.

8.12.3.8 set_first_step_derivs_flag()

```
virtual void jeod::JeodTrickIntegrator::set_first_step_derivs_flag (
            bool value )  [inline], [virtual]
```

Set the flag that tells the simulation engine to compute derivatives on the initial step of each integration cycle.

Parameters

in	*value*	Value of the first step derivatives flag

Definition at line 185 of file jeod_trick_integrator.hh.

References trick_integrator.

8.12.3.9 set_step_number()

```
virtual void jeod::JeodTrickIntegrator::set_step_number (
            unsigned int stepno )  [inline], [virtual]
```

Set the step number within an integration cycle.

Parameters

in	*stepno*	Step number

Definition at line 216 of file jeod_trick_integrator.hh.

References trick_integrator.

8.12.3.10 set_time()

```
virtual void jeod::JeodTrickIntegrator::set_time (
            double sim_time )  [inline], [virtual]
```

Update the time model given the simulation time.

Parameters

in	*sim_time*	Simulation time

Definition at line 225 of file jeod_trick_integrator.hh.

References trick_integrator.

8.12.4 Friends And Related Function Documentation

8.12.4.1 init_attrjeod__JeodTrickIntegrator

```
void init_attrjeod__JeodTrickIntegrator ( )  [friend]
```

8.12.4.2 InputProcessor

```
friend class InputProcessor  [friend]
```

Definition at line 120 of file jeod_trick_integrator.hh.

8.12.5 Field Documentation

8.12.5.1 default_first_step_deriv

```
bool jeod::JeodTrickIntegrator::default_first_step_deriv  [private]
```

Default value of trick_integrator.first_step_deriv.

trick_units(–)

Definition at line 243 of file jeod_trick_integrator.hh.

Referenced by reset_first_step_derivs_flag(), and restore_first_step_derivs_flag().

8.12.5.2 trick_integrator

```
TrickJeodIntegrator jeod::JeodTrickIntegrator::trick_integrator  [private]
```

Trick integration structure.

trick_units(–)

Definition at line 238 of file jeod_trick_integrator.hh.

Referenced by get_dt(), get_first_step_derivs_flag(), get_integrator(), reset_first_step_derivs_flag(), restore_first↩_step_derivs_flag(), set_first_step_derivs_flag(), set_step_number(), and set_time().

The documentation for this class was generated from the following file:

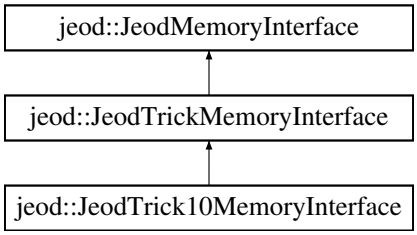
- jeod_trick_integrator.hh

8.13 jeod::JeodTrickMemoryInterface Class Reference

A TrickMemoryInterface implements the two required methods needed to register and deregister memory with the simulation engine, Trick in this case.

`#include <trick_memory_interface.hh>`

Inheritance diagram for jeod::JeodTrickMemoryInterface:



Data Structures

- struct AllocationMapEntry

  Describes a chunk of JEOD-allocated memory.
- struct ContainerListEntry

  Describes a Checkpointable object.

Public Member Functions

- JeodTrickMemoryInterface ()

  JeodTrickMemoryInterface default constructor.
- virtual ~JeodTrickMemoryInterface ()

  JeodTrickMemoryInterface destructor.
- void set_mode (JeodSimulationInterface::Mode new_mode)

  Set the mode and perform mode transitions.
- std::string construct_identifier (uint32_t unique_id_number)

  Construct an identifier for a chunk of JEOD-allocated memory.
- virtual const struct ATTRIBUTES_tag * find_attributes (const std::string &type_name) const

  Find the attributes for a class in the symbol table.
- virtual const struct ATTRIBUTES_tag * find_attributes (const std::type_info &data_type) const

  Find the attributes for a class in the symbol table.
- virtual struct ATTRIBUTES_tag primitive_attributes (const std::type_info &data_type) const

  Create an attributes structure that represents a primitive type.
- virtual struct ATTRIBUTES_tag pointer_attributes (const struct ATTRIBUTES_tag &target_attr) const

  Create an attributes structure that represents a pointer type.
- virtual struct ATTRIBUTES_tag void_pointer_attributes () const

  Create an attributes structure that represents a void pointer.*
- virtual struct ATTRIBUTES_tag structure_attributes (const struct ATTRIBUTES_tag *target_attr, std::size_t target_size) const

  Create an attributes structure that represents a structured type.
- virtual bool register_allocation (const void *addr, const JeodMemoryItem &item, const JeodMemoryType↩ Descriptor &tdesc, const char *file, unsigned int line)

  Register newly allocated memory with Trick.

- virtual void deregister_allocation (const void *addr, const JeodMemoryItem &item, const JeodMemoryType↩ Descriptor &tdesc, const char *file, unsigned int line)

  Delete Trick information about some pointer – but not the pointer itself.
- virtual void register_container (const void *owner, const JeodMemoryTypeDescriptor &owner_type, const char *elem_name, JeodCheckpointable &container)

  Register the checkpointable object with Trick.
- virtual void deregister_container (const void *owner, const JeodMemoryTypeDescriptor &owner_type, const char *elem_name, JeodCheckpointable &container)

  Revoke the registrations performed by register_container.
- virtual const std::string get_name_at_address (const void *addr, const JeodMemoryTypeDescriptor *tdesc) const

  Stubbed-out implementation of get_name_at_address for Trick implementations that do not fully support JEOD checkpoint/restart requirements.
- virtual void * get_address_at_name (const std::string &name) const

  Stubbed-out implementation of get_address_at_name for Trick implementations that do not fully support JEOD checkpoint/restart requirements.
- virtual bool is_checkpoint_restart_supported () const

  The generic Trick memory interface does not support checkpoint/restart.
- virtual const std::string get_trick_checkpoint_file (bool checkpoint)

  Get the name of the current Trick checkpoint file.
- virtual void checkpoint_containers ()

  Dump the container checkpointable objects to the checkpoint file.
- virtual void restore_containers ()

  Restore the container checkpointables objects from the checkpoint file.
- virtual void checkpoint_allocations ()

  Dump the allocation information to the checkpoint file.
- virtual void restore_allocations (JeodMemoryManager &memory_manager)

  Restore the allocated data per the checkpoint file.

Protected Types

- typedef std::map< uint32_t, AllocationMapEntry > AllocationMap

  Maps JEOD-allocated data names to (type, size) pairs.
- typedef std::list< ContainerListEntry > ContainerList

  Container of a list of ContainerListEntry objects.

Protected Attributes

- void * dlhandle

  dlhandle, from dlopen.
- AllocationMap allocation_map

  Map of allocated names to type info.
- ContainerList container_list

  List of container checkpointables.
- const std::string id_prefix

  Prefix used for constructing a unique name for JEOD-allocated memory.
- const uint32_t id_length

  Number of digits in the numeric part of the unique identifier.
- JeodSimulationInterface::Mode mode

  Simulation interface mode.

Private Member Functions

- JeodTrickMemoryInterface (const JeodTrickMemoryInterface &)

  Not implemented.
- JeodTrickMemoryInterface & operator= (const JeodTrickMemoryInterface &)

  Not implemented.

Friends

- class InputProcessor
- void init_attrjeod__JeodTrickMemoryInterface ()

8.13.1 Detailed Description

A TrickMemoryInterface implements the two required methods needed to register and deregister memory with the simulation engine, Trick in this case.

Definition at line 98 of file trick_memory_interface.hh.

8.13.2 Member Typedef Documentation

8.13.2.1 AllocationMap

```
typedef std::map<uint32_t, AllocationMapEntry> jeod::JeodTrickMemoryInterface::AllocationMap
[protected]
```

Maps JEOD-allocated data names to (type, size) pairs.

Definition at line 333 of file trick_memory_interface.hh.

8.13.2.2 ContainerList

```
typedef std::list<ContainerListEntry> jeod::JeodTrickMemoryInterface::ContainerList  [protected]
```

Container of a list of ContainerListEntry objects.

Definition at line 338 of file trick_memory_interface.hh.

8.13.3 Constructor & Destructor Documentation

8.13.3.1 JeodTrickMemoryInterface() [1/2]

```
jeod::JeodTrickMemoryInterface::JeodTrickMemoryInterface ( )
```

JeodTrickMemoryInterface default constructor.

Definition at line 55 of file trick_memory_interface.cc.

References dlhandle, and jeod::SimInterfaceMessages::implementation_error.

8.13.3.2 ~JeodTrickMemoryInterface()

```
jeod::JeodTrickMemoryInterface::~JeodTrickMemoryInterface ( )  [virtual]
```

JeodTrickMemoryInterface destructor.

Definition at line 76 of file trick_memory_interface.cc.

References dlhandle.

8.13.3.3 JeodTrickMemoryInterface() [2/2]

```
jeod::JeodTrickMemoryInterface::JeodTrickMemoryInterface (
            const JeodTrickMemoryInterface &  )  [private]
```

Not implemented.

8.13.4 Member Function Documentation

8.13.4.1 checkpoint_allocations()

```
virtual void jeod::JeodTrickMemoryInterface::checkpoint_allocations (
            void  )  [inline], [virtual]
```

Dump the allocation information to the checkpoint file.

Note

The default implementation does nothing; checkpoint/restart is not supported by default.

Reimplemented in jeod::JeodTrick10MemoryInterface.

Definition at line 227 of file trick_memory_interface.hh.

8.13.4.2 checkpoint_containers()

```
virtual void jeod::JeodTrickMemoryInterface::checkpoint_containers (
            void  )  [inline], [virtual]
```

Dump the container checkpointable objects to the checkpoint file.

Note

The default implementation does nothing; checkpoint/restart is not supported by default.

Reimplemented in jeod::JeodTrick10MemoryInterface.

Definition at line 211 of file trick_memory_interface.hh.

8.13.4.3 construct_identifier()

```
std::string jeod::JeodTrickMemoryInterface::construct_identifier (
            uint32_t unique_id_number )
```

Construct an identifier for a chunk of JEOD-allocated memory.

Returns

Identifier string

Parameters

unique_id_number	Identifier number

Definition at line 103 of file trick_memory_interface.cc.

References id_length, and id_prefix.

Referenced by jeod::JeodTrick10MemoryInterface::checkpoint_allocations(), and register_allocation().

8.13.4.4 deregister_allocation()

```
void jeod::JeodTrickMemoryInterface::deregister_allocation (
            const void * addr,
            const JeodMemoryItem & item,
            const JeodMemoryTypeDescriptor & tdesc,
            const char * file,
            unsigned int line )  [virtual]
```

Delete Trick information about some pointer – but not the pointer itself.

Assumptions and Limitations

- Some other agent must freeing the memory at the input address itself. This function merely deletes Trick's knowledge of that pointer.

Parameters

in	*addr*	Allocated memory
in	*item*	Description of the memory
in	*tdesc*	Description of the type
in	*file*	Source file containing JEOD_ALLOC
in	*line*	Line number containing JEOD_ALLOC

Implements jeod::JeodMemoryInterface.

Definition at line 138 of file trick_memory_interface_alloc.cc.

References allocation_map, jeod::SimInterfaceMessages::interface_error, and trick_MM.

8.13.4.5 deregister_container()

```
void jeod::JeodTrickMemoryInterface::deregister_container (
            const void * owner,
            const JeodMemoryTypeDescriptor & owner_type,
            const char * elem_name,
            JeodCheckpointable & container )  [virtual]
```

Revoke the registrations performed by register_container.

This function is typically called at destruction time via JEOD_DEREGISTER_CHECKPOINTABLE. This default implementation does nothing.

Parameters

in	*owner*	Owner of the container
in	*owner_type*	Owner type descriptor
in	*elem_name*	Container element
in,out	*container*	The container

Implements jeod::JeodMemoryInterface.

Reimplemented in jeod::JeodTrick10MemoryInterface.

Definition at line 148 of file trick_memory_interface.cc.

8.13.4.6 find_attributes() [1/2]

```
const struct ATTRIBUTES_tag * jeod::JeodTrickMemoryInterface::find_attributes (
            const std::string & type_name ) const  [virtual]
```

Find the attributes for a class in the symbol table.

Returns

Found attributes

Parameters

in	*type_name*	Demangled type name

Implements jeod::JeodMemoryInterface.

Definition at line 55 of file trick_memory_interface_attrib.cc.

References dlhandle, and jeod::SimInterfaceMessages::interface_error.

Referenced by find_attributes().

8.13.4.7 find_attributes() [2/2]

```
const struct ATTRIBUTES_tag * jeod::JeodTrickMemoryInterface::find_attributes (
            const std::type_info & data_type ) const  [virtual]
```

Find the attributes for a class in the symbol table.

Returns

Found attributes

Parameters

in	*data_type*	Data type descriptor

Implements jeod::JeodMemoryInterface.

Definition at line 86 of file trick_memory_interface_attrib.cc.

References find_attributes().

8.13.4.8 get_address_at_name()

```
void * jeod::JeodTrickMemoryInterface::get_address_at_name (
            const std::string & name ) const  [virtual]
```

Stubbed-out implementation of get_address_at_name for Trick implementations that do not fully support JEOD checkpoint/restart requirements.

Returns

Address of named item in memory

Parameters

name	Name of item to be found

Implements jeod::JeodMemoryInterface.

Reimplemented in jeod::JeodTrick10MemoryInterface.

Definition at line 182 of file trick_memory_interface.cc.

8.13.4.9 get_name_at_address()

```
const std::string jeod::JeodTrickMemoryInterface::get_name_at_address (
            const void * addr,
            const JeodMemoryTypeDescriptor * tdesc ) const  [virtual]
```

Stubbed-out implementation of get_name_at_address for Trick implementations that do not fully support JEOD checkpoint/restart requirements.

Returns

Name of the address, if any.

Parameters

addr	Address of memory whose name is to be found
tdesc	How to interpret address

Implements jeod::JeodMemoryInterface.

Reimplemented in jeod::JeodTrick10MemoryInterface.

Definition at line 165 of file trick_memory_interface.cc.

8.13.4.10 get_trick_checkpoint_file()

```
virtual const std::string jeod::JeodTrickMemoryInterface::get_trick_checkpoint_file (
            bool checkpoint )  [inline], [virtual]
```

Get the name of the current Trick checkpoint file.

Parameters

in	*checkpoint*	True for checkpoint, false for restart

Returns

Current checkpoint file, or the empty string.

Note

The default implementation always returns the empty string; checkpoint/restart is not supported by default.

Reimplemented in jeod::JeodTrick10MemoryInterface.

Definition at line 199 of file trick_memory_interface.hh.

8.13.4.11 is_checkpoint_restart_supported()

```
virtual bool jeod::JeodTrickMemoryInterface::is_checkpoint_restart_supported (
            void  ) const  [inline], [virtual]
```

The generic Trick memory interface does not support checkpoint/restart.

Implements jeod::JeodMemoryInterface.

Reimplemented in jeod::JeodTrick10MemoryInterface.

Definition at line 189 of file trick_memory_interface.hh.

8.13.4.12 operator=()

```
JeodTrickMemoryInterface& jeod::JeodTrickMemoryInterface::operator= (
            const JeodTrickMemoryInterface &  )  [private]
```

Not implemented.

8.13.4.13 pointer_attributes()

```
struct ATTRIBUTES_tag jeod::JeodTrickMemoryInterface::pointer_attributes (
            const struct ATTRIBUTES_tag & target_attr ) const  [virtual]
```

Create an attributes structure that represents a pointer type.

Returns

Constructed pointer attributes.

Parameters

in	*target_attr*	Pointed-to type attributes.

Implements jeod::JeodMemoryInterface.

Definition at line 206 of file trick_memory_interface_attrib.cc.

Referenced by jeod::JeodTrick10MemoryInterface::translate_addr_to_name().

8.13.4.14 primitive_attributes()

```
struct ATTRIBUTES_tag jeod::JeodTrickMemoryInterface::primitive_attributes (
            const std::type_info & data_type ) const  [virtual]
```

Create an attributes structure that represents a primitive type.

Returns

Constructed attributes.

Parameters

in	*data_type*	Data type descriptor

Implements jeod::JeodMemoryInterface.

Definition at line 100 of file trick_memory_interface_attrib.cc.

References jeod::SimInterfaceMessages::interface_error.

8.13.4.15 register_allocation()

```
bool jeod::JeodTrickMemoryInterface::register_allocation (
            const void * addr,
            const JeodMemoryItem & item,
            const JeodMemoryTypeDescriptor & tdesc,
            const char * file,
            unsigned int line )  [virtual]
```

Register newly allocated memory with Trick.

Assumptions and Limitations

- Memory was indeed allocated.
- The input address is not null.
- The number of elements is positive.

Returns

True if registered

Parameters

in	*addr*	Allocated memory
in	*item*	Description of the memory
in	*tdesc*	Description of the type
in	*file*	Source file containing JEOD_ALLOC
in	*line*	Line number containing JEOD_ALLOC

Implements jeod::JeodMemoryInterface.

Definition at line 75 of file trick_memory_interface_alloc.cc.

References allocation_map, construct_identifier(), jeod::SimInterfaceMessages::interface_error, and trick_MM.

8.13.4.16 register_container()

```
void jeod::JeodTrickMemoryInterface::register_container (
            const void * owner,
            const JeodMemoryTypeDescriptor & owner_type,
            const char * elem_name,
            JeodCheckpointable & container )  [virtual]
```

Register the checkpointable object with Trick.

This function is typically called at construction or initialization time via JEOD_REGISTER_CHECKPOINTABLE. This default implementation does nothing.

Parameters

in	*owner*	Owner of the container
in	*owner_type*	Owner type descriptor
in	*elem_name*	Container element
in,out	*container*	The container

Implements jeod::JeodMemoryInterface.

Reimplemented in jeod::JeodTrick10MemoryInterface.

Definition at line 127 of file trick_memory_interface.cc.

8.13.4.17 restore_allocations()

```
virtual void jeod::JeodTrickMemoryInterface::restore_allocations (
            JeodMemoryManager & memory_manager )  [inline], [virtual]
```

Restore the allocated data per the checkpoint file.

Parameters

memory_manager	JEOD memory manager

Note

The default implementation does nothing; checkpoint/restart is not supported by default.

Reimplemented in jeod::JeodTrick10MemoryInterface.

Definition at line 236 of file trick_memory_interface.hh.

8.13.4.18 restore_containers()

```
virtual void jeod::JeodTrickMemoryInterface::restore_containers (
            void  )  [inline], [virtual]
```

Restore the container checkpointables objects from the checkpoint file.

Note

The default implementation does nothing; checkpoint/restart is not supported by default.

Reimplemented in jeod::JeodTrick10MemoryInterface.

Definition at line 219 of file trick_memory_interface.hh.

8.13.4.19 set_mode()

```
void jeod::JeodTrickMemoryInterface::set_mode (
            JeodSimulationInterface::Mode new_mode )
```

Set the mode and perform mode transitions.

Parameters

new_mode	New mode

Definition at line 90 of file trick_memory_interface.cc.

References mode.

Referenced by jeod::BasicJeodTrickSimInterface::set_mode().

8.13.4.20 structure_attributes()

```
struct ATTRIBUTES_tag jeod::JeodTrickMemoryInterface::structure_attributes (
            const struct ATTRIBUTES_tag * target_attr,
            std::size_t target_size ) const  [virtual]
```

Create an attributes structure that represents a structured type.

Returns

Constructed structure attributes.

Parameters

in	*target_attr*	Return value from find_attributes.
in	*target_size*	Structure size.

Implements jeod::JeodMemoryInterface.

Definition at line 279 of file trick_memory_interface_attrib.cc.

8.13.4.21 void_pointer_attributes()

```
struct ATTRIBUTES_tag jeod::JeodTrickMemoryInterface::void_pointer_attributes (
            void  ) const  [virtual]
```

Create an attributes structure that represents a void* pointer.

Returns

Constructed pointer attributes.

Implements jeod::JeodMemoryInterface.

Definition at line 254 of file trick_memory_interface_attrib.cc.

8.13.5 Friends And Related Function Documentation

8.13.5.1 init_attrjeod__JeodTrickMemoryInterface

```
void init_attrjeod__JeodTrickMemoryInterface ( )  [friend]
```

8.13.5.2 InputProcessor

```
friend class InputProcessor  [friend]
```

Definition at line 100 of file trick_memory_interface.hh.

8.13.6 Field Documentation

8.13.6.1 allocation_map

```
AllocationMap jeod::JeodTrickMemoryInterface::allocation_map  [protected]
```

Map of allocated names to type info.

trick_io(**)

Definition at line 351 of file trick_memory_interface.hh.

Referenced by jeod::JeodTrick10MemoryInterface::checkpoint_allocations(), deregister_allocation(), and register↩
_allocation().

8.13.6.2 container_list

```
ContainerList jeod::JeodTrickMemoryInterface::container_list  [protected]
```

List of container checkpointables.

trick_io(**)

Definition at line 356 of file trick_memory_interface.hh.

Referenced by jeod::JeodTrick10MemoryInterface::checkpoint_allocations(), jeod::JeodTrick10MemoryInterface↩
::checkpoint_containers(), jeod::JeodTrick10MemoryInterface::deregister_container(), jeod::JeodTrick10Memory↩
Interface::register_container(), jeod::JeodTrick10MemoryInterface::restore_allocations(), and jeod::JeodTrick10↩
MemoryInterface::restore_containers().

8.13.6.3 dlhandle

```
void* jeod::JeodTrickMemoryInterface::dlhandle  [protected]
```

dlhandle, from dlopen.

trick_io(**)

Definition at line 346 of file trick_memory_interface.hh.

Referenced by find_attributes(), JeodTrickMemoryInterface(), and ~JeodTrickMemoryInterface().

8.13.6.4 id_length

```
const uint32_t jeod::JeodTrickMemoryInterface::id_length  [protected]
```

Number of digits in the numeric part of the unique identifier.

trick_io(*o) trick_units(–)

Definition at line 366 of file trick_memory_interface.hh.

Referenced by construct_identifier().

8.13.6.5 id_prefix

```
const std::string jeod::JeodTrickMemoryInterface::id_prefix  [protected]
```

Prefix used for constructing a unique name for JEOD-allocated memory.

trick_io(*o) trick_units(–)

Definition at line 361 of file trick_memory_interface.hh.

Referenced by construct_identifier().

8.13.6.6 mode

```
JeodSimulationInterface::Mode jeod::JeodTrickMemoryInterface::mode  [protected]
```

Simulation interface mode.

trick_units(–)

Definition at line 371 of file trick_memory_interface.hh.

Referenced by set_mode().

The documentation for this class was generated from the following files:

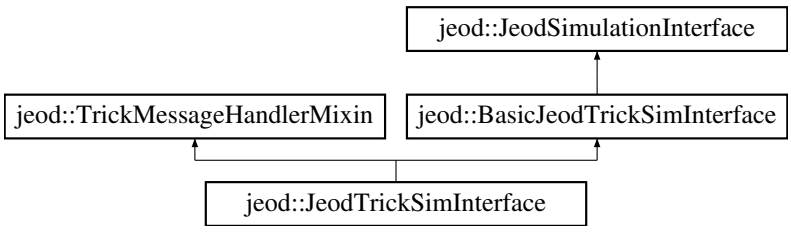
- trick_memory_interface.hh
- trick_memory_interface.cc
- trick_memory_interface_alloc.cc
- trick_memory_interface_attrib.cc

8.14 jeod::JeodTrickSimInterface Class Reference

A JEODTrickSimInterface implements the required capabilities of the generic JeodSimulationInterface in a Trick simulation environment.

`#include <trick_sim_interface.hh>`

Inheritance diagram for jeod::JeodTrickSimInterface:



Public Member Functions

- JeodTrickSimInterface ()

  Non-default constructor.
- virtual ~JeodTrickSimInterface ()

  Destructor.

Private Member Functions

- JeodTrickSimInterface (const JeodTrickSimInterface &)

  Not implemented.
- JeodTrickSimInterface & operator= (const JeodTrickSimInterface &)

  Not implemented.

Friends

- class InputProcessor
- void init_attrjeod__JeodTrickSimInterface ()

Additional Inherited Members

8.14.1 Detailed Description

A JEODTrickSimInterface implements the required capabilities of the generic JeodSimulationInterface in a Trick simulation environment.

By virtue of member data ownership, the class creates the requisite MessageHandler and MemoryManager and does so in the correct order.

Definition at line 296 of file trick_sim_interface.hh.

8.14.2 Constructor & Destructor Documentation

8.14.2.1 JeodTrickSimInterface() [1/2]

```
jeod::JeodTrickSimInterface::JeodTrickSimInterface ( )  [inline], [explicit]
```

Non-default constructor.

Definition at line 305 of file trick_sim_interface.hh.

8.14.2.2 ~JeodTrickSimInterface()

```
virtual jeod::JeodTrickSimInterface::~JeodTrickSimInterface ( )  [inline], [virtual]
```

Destructor.

Definition at line 311 of file trick_sim_interface.hh.

8.14.2.3 JeodTrickSimInterface() [2/2]

```
jeod::JeodTrickSimInterface::JeodTrickSimInterface (
            const JeodTrickSimInterface &  )  [private]
```

Not implemented.

8.14.3 Member Function Documentation

8.14.3.1 operator=()

```
JeodTrickSimInterface& jeod::JeodTrickSimInterface::operator= (
            const JeodTrickSimInterface &  )  [private]
```

Not implemented.

8.14.4 Friends And Related Function Documentation

8.14.4.1 init_attrjeod__JeodTrickSimInterface

```
void init_attrjeod__JeodTrickSimInterface ( )  [friend]
```

8.14.4.2 InputProcessor

```
friend class InputProcessor  [friend]
```

Definition at line 298 of file trick_sim_interface.hh.

The documentation for this class was generated from the following file:

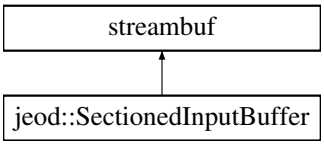
- trick_sim_interface.hh

8.15 jeod::SectionedInputBuffer Class Reference

A SectionedInputBuffer is a std::streambuf that reads from a section in a checkpoint file.

`#include <checkpoint_input_manager.hh>`

Inheritance diagram for jeod::SectionedInputBuffer:



Public Member Functions

- ~SectionedInputBuffer ()

  Destructor.
- bool operator! () const

  Conversion to boolean.

Private Member Functions

- SectionedInputBuffer (void)

  Default constructor.
- void activate (std::ifstream &stream, std::size_t spos, std::size_t epos)

  Activate the object.
- void deactivate (void)

  Deactivate the object.
- virtual std::streambuf::int_type underflow ()

  Get a character in the case of depletion of the read buffer.
- SectionedInputBuffer (const SectionedInputBuffer &)

  Not implemented.
- SectionedInputBuffer & operator= (const SectionedInputBuffer &)

  Not implemented.

Private Attributes

- std::filebuf * file_buf

  The file buffer that reads from the checkpoint file.
- size_t start_pos

  The position of the start of the contents of the checkpoint file section being read by this object.
- size_t end_pos

  The position just after the end of the contents of the checkpoint file section being read by this object.
- size_t curr_pos

  The current position of the file_buf reader.
- bool at_eof

  At EOF in the file or in the section?
- char buf

  Input buffer.

Friends

- class SectionedInputStream

8.15.1 Detailed Description

A SectionedInputBuffer is a std::streambuf that reads from a section in a checkpoint file.

This class will indicate EOF when the input pointer in the checkpoint file file buffer goes beyond the end of the section.

This is a barebones implementation. It does not provide buffering, it does not support seek and tell, and it does not support putback or unget.

Note that with the exception of the destructor and the inherited members from std::streambuf, *everything* in this class is private. This class is not extensible.

Definition at line 89 of file checkpoint_input_manager.hh.

8.15.2 Constructor & Destructor Documentation

8.15.2.1 ~SectionedInputBuffer()

```
jeod::SectionedInputBuffer::~SectionedInputBuffer ( )  [inline]
```

Destructor.

For now, this does nothing.

Definition at line 98 of file checkpoint_input_manager.hh.

8.15.2.2 SectionedInputBuffer() [1/2]

```
jeod::SectionedInputBuffer::SectionedInputBuffer (
            void  )  [private]
```

Default constructor.

This constructor creates an empty SectionedInputBuffer – one that will return EOF on the first read attempt. An empty SectionedInputBuffer has two purposes:

- As the basis for a copy constructor of a containing stream, and
- As a graceful means of handling of erroneous conditions.

Definition at line 45 of file checkpoint_input_manager.cc.

8.15.2.3 SectionedInputBuffer() [2/2]

```
jeod::SectionedInputBuffer::SectionedInputBuffer (
            const SectionedInputBuffer &  )  [private]
```

Not implemented.

8.15.3 Member Function Documentation

8.15.3.1 activate()

```
void jeod::SectionedInputBuffer::activate (
            std::ifstream & stream,
            std::size_t spos,
            std::size_t epos )  [private]
```

Activate the object.

Note

Using the object for reading prior to activation will result in EOF.

Parameters

in	*stream*	Checkpoint file input file stream
in	*spos*	Section data start position
in	*epos*	Section data end position

Definition at line 68 of file checkpoint_input_manager.cc.

References at_eof, curr_pos, end_pos, file_buf, and start_pos.

Referenced by jeod::SectionedInputStream::activate().

8.15.3.2 deactivate()

```
void jeod::SectionedInputBuffer::deactivate (
             void  )  [inline], [private]
```

Deactivate the object.

Used to force a badly behaving stream to disconnect.

Definition at line 123 of file checkpoint_input_manager.hh.

References at_eof, and file_buf.

Referenced by jeod::SectionedInputStream::deactivate().

8.15.3.3 operator"!()

```
bool jeod::SectionedInputBuffer::operator!  ( ) const  [inline]
```

Conversion to boolean.

Returns

False if object is OK.

Definition at line 105 of file checkpoint_input_manager.hh.

References file_buf.

8.15.3.4 operator=()

```
SectionedInputBuffer& jeod::SectionedInputBuffer::operator= (
             const SectionedInputBuffer &  )  [private]
```

Not implemented.

8.15.3.5 underflow()

```
std::streambuf::int_type jeod::SectionedInputBuffer::underflow (
            void  )  [private], [virtual]
```

Get a character in the case of depletion of the read buffer.

For now, the buffer is always depleted.

Returns

Character read from the underlying file.

Definition at line 86 of file checkpoint_input_manager.cc.

References at_eof, buf, curr_pos, end_pos, and file_buf.

8.15.4 Friends And Related Function Documentation

8.15.4.1 SectionedInputStream

```
friend class SectionedInputStream  [friend]
```

Definition at line 90 of file checkpoint_input_manager.hh.

8.15.5 Field Documentation

8.15.5.1 at_eof

```
bool jeod::SectionedInputBuffer::at_eof  [private]
```

At EOF in the file or in the section?

trick_io(**)

Definition at line 159 of file checkpoint_input_manager.hh.

Referenced by activate(), deactivate(), and underflow().

8.15.5.2 buf

```
char jeod::SectionedInputBuffer::buf  [private]
```

Input buffer.

trick_io(**)

Definition at line 164 of file checkpoint_input_manager.hh.

Referenced by underflow().

8.15.5.3 curr_pos

```
size_t jeod::SectionedInputBuffer::curr_pos  [private]
```

The current position of the file_buf reader.

trick_io(**)

Definition at line 154 of file checkpoint_input_manager.hh.

Referenced by activate(), and underflow().

8.15.5.4 end_pos

```
size_t jeod::SectionedInputBuffer::end_pos  [private]
```

The position just after the end of the contents of the checkpoint file section being read by this object.

trick_io(**)

Definition at line 149 of file checkpoint_input_manager.hh.

Referenced by activate(), and underflow().

8.15.5.5 file_buf

```
std::filebuf* jeod::SectionedInputBuffer::file_buf  [private]
```

The file buffer that reads from the checkpoint file.

trick_io(**)

Definition at line 137 of file checkpoint_input_manager.hh.

Referenced by activate(), deactivate(), operator!(), and underflow().

8.15.5.6 start_pos

```
size_t jeod::SectionedInputBuffer::start_pos  [private]
```

The position of the start of the contents of the checkpoint file section being read by this object.

trick_io(**)

Definition at line 143 of file checkpoint_input_manager.hh.

Referenced by activate().

The documentation for this class was generated from the following files:

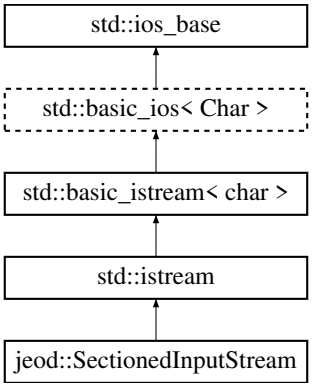
- checkpoint_input_manager.hh
- checkpoint_input_manager.cc

8.16 jeod::SectionedInputStream Class Reference

A SectionedInputStream is a std::istream that reads from a section in a checkpoint file.

```
#include <checkpoint_input_manager.hh>
```

Inheritance diagram for jeod::SectionedInputStream:



Public Member Functions

- SectionedInputStream ()

  Construct a SectionedInputStream object.
- SectionedInputStream (const SectionedInputStream &)

  Construct a SectionedInputStream object by copying from another.
- ~SectionedInputStream ()

  Destruct a SectionedInputStream object.
- bool is_activatable () const

  Determine if the stream is able to be activated.
- bool activate ()

  Activate the object.
- void deactivate (void)

  Deactivate the object.
- bool operator! () const

  Conversion to boolean.
- operator void * () const

  Conversion to void.*

Private Member Functions

- SectionedInputStream (CheckPointInputManager *mngr, std::ifstream &fstream, std::size_t spos, std::size_t epos)

  Construct a SectionedInputStream object that is connected to a file stream and to a CheckPointInputManager.
- SectionedInputStream & operator= (const SectionedInputStream &)

  Not implemented.

Private Attributes

- SectionedInputBuffer sectbuf

  The std::streambuf that does the reading from the file.
- CheckPointInputManager * manager

  The input manager that created this object.
- std::ifstream * stream

  The C++ file stream that reads from the checkpoint file.
- size_t start_pos

  The position of the start of the contents of the checkpoint file section being read by this object.
- size_t end_pos

  The position just after the end of the contents of the checkpoint file section being read by this object.
- bool is_copy

  Is this a copy of some other SectionedInputStream? Copies of copies are verboten.
- bool is_active

  Is this an active object? In the end, there can be only one.

Friends

- class CheckPointInputManager

8.16.1 Detailed Description

A SectionedInputStream is a std::istream that reads from a section in a checkpoint file.

This class will indicate EOF when the input pointer in the checkpoint file file buffer goes beyond the end of the section.

This is a barebones implementation. It does not provide buffering, it does not support seek and tell, and it does not support putback or unget.

Usage

A SectionedInputStream object is used in a preload_checkpoint or restart job to read and then act on contents stored in a checkpoint file.

```
return_type function_name (
   SomeStructureType & stuff_to_restore)
{
   std::string section_name;
   double number;
   char c_style_line[256];
   std::string cpp_line;
   char character;
   int char_as_int;

   std::string section_name;
   // Set to name of the checkpoint section

   // Construct a checkpoint input stream.
   // Notes:
   // - This object must go out of scope by the end of the job.
   // - DO NOT make a copy of this object.
   // - DO NOT save a pointer to this object in a permanent structure.
   // - The code below assumes that function_name is called as a
   //   preload_checkpoint or a restart job.
  SectionedInputStream reader (
     JeodSimulationInterface::get_checkpoint_reader(
      section_name));

  // Activate the reader.
  // Fail to do so and you'll get EOF on the first read.
  reader.activate();

  // You can use the C++ operator >> to read various kinds of data ...
  reader >> number;

  // ... even data structures if the structure has a deserializer.
  reader >> stuff_to_restore;

  // Lines can be read with the getline member or std::getline global.
  reader.getline (c_style_line, 255);
  std::getline (reader, cpp_line);

  // Individual characters can be read in a variety of ways.
  reader >> std::noskipws >> character;
  reader.get (character);
  char_as_int = reader.rdbuf()->sbumpc();

  // A bunch of numbers can be read using operator >>:
  while (!! (reader >> number)) {
     stuff_to_restore.add_number (number);
  }

  // An alternative is to implicitly use operator void*:
  while (reader >> number) {
     stuff_to_restore.add_number (number);
  }

  // The file can be scanned via getline, here using the bang-bang trick:
  while (!! std::getline (reader, cpp_string)) {
     process_line (cpp_string);
  }

  // Same as the above, but implicitly using operator void*:
  while (std::getline (reader, cpp_string)) {
     process_line (cpp_string);
  }

  // The file can be processed a character at a time.
  // Once again, either the bang-bang trick or operator void* can be
  // used to check for EOF.
  while (!! std::get (reader, character)) {
     stuff_to_restore.add_char (character);
  }

  // Yet another alternative is to test for EOF using sbumpc:
  while ((char_as_int = rdbuf->sbumpc()) != EOF) {
     stuff_to_restore.add_char ((char)char_as_int);
  }

  // Or use sgetc/sbumpc if the above grates too much:
  while (reader.rdbuf->sgetc()) != EOF) {
     stuff_to_restore.add_char ((char)reader.rdbuf()->sbumpc());
  }
}
```

Diagnosing problems

- Nothing is being read. This can be caused by several problems, described below.

- Is the JEOD checkpoint file open for input?
  Checkpoint file sections can only be read from a JEOD checkpoint file that is open for input. In a Trick context, the checkpoint file is only open for preload_checkpoint and restart jobs. Reading from a checkpoint file in other contexts won't work.
- Are multiple threads trying to read from the same checkpoint file?
  Don't do that. This package is not thread-safe.
- Have you cached some another active checkpoint reader somewhere?
  Don't do that, either. Only one reader can be active at a time.
- Is the checkpoint file section in the checkpoint file?
  You will get a diagnostic message if the section doesn't exist.
- Is the checkpoint reader viable?
  The above problems will result in a non-viable checkpoint reader. The method is_activatable() can be called prior to calling activate() to check whether the stream is viable.
- Did you call reader.activate()?
  Whether compilers make two different objects in the construction of the SectionedInputStream or just one object depends on the compiler and on the optimization level. Making the package robustly handle the complexities of RVO (return value optimization) was too much for the author of the package. The call to reader.activate() is essential.
- Did the call to reader.activate() work?
  The method activate() returns true or false to indicate success or failure. While the above code did not check status, doing so is a good idea.
- Did you call reader.deactivate()?
  Don't do that until you are done reading. The call to deactivate() is irreversible.
- Did you mix scanned input with line reading?
  As with any other stream, operator >> will mark the stream as failed if the operator fails to parse.

Definition at line 309 of file checkpoint_input_manager.hh.

8.16.2 Constructor & Destructor Documentation

8.16.2.1 SectionedInputStream() [1/3]

```
jeod::SectionedInputStream::SectionedInputStream ( )
```

Construct a SectionedInputStream object.

Note

This default constructor creates a disconnected and hence unusable stream. Usable streams are created by the non-default constructor.

Definition at line 130 of file checkpoint_input_manager.cc.

8.16.2.2 SectionedInputStream() [2/3]

```
jeod::SectionedInputStream::SectionedInputStream (
            const SectionedInputStream & source )
```

Construct a SectionedInputStream object by copying from another.

Parameters

in	*source*	Source object

Definition at line 176 of file checkpoint_input_manager.cc.

References jeod::SimInterfaceMessages::implementation_error, is_active, is_copy, manager, and stream.

8.16.2.3 ~SectionedInputStream()

```
jeod::SectionedInputStream::~SectionedInputStream (
            void  )
```

Destruct a SectionedInputStream object.

Definition at line 204 of file checkpoint_input_manager.cc.

References jeod::CheckPointInputManager::deregister_reader(), is_active, and manager.

8.16.2.4 SectionedInputStream() [3/3]

```
jeod::SectionedInputStream::SectionedInputStream (
            CheckPointInputManager * mngr,
            std::ifstream & ifstream,
            std::size_t spos,
            std::size_t epos )  [private]
```

Construct a SectionedInputStream object that is connected to a file stream and to a CheckPointInputManager.

Parameters

in	*mngr*	The stream manager
in	*ifstream*	The input file stream
in	*spos*	Start position of section data
in	*epos*	End position of section data

Definition at line 153 of file checkpoint_input_manager.cc.

8.16.3 Member Function Documentation

8.16.3.1 activate()

```
bool jeod::SectionedInputStream::activate (
            void  )
```

Activate the object.

Note

Using the object for reading prior to activation will result in EOF.

Returns

True if activated.

Definition at line 246 of file checkpoint_input_manager.cc.

References jeod::SectionedInputBuffer::activate(), end_pos, jeod::SimInterfaceMessages::implementation_error, is_active, manager, jeod::CheckPointInputManager::register_reader(), sectbuf, start_pos, and stream.

Referenced by jeod::JeodTrick10MemoryInterface::restore_allocations(), and jeod::JeodTrick10MemoryInterface::restore_containers().

8.16.3.2 deactivate()

```
void jeod::SectionedInputStream::deactivate (
            void  )
```

Deactivate the object.

Note

Deactivation is undoable.

Definition at line 292 of file checkpoint_input_manager.cc.

References jeod::SectionedInputBuffer::deactivate(), jeod::CheckPointInputManager::deregister_reader(), is_active, manager, sectbuf, and stream.

Referenced by jeod::CheckPointInputManager::create_trick_section_reader(), and jeod::JeodTrick10MemoryInterface::restore_containers().

8.16.3.3 is_activatable()

```
bool jeod::SectionedInputStream::is_activatable (
            void  ) const
```

Determine if the stream is able to be activated.

Returns

True if object can be activated.

Definition at line 219 of file checkpoint_input_manager.cc.

References jeod::CheckPointInputManager::have_active_reader(), is_active, manager, and stream.

8.16.3.4 operator void *()

```
jeod::SectionedInputStream::operator void * ( ) const  [inline]
```

Conversion to void*.

This method provides an alternative to the bang-bang trick to determine if the object is OK.

Returns

this pointer (cast to void*) if object is OK, NULL otherwise.

Definition at line 346 of file checkpoint_input_manager.hh.

8.16.3.5 operator"!()

```
bool jeod::SectionedInputStream::operator!  ( ) const  [inline]
```

Conversion to boolean.

Use the bang-bang trick to determine if the object is OK.

Returns

False if object is OK, true if something is wrong.

Definition at line 337 of file checkpoint_input_manager.hh.

References is_active, sectbuf, and stream.

8.16.3.6 operator=()

```
SectionedInputStream& jeod::SectionedInputStream::operator= (
            const SectionedInputStream &  )  [private]
```

Not implemented.

8.16.4 Friends And Related Function Documentation

8.16.4.1 CheckPointInputManager

```
friend class CheckPointInputManager  [friend]
```

Definition at line 310 of file checkpoint_input_manager.hh.

8.16.5 Field Documentation

8.16.5.1 end_pos

```
size_t jeod::SectionedInputStream::end_pos  [private]
```

The position just after the end of the contents of the checkpoint file section being read by this object.

trick_io(**)

Definition at line 387 of file checkpoint_input_manager.hh.

Referenced by activate().

8.16.5.2 is_active

```
bool jeod::SectionedInputStream::is_active  [private]
```

Is this an active object? In the end, there can be only one.

trick_io(**)

Definition at line 399 of file checkpoint_input_manager.hh.

Referenced by activate(), deactivate(), is_activatable(), operator!(), SectionedInputStream(), and ~SectionedInputStream().

8.16.5.3 is_copy

```
bool jeod::SectionedInputStream::is_copy  [private]
```

Is this a copy of some other SectionedInputStream? Copies of copies are verboten.

trick_io(**)

Definition at line 393 of file checkpoint_input_manager.hh.

Referenced by SectionedInputStream().

8.16.5.4 manager

```
CheckPointInputManager* jeod::SectionedInputStream::manager  [private]
```

The input manager that created this object.

trick_io(**)

Definition at line 370 of file checkpoint_input_manager.hh.

Referenced by activate(), deactivate(), is_activatable(), SectionedInputStream(), and ~SectionedInputStream().

8.16.5.5 sectbuf

```
SectionedInputBuffer jeod::SectionedInputStream::sectbuf  [private]
```

The std::streambuf that does the reading from the file.

trick_io(**)

Definition at line 365 of file checkpoint_input_manager.hh.

Referenced by activate(), deactivate(), and operator!().

8.16.5.6 start_pos

```
size_t jeod::SectionedInputStream::start_pos  [private]
```

The position of the start of the contents of the checkpoint file section being read by this object.

trick_io(**)

Definition at line 381 of file checkpoint_input_manager.hh.

Referenced by activate().

8.16.5.7 stream

```
std::ifstream* jeod::SectionedInputStream::stream  [private]
```

The C++ file stream that reads from the checkpoint file.

trick_io(**)

Definition at line 375 of file checkpoint_input_manager.hh.

Referenced by activate(), deactivate(), is_activatable(), operator!(), and SectionedInputStream().

The documentation for this class was generated from the following files:

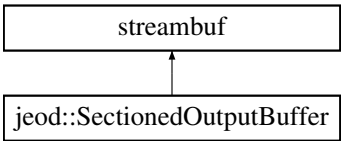
- checkpoint_input_manager.hh
- checkpoint_input_manager.cc

8.17 jeod::SectionedOutputBuffer Class Reference

A SectionedOutputBuffer is a std::streambuf that writes a section of a checkpoint file.

```
#include <checkpoint_output_manager.hh>
```

Inheritance diagram for jeod::SectionedOutputBuffer:



Public Member Functions

- ~SectionedOutputBuffer ()

  Destructor.
- bool operator! () const

  Conversion to boolean.

Private Member Functions

- SectionedOutputBuffer (void)

  Default constructor.
- SectionedOutputBuffer (std::ofstream *stream)
- void activate (std::ofstream &stream)

  Activate the object.
- void deactivate (void)

  Deactivate the object.
- virtual std::streambuf::int_type overflow (std::streambuf::int_type c)

  Write a character in the case of overflow of the write buffer.
- SectionedOutputBuffer (const SectionedOutputBuffer &)

  Not implemented.
- SectionedOutputBuffer & operator= (const SectionedOutputBuffer &)

  Not implemented.

Private Attributes

- std::filebuf * file_buf

  The file buffer that writes to the checkpoint file.

Friends

- class SectionedOutputStream

8.17.1 Detailed Description

A SectionedOutputBuffer is a std::streambuf that writes a section of a checkpoint file.

This is a barebones implementation. It does not provide buffering, and it does not support seek and tell.

Note that with the exception of the destructor and the inherited members from std::streambuf, *everything* in this class is private. This class is not extensible.

Definition at line 86 of file checkpoint_output_manager.hh.

8.17.2 Constructor & Destructor Documentation

8.17.2.1 ~SectionedOutputBuffer()

```
jeod::SectionedOutputBuffer::~SectionedOutputBuffer ( )  [inline]
```

Destructor.

For now, this does nothing.

Definition at line 95 of file checkpoint_output_manager.hh.

8.17.2.2 SectionedOutputBuffer() [1/3]

```
jeod::SectionedOutputBuffer::SectionedOutputBuffer (
            void  )  [private]
```

Default constructor.

This constructor creates an empty SectionedOutputBuffer – one that will return EOF on the first write attempt. An empty SectionedOutputBuffer has two purposes:

- As the basis for a copy constructor of a containing stream, and
- As a graceful means of handling of erroneous conditions.

Definition at line 45 of file checkpoint_output_manager.cc.

8.17.2.3 SectionedOutputBuffer() [2/3]

```
jeod::SectionedOutputBuffer::SectionedOutputBuffer (
            std::ofstream * stream )  [explicit], [private]
```

8.17.2.4 SectionedOutputBuffer() [3/3]

```
jeod::SectionedOutputBuffer::SectionedOutputBuffer (
            const SectionedOutputBuffer &  )  [private]
```

Not implemented.

8.17.3 Member Function Documentation

8.17.3.1 activate()

```
void jeod::SectionedOutputBuffer::activate (
            std::ofstream & stream )  [private]
```

Activate the object.

Note

Using the object for writing prior to activation will result in EOF.

Parameters

in	*stream*	Output file stream

Definition at line 61 of file checkpoint_output_manager.cc.

References file_buf.

Referenced by jeod::SectionedOutputStream::activate().

8.17.3.2 deactivate()

```
void jeod::SectionedOutputBuffer::deactivate (
            void  )  [inline], [private]
```

Deactivate the object.

Used to disconnect the buffer when the stream is done, sometimes by force.

Definition at line 121 of file checkpoint_output_manager.hh.

References file_buf.

Referenced by jeod::SectionedOutputStream::deactivate().

8.17.3.3 operator"!()

```
bool jeod::SectionedOutputBuffer::operator!  ( ) const  [inline]
```

Conversion to boolean.

Returns

False if object is OK.

Definition at line 102 of file checkpoint_output_manager.hh.

References file_buf.

8.17.3.4 operator=()

```
SectionedOutputBuffer& jeod::SectionedOutputBuffer::operator= (
            const SectionedOutputBuffer &  )  [private]
```

Not implemented.

8.17.3.5 overflow()

```
std::streambuf::int_type jeod::SectionedOutputBuffer::overflow (
            std::streambuf::int_type ch )  [private], [virtual]
```

Write a character in the case of overflow of the write buffer.

For now, the buffer always overflows.

Returns

Status: EOF => failed

Parameters

in	*ch*	Character to be writter

Definition at line 76 of file checkpoint_output_manager.cc.

References file_buf.

8.17.4 Friends And Related Function Documentation

8.17.4.1 SectionedOutputStream

```
friend class SectionedOutputStream  [friend]
```

Definition at line 87 of file checkpoint_output_manager.hh.

8.17.5 Field Documentation

8.17.5.1 file_buf

```
std::filebuf* jeod::SectionedOutputBuffer::file_buf  [private]
```

The file buffer that writes to the checkpoint file.

trick_io(**)

Definition at line 137 of file checkpoint_output_manager.hh.

Referenced by activate(), deactivate(), operator!(), and overflow().

The documentation for this class was generated from the following files:

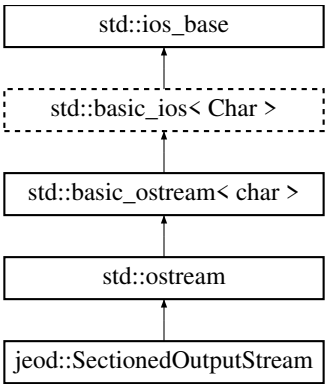
- checkpoint_output_manager.hh
- checkpoint_output_manager.cc

8.18 jeod::SectionedOutputStream Class Reference

A SectionedOutputStream is a std::ostream that writes a section of a checkpoint file.

`#include <checkpoint_output_manager.hh>`

Inheritance diagram for jeod::SectionedOutputStream:



Public Member Functions

- SectionedOutputStream ()

  Construct a SectionedOutputStream object.
- SectionedOutputStream (const SectionedOutputStream &)

  Construct a SectionedOutputStream object by copying from another.
- ~SectionedOutputStream ()

  Destruct a SectionedOutputStream object.
- bool is_activatable () const

  Determine if the stream is able to be activated.
- bool activate ()

  Activate the object.
- void deactivate ()

  Deactivate the object.
- bool operator! () const

  Conversion to boolean.
- operator void * () const

  Conversion to void.*

Private Member Functions

- SectionedOutputStream (CheckPointOutputManager *mngr, std::ofstream &ofstream, const std::string &start_marker, const std::string &end_marker, const std::string &section_name)

  Construct a SectionedOutputStream object that is connected to a file stream and to a CheckPointOutputManager.
- SectionedOutputStream & operator= (const SectionedOutputStream &)

  Not implemented.

Private Attributes

- SectionedOutputBuffer sectbuf

  The std::streambuf that does the writing to the file.
- CheckPointOutputManager * manager

  The input manager that created this object.
- std::ofstream * stream

  The C++ file stream that writes to the checkpoint file.
- const std::string * section_start

  The string that indicates the start of a checkpoint file section.
- const std::string * section_end

  The string that indicates the start of a checkpoint file section.
- const std::string tag

  The name of the checkpoint file section.
- bool is_copy

  Is this a copy of some other SectionedOutputStream? Copies of copies are verboten.
- bool is_active

  Is this an active object? In the end, there can be only one.

Friends

- class CheckPointOutputManager

8.18.1 Detailed Description

A SectionedOutputStream is a std::ostream that writes a section of a checkpoint file.

This class automatically writes the start and end markers. Standard C++ output mechanisms can be used to write the contents of the section.

This is a barebones implementation. It does not provide buffering, it does not support seek and tell, and it does not support putback or unget.

Note that most of the content of this class is private. This class is not extensible and is intended to be used within the context of a CheckPointOutputManager.

Definition at line 168 of file checkpoint_output_manager.hh.

8.18.2 Constructor & Destructor Documentation

8.18.2.1 SectionedOutputStream() [1/3]

```
jeod::SectionedOutputStream::SectionedOutputStream ( )
```

Construct a SectionedOutputStream object.

Note

This default constructor creates a disconnected and hence unusable stream. Usable streams are created by the non-default constructor.

Definition at line 118 of file checkpoint_output_manager.cc.

8.18.2.2 SectionedOutputStream() [2/3]

```
jeod::SectionedOutputStream::SectionedOutputStream (
            const SectionedOutputStream & source )
```

Construct a SectionedOutputStream object by copying from another.

Parameters

in	*source*	Source object

Definition at line 168 of file checkpoint_output_manager.cc.

References jeod::SimInterfaceMessages::implementation_error, is_active, is_copy, manager, and stream.

8.18.2.3 ~SectionedOutputStream()

```
jeod::SectionedOutputStream::~SectionedOutputStream (
            void  )
```

Destruct a SectionedOutputStream object.

Definition at line 197 of file checkpoint_output_manager.cc.

References deactivate().

8.18.2.4 SectionedOutputStream() [3/3]

```
jeod::SectionedOutputStream::SectionedOutputStream (
            CheckPointOutputManager * mngr,
            std::ofstream & ofstream,
            const std::string & start_marker,
            const std::string & end_marker,
            const std::string & section_name )  [private]
```

Construct a SectionedOutputStream object that is connected to a file stream and to a CheckPointOutputManager.

Parameters

in	*mngr*	The stream manager
in	*ofstream*	The output file stream
in	*start_marker*	Start of section marker
in	*end_marker*	End of section marker
in	*section_name*	Name of the section

Definition at line 143 of file checkpoint_output_manager.cc.

8.18.3 Member Function Documentation

8.18.3.1 activate()

```
bool jeod::SectionedOutputStream::activate (
             void  )
```

Activate the object.

Note

Using the object for writing prior to activation will write nothing.

Returns

True if activated.

Definition at line 237 of file checkpoint_output_manager.cc.

References jeod::SectionedOutputBuffer::activate(), jeod::SimInterfaceMessages::implementation_error, is_active, manager, jeod::CheckPointOutputManager::register_writer(), sectbuf, section_start, stream, and tag.

Referenced by jeod::JeodTrick10MemoryInterface::checkpoint_allocations(), and jeod::JeodTrick10Memory↩Interface::checkpoint_containers().

8.18.3.2 deactivate()

```
void jeod::SectionedOutputStream::deactivate (
             void  )
```

Deactivate the object.

Note

Deactivation is undoable.

Definition at line 291 of file checkpoint_output_manager.cc.

References jeod::SectionedOutputBuffer::deactivate(), jeod::CheckPointOutputManager::deregister_writer(), is_↩active, manager, sectbuf, section_end, stream, and tag.

Referenced by jeod::JeodTrick10MemoryInterface::checkpoint_containers(), jeod::CheckPointOutputManager↩::create_trick_section_writer(), and ∼SectionedOutputStream().

8.18.3.3 is_activatable()

```
bool jeod::SectionedOutputStream::is_activatable (
             void  ) const
```

Determine if the stream is able to be activated.

Returns

True if object can be activated.

Definition at line 210 of file checkpoint_output_manager.cc.

References jeod::CheckPointOutputManager::have_active_writer(), is_active, manager, and stream.

8.18.3.4 operator void ∗()

```
jeod::SectionedOutputStream::operator void * ( ) const  [inline]
```

Conversion to void∗.

This method provides an alternative to the bang-bang trick to determine if the object is OK.

Returns

this pointer (cast to void∗) if object is OK, NULL otherwise.

Definition at line 204 of file checkpoint_output_manager.hh.

8.18.3.5 operator"!()

```
bool jeod::SectionedOutputStream::operator!  ( ) const  [inline]
```

Conversion to boolean.

Returns

False if object is OK.

Definition at line 195 of file checkpoint_output_manager.hh.

References is_active, sectbuf, and stream.

8.18.3.6 operator=()

```
SectionedOutputStream& jeod::SectionedOutputStream::operator= (
            const SectionedOutputStream &  )  [private]
```

Not implemented.

8.18.4 Friends And Related Function Documentation

8.18.4.1 CheckPointOutputManager

```
friend class CheckPointOutputManager  [friend]
```

Definition at line 169 of file checkpoint_output_manager.hh.

8.18.5 Field Documentation

8.18.5.1 is_active

```
bool jeod::SectionedOutputStream::is_active  [private]
```

Is this an active object? In the end, there can be only one.

trick_io(**)

Definition at line 261 of file checkpoint_output_manager.hh.

Referenced by activate(), deactivate(), is_activatable(), operator!(), and SectionedOutputStream().

8.18.5.2 is_copy

```
bool jeod::SectionedOutputStream::is_copy  [private]
```

Is this a copy of some other SectionedOutputStream? Copies of copies are verboten.

trick_io(**)

Definition at line 255 of file checkpoint_output_manager.hh.

Referenced by SectionedOutputStream().

8.18.5.3 manager

```
CheckPointOutputManager* jeod::SectionedOutputStream::manager  [private]
```

The input manager that created this object.

trick_io(**)

Definition at line 229 of file checkpoint_output_manager.hh.

Referenced by activate(), deactivate(), is_activatable(), and SectionedOutputStream().

8.18.5.4 sectbuf

```
SectionedOutputBuffer jeod::SectionedOutputStream::sectbuf  [private]
```

The std::streambuf that does the writing to the file.

trick_io(**)

Definition at line 224 of file checkpoint_output_manager.hh.

Referenced by activate(), deactivate(), and operator!().

8.18.5.5 section_end

```
const std::string* jeod::SectionedOutputStream::section_end  [private]
```

The string that indicates the start of a checkpoint file section.

Definition at line 244 of file checkpoint_output_manager.hh.

Referenced by deactivate().

8.18.5.6 section_start

```
const std::string* jeod::SectionedOutputStream::section_start  [private]
```

The string that indicates the start of a checkpoint file section.

Definition at line 239 of file checkpoint_output_manager.hh.

Referenced by activate().

8.18.5.7 stream

```
std::ofstream* jeod::SectionedOutputStream::stream  [private]
```

The C++ file stream that writes to the checkpoint file.

trick_io(**)

Definition at line 234 of file checkpoint_output_manager.hh.

Referenced by activate(), deactivate(), is_activatable(), operator!(), and SectionedOutputStream().

8.18.5.8 tag

```
const std::string jeod::SectionedOutputStream::tag  [private]
```

The name of the checkpoint file section.

Definition at line 249 of file checkpoint_output_manager.hh.

Referenced by activate(), and deactivate().

The documentation for this class was generated from the following files:

- checkpoint_output_manager.hh
- checkpoint_output_manager.cc

8.19 jeod::CheckPointInputManager::SectionInfo Struct Reference

A SectionInfo contains the start and end positions of a checkpoint file section.

Public Member Functions

- SectionInfo (std::size_t start, std::size_t end)

  Non-default constructor.

Data Fields

- size_t start_pos

  Position of the first readable character of a section.
- size_t end_pos

  Position of the first unreadable character after a section.

8.19.1 Detailed Description

A SectionInfo contains the start and end positions of a checkpoint file section.

Definition at line 472 of file checkpoint_input_manager.hh.

8.19.2 Constructor & Destructor Documentation

8.19.2.1 SectionInfo()

```
jeod::CheckPointInputManager::SectionInfo::SectionInfo (
            std::size_t start,
            std::size_t end ) [inline]
```

Non-default constructor.

Parameters

in	*start*	Start position
in	*end*	End position

Definition at line 488 of file checkpoint_input_manager.hh.

8.19.3 Field Documentation

8.19.3.1 end_pos

```
size_t jeod::CheckPointInputManager::SectionInfo::end_pos
```

Position of the first unreadable character after a section.

trick_io(**)

Definition at line 481 of file checkpoint_input_manager.hh.

Referenced by jeod::CheckPointInputManager::create_section_reader().

8.19.3.2 start_pos

```
size_t jeod::CheckPointInputManager::SectionInfo::start_pos
```

Position of the first readable character of a section.

trick_io(**)

Definition at line 476 of file checkpoint_input_manager.hh.

Referenced by jeod::CheckPointInputManager::create_section_reader().

The documentation for this struct was generated from the following file:

- checkpoint_input_manager.hh

8.20 jeod::SimInterfaceMessages Class Reference

Specifies the message IDs used in the sim_interface model.

```
#include <sim_interface_messages.hh>
```

Static Public Attributes

- static char const * singleton_error = "utils/sim_interface/" "singleton_error"

  Message issued when multiple instance of a class that should be a singleton are created or when no such instance exists (but should).
- static char const * interface_error = "utils/sim_interface/" "interface_error"

  Message issued when issues arise from interacting with the sim engine.
- static char const * phasing_error = "utils/sim_interface/" "phasing_error"

  Message issued when things happen out of order.
- static char const * integration_error = "utils/sim_interface/" "integration_error"

  Message issued when something goes awry with integration.
- static char const * implementation_error = "utils/sim_interface/" "implementation_error"

  Message issued when something went wrong with the implementation.

Private Member Functions

- SimInterfaceMessages (void)
- SimInterfaceMessages (const SimInterfaceMessages &)
- SimInterfaceMessages & operator= (const SimInterfaceMessages &)

8.20.1 Detailed Description

Specifies the message IDs used in the sim_interface model.

Definition at line 79 of file sim_interface_messages.hh.

8.20.2 Constructor & Destructor Documentation

8.20.2.1 SimInterfaceMessages() [1/2]

```
jeod::SimInterfaceMessages::SimInterfaceMessages (
            void ) [private]
```

8.20.2.2 SimInterfaceMessages() [2/2]

```
jeod::SimInterfaceMessages::SimInterfaceMessages (
            const SimInterfaceMessages & ) [private]
```

8.20.3 Member Function Documentation

8.20.3.1 operator=()

```
SimInterfaceMessages& jeod::SimInterfaceMessages::operator= (
            const SimInterfaceMessages & ) [private]
```

8.20.4 Field Documentation

8.20.4.1 implementation_error

```
char const * jeod::SimInterfaceMessages::implementation_error = "utils/sim_interface/" "implementation↩
_error" [static]
```

Message issued when something went wrong with the implementation.

trick_units(–)

Definition at line 109 of file sim_interface_messages.hh.

Referenced by jeod::SectionedOutputStream::activate(), jeod::SectionedInputStream::activate(), jeod::Check↩PointInputManager::CheckPointInputManager(), jeod::CheckPointOutputManager::CheckPointOutputManager(), jeod::CheckPointInputManager::create_section_reader(), jeod::CheckPointOutputManager::create_section_↩writer(), jeod::CheckPointInputManager::initialize(), jeod::JeodTrickMemoryInterface::JeodTrickMemoryInterface(), jeod::SectionedInputStream::SectionedInputStream(), jeod::SectionedOutputStream::SectionedOutputStream(), and jeod::JeodSimulationInterface::set_mode().

8.20.4.2 integration_error

```
char const * jeod::SimInterfaceMessages::integration_error = "utils/sim_interface/" "integration↩
_error"  [static]
```

Message issued when something goes awry with integration.

trick_units(–)

Definition at line 104 of file sim_interface_messages.hh.

Referenced by jeod::JeodDynbodyIntegrationLoop::add_integrable_object(), jeod::JeodDynbodyIntegrationLoop↩
::initialize_integ_loop(), jeod::JeodDynbodyIntegrationLoop::integrate_dt(), jeod::JeodDynbodyIntegrationLoop::↩
JeodDynbodyIntegrationLoop(), and jeod::JeodDynbodyIntegrationLoop::update_integration_group().

8.20.4.3 interface_error

```
char const * jeod::SimInterfaceMessages::interface_error = "utils/sim_interface/" "interface_↩
error"  [static]
```

Message issued when issues arise from interacting with the sim engine.

trick_units(–)

Definition at line 94 of file sim_interface_messages.hh.

Referenced by jeod::JeodTrick10MemoryInterface::checkpoint_allocations(), jeod::JeodTrick10MemoryInterface↩
::checkpoint_containers(), jeod::JeodTrickMemoryInterface::deregister_allocation(), jeod::JeodTrick10Memory↩
Interface::deregister_container(), jeod::JeodTrickMemoryInterface::find_attributes(), jeod::JeodTrick10Memory↩
Interface::JeodTrick10MemoryInterface(), jeod::JeodTrickMemoryInterface::primitive_attributes(), jeod::Jeod↩
TrickMemoryInterface::register_allocation(), jeod::JeodTrick10MemoryInterface::register_container(), jeod::Jeod↩
Trick10MemoryInterface::restore_allocations(), jeod::JeodTrick10MemoryInterface::restore_containers(), jeod↩
::JeodTrick10MemoryInterface::translate_addr_to_name(), and jeod::JeodTrick10MemoryInterface::translate_↩
name_to_addr().

8.20.4.4 phasing_error

```
char const * jeod::SimInterfaceMessages::phasing_error = "utils/sim_interface/" "phasing_↩
error"  [static]
```

Message issued when things happen out of order.

trick_units(–)

Definition at line 99 of file sim_interface_messages.hh.

Referenced by jeod::BasicJeodTrickSimInterface::get_checkpoint_reader_internal(), jeod::BasicJeodTrickSim↩
Interface::get_checkpoint_writer_internal(), and jeod::JeodSimulationInterface::set_mode().

8.20.4.5 singleton_error

```
char const * jeod::SimInterfaceMessages::singleton_error = "utils/sim_interface/" "singleton_↩
error"  [static]
```

Message issued when multiple instance of a class that should be a singleton are created or when no such instance exists (but should).

trick_units(–)

Definition at line 89 of file sim_interface_messages.hh.

Referenced by jeod::JeodSimulationInterface::create_integrator_interface(), jeod::JeodSimulationInterface::get↩_address_at_name(), jeod::JeodSimulationInterface::get_checkpoint_reader(), jeod::JeodSimulationInterface↩::get_checkpoint_writer(), jeod::JeodSimulationInterface::get_job_cycle(), jeod::JeodSimulationInterface::get_↩memory_interface(), jeod::JeodSimulationInterface::get_name_at_address(), and jeod::JeodSimulationInterface↩::JeodSimulationInterface().

The documentation for this class was generated from the following files:

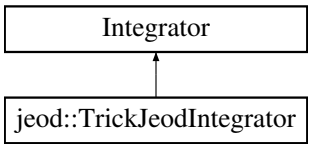
- sim_interface_messages.hh
- sim_interface_messages.cc

8.21 jeod::TrickJeodIntegrator Class Reference

A TrickJeodIntegrator specializes the Trick::Integrator to provide the Trick side of the integration interface between Trick and JEOD.

```
#include <jeod_trick_integrator.hh>
```

Inheritance diagram for jeod::TrickJeodIntegrator:



Public Member Functions

- virtual ~TrickJeodIntegrator ()

  Destructor.
- int integrate ()

  Does nothing.
- void initialize (int, double)

  Does nothing.

8.21.1 Detailed Description

A TrickJeodIntegrator specializes the Trick::Integrator to provide the Trick side of the integration interface between Trick and JEOD.

Definition at line 83 of file jeod_trick_integrator.hh.

8.21.2 Constructor & Destructor Documentation

8.21.2.1 ~TrickJeodIntegrator()

```
virtual jeod::TrickJeodIntegrator::~TrickJeodIntegrator ( )  [inline], [virtual]
```

Destructor.

Definition at line 96 of file jeod_trick_integrator.hh.

8.21.3 Member Function Documentation

8.21.3.1 initialize()

```
void jeod::TrickJeodIntegrator::initialize (
             int ,
             double  )  [inline]
```

Does nothing.

Definition at line 111 of file jeod_trick_integrator.hh.

8.21.3.2 integrate()

```
int jeod::TrickJeodIntegrator::integrate ( )  [inline]
```

Does nothing.

Definition at line 106 of file jeod_trick_integrator.hh.

The documentation for this class was generated from the following file:

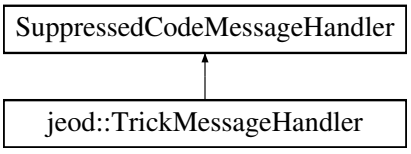
- jeod_trick_integrator.hh

8.22 jeod::TrickMessageHandler Class Reference

The MessageHandler class for designed for use in Trick-based simulations.

```
#include <trick_message_handler.hh>
```

Inheritance diagram for jeod::TrickMessageHandler:



Public Member Functions

- TrickMessageHandler (void)

  Default constructor.
- virtual ~TrickMessageHandler (void)

  Destructor.
- virtual void register_contents (void)

  Register the TrickMessageHandler's checkpointable contents.

Protected Member Functions

- virtual void process_message (int severity, const char *prefix, const char *file, unsigned int line, const char *msg_code, const char *format, va_list args) const

  Handle a message.

Private Member Functions

- TrickMessageHandler (const TrickMessageHandler &)

  Not implemented.
- TrickMessageHandler & operator= (const TrickMessageHandler &)

  Not implemented.

Friends

- class InputProcessor
- void init_attrjeod__TrickMessageHandler ()

8.22.1 Detailed Description

The MessageHandler class for designed for use in Trick-based simulations.

Definition at line 91 of file trick_message_handler.hh.

8.22.2 Constructor & Destructor Documentation

8.22.2.1 TrickMessageHandler() [1/2]

```
jeod::TrickMessageHandler::TrickMessageHandler (
            void  )  [inline]
```

Default constructor.

Definition at line 108 of file trick_message_handler.hh.

8.22.2.2 ~TrickMessageHandler()

```
virtual jeod::TrickMessageHandler::~TrickMessageHandler (
            void  )  [inline], [virtual]
```

Destructor.

Definition at line 113 of file trick_message_handler.hh.

8.22.2.3 TrickMessageHandler() [2/2]

```
jeod::TrickMessageHandler::TrickMessageHandler (
            const TrickMessageHandler &  )  [private]
```

Not implemented.

8.22.3 Member Function Documentation

8.22.3.1 operator=()

```
TrickMessageHandler& jeod::TrickMessageHandler::operator= (
            const TrickMessageHandler &  )  [private]
```

Not implemented.

8.22.3.2 process_message()

```
void jeod::TrickMessageHandler::process_message (
            int severity,
            const char * prefix,
            const char * file,
            unsigned int line,
            const char * msg_code,
            const char * format,
            va_list args ) const  [protected], [virtual]
```

Handle a message.

All calls to the message-generating MessageHandler methods eventually result in a call to thisTrickMessage↩ Handler::process_message method. This method uses the Trick function exec_terminate to process fatal errors. The Trick function send_hs is used for all non-fatal messages, but only if the message severity is at or below the message suppression level.

Parameters

in	*severity*	Severity level
in	*prefix*	Message prefix (e.g., Error)
in	*file*	Typically **FILE**
in	*line*	Typically **LINE**
in	*msg_code*	Message code
in	*format*	sprintf format
in	*args*	Arguments

Definition at line 86 of file trick_message_handler.cc.

References MAX_MSG_SIZE.

8.22.3.3 register_contents()

```
void jeod::TrickMessageHandler::register_contents (
             void  )  [virtual]
```

Register the TrickMessageHandler's checkpointable contents.

Definition at line 62 of file trick_message_handler.cc.

8.22.4 Friends And Related Function Documentation

8.22.4.1 init_attrjeod__TrickMessageHandler

```
void init_attrjeod__TrickMessageHandler ( )  [friend]
```

8.22.4.2 InputProcessor

```
friend class InputProcessor  [friend]
```

Definition at line 92 of file trick_message_handler.hh.

The documentation for this class was generated from the following files:

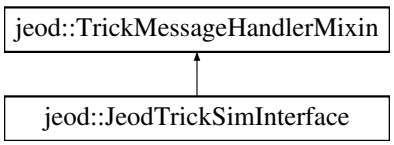
- trick_message_handler.hh
- trick_message_handler.cc

8.23 jeod::TrickMessageHandlerMixin Class Reference

The TrickMessageHandlerMixin implements the required capabilities of the generic JeodSimulationInterface in a Trick simulation environment.

`#include <trick_sim_interface.hh>`

Inheritance diagram for jeod::TrickMessageHandlerMixin:



Public Member Functions

- TrickMessageHandlerMixin ()

  Default constructor.
- virtual ~TrickMessageHandlerMixin ()

  Destructor.

Protected Attributes

- TrickMessageHandler message_handler

  The global MessageHandler.

Private Member Functions

- TrickMessageHandlerMixin (const TrickMessageHandlerMixin &)

  Not implemented.
- TrickMessageHandlerMixin & operator= (const TrickMessageHandlerMixin &)

  Not implemented.

Friends

- class InputProcessor
- void init_attrjeod__TrickMessageHandlerMixin ()

8.23.1 Detailed Description

The TrickMessageHandlerMixin implements the required capabilities of the generic JeodSimulationInterface in a Trick simulation environment.

By virtue of member data ownership, the class creates the requisite MessageHandler and MemoryManager and does so in the correct order.

Definition at line 247 of file trick_sim_interface.hh.

8.23.2 Constructor & Destructor Documentation

8.23.2.1 TrickMessageHandlerMixin() [1/2]

```
jeod::TrickMessageHandlerMixin::TrickMessageHandlerMixin ( )  [inline]
```

Default constructor.

Definition at line 255 of file trick_sim_interface.hh.

8.23.2.2 ~TrickMessageHandlerMixin()

```
virtual jeod::TrickMessageHandlerMixin::~TrickMessageHandlerMixin ( )  [inline], [virtual]
```

Destructor.

Definition at line 260 of file trick_sim_interface.hh.

8.23.2.3 TrickMessageHandlerMixin() [2/2]

```
jeod::TrickMessageHandlerMixin::TrickMessageHandlerMixin (
            const TrickMessageHandlerMixin &  )  [private]
```

Not implemented.

8.23.3 Member Function Documentation

8.23.3.1 operator=()

```
TrickMessageHandlerMixin& jeod::TrickMessageHandlerMixin::operator= (
            const TrickMessageHandlerMixin &  )  [private]
```

Not implemented.

8.23.4 Friends And Related Function Documentation

8.23.4.1 init_attrjeod__TrickMessageHandlerMixin

```
void init_attrjeod__TrickMessageHandlerMixin ( )  [friend]
```

8.23.4.2 InputProcessor

```
friend class InputProcessor  [friend]
```

Definition at line 248 of file trick_sim_interface.hh.

8.23.5 Field Documentation

8.23.5.1 message_handler

```
TrickMessageHandler jeod::TrickMessageHandlerMixin::message_handler  [protected]
```

The global MessageHandler.

trick_units(–)

Definition at line 269 of file trick_sim_interface.hh.

The documentation for this class was generated from the following file:

- trick_sim_interface.hh

Chapter 9

File Documentation

9.1 checkpoint_input_manager.cc File Reference

Define CheckPointInputManager member functions and of related classes.

```
#include <cstddef>
#include <cstring>
#include <iostream>
#include "utils/message/include/message_handler.hh"
#include "../include/checkpoint_input_manager.hh"
#include "../include/sim_interface_messages.hh"
```

Namespaces

- jeod

  Namespace jeod.

9.1.1 Detailed Description

Define CheckPointInputManager member functions and of related classes.

9.2 checkpoint_input_manager.hh File Reference

Define class CheckPointInputManager and related classes.

```
#include "utils/sim_interface/include/config.hh"
#include "utils/sim_interface/include/jeod_class.hh"
#include <cstddef>
#include <istream>
#include <fstream>
#include <string>
#include <map>
```

Data Structures

- class jeod::SectionedInputBuffer

  A SectionedInputBuffer is a std::streambuf that reads from a section in a checkpoint file.
- class jeod::SectionedInputStream

  A SectionedInputStream is a std::istream that reads from a section in a checkpoint file.
- class jeod::CheckPointInputManager

  A CheckPointInputManager provides tools for reading a checkpoint file.
- struct jeod::CheckPointInputManager::SectionInfo

  A SectionInfo contains the start and end positions of a checkpoint file section.

Namespaces

- jeod

  Namespace jeod.

9.2.1 Detailed Description

Define class CheckPointInputManager and related classes.

9.3 checkpoint_output_manager.cc File Reference

Define CheckPointOutputManager member functions and of related classes.

```
#include <cstddef>
#include <cstring>
#include <iostream>
#include "utils/message/include/message_handler.hh"
#include "../include/checkpoint_output_manager.hh"
#include "../include/sim_interface_messages.hh"
```

Namespaces

- jeod

  Namespace jeod.

9.3.1 Detailed Description

Define CheckPointOutputManager member functions and of related classes.

9.4 checkpoint_output_manager.hh File Reference

Define class CheckPointOutputManager and related classes.

```
#include "utils/sim_interface/include/jeod_class.hh"
#include <ostream>
#include <fstream>
#include <string>
#include <map>
```

Data Structures

- class jeod::SectionedOutputBuffer

  A SectionedOutputBuffer is a std::streambuf that writes a section of a checkpoint file.
- class jeod::SectionedOutputStream

  A SectionedOutputStream is a std::ostream that writes a section of a checkpoint file.
- class jeod::CheckPointOutputManager

  A CheckPointOutputManager provides the basic tools for writing a checkpoint file.

Namespaces

- jeod

  Namespace jeod.

9.4.1 Detailed Description

Define class CheckPointOutputManager and related classes.

9.5 class_declarations.hh File Reference

Forward declarations of classes defined in the utils/sim_interface model.

Namespaces

- jeod

  Namespace jeod.

9.5.1 Detailed Description

Forward declarations of classes defined in the utils/sim_interface model.

9.6 config.hh File Reference

Configure JEOD for use by some simulation engine.

```
#include "config_trick10.hh"
```

Macros

- #define JEOD_UNUSED
- #define ER7_UTILS_UNUSED
- #define ER7_UTILS_RESTRICT
- #define ER7_UTILS_ALWAYS_INLINE

9.6.1 Detailed Description

Configure JEOD for use by some simulation engine.

9.7 config_test_harness.hh File Reference

Configure JEOD for use in standalone test mode.

Macros

- #define JEOD_ATTRIBUTES_TYPE int
- #define JEOD_ATTRIBUTES_POINTER_TYPE void *
- #define JEOD_SIM_INTEGRATOR_POINTER_TYPE void *

9.7.1 Detailed Description

Configure JEOD for use in standalone test mode.

9.8 config_trick10.hh File Reference

Configure JEOD for use in a Trick10 environment.

Macros

- #define JEOD_SIZE_T size_t
- #define JEOD_PTRDIFF_T long int
- #define JEOD_INTPTR_T long int
- #define JEOD_UINTPTR_T unsigned long int
- #define JEOD_CLASS_ESTABLISH_FRIENDS(class_name)
- #define JEOD_ATTRIBUTES_SIM_ENGINE_HEADER "sim_services/MemoryManager/include/attributes.h"
- #define JEOD_ATTRIBUTES_TYPE struct ATTRIBUTES_tag
- #define JEOD_ATTRIBUTES_POINTER_TYPE JEOD_ATTRIBUTES_TYPE *
- #define JEOD_SIM_INTEGRATOR_SIM_ENGINE_HEADER "sim_services/Integrator/include/Integrator.hh"
- #define JEOD_SIM_INTEGRATOR_FORWARD namespace Trick { class Integrator; }
- #define JEOD_SIM_INTEGRATOR_POINTER_TYPE Trick::Integrator *
- #define JEOD_SIM_INTEGRATOR_ENUM Integrator_type

9.8.1 Detailed Description

Configure JEOD for use in a Trick10 environment.

9.9 jeod_class.hh File Reference

Define the JEOD class declaration macros JEOD_MAKE_SIM_INTERFACES and and JEOD_DECLARE_SIM_I↩ NTERFACES.

```
#include "config.hh"
```

Macros

- #define JEOD_MAKE_SIM_INTERFACES(class_name) JEOD_CLASS_ESTABLISH_FRIENDS(class_↩ name)

  JEOD_MAKE_SIM_INTERFACES(class_name) Defines friends of the given class.
- #define JEOD_DECLARE_SIM_INTERFACES(class_name)

  JEOD_DECLARE_SIM_INTERFACES(class_name) Forward declare classes and external functions needed to make the JEOD_MAKE_SIM_INTERFACES(class_name) expansion compilable.

9.9.1 Detailed Description

Define the JEOD class declaration macros JEOD_MAKE_SIM_INTERFACES and and JEOD_DECLARE_SIM_I↩ NTERFACES.

All JEOD class definitions must invoke JEOD_MAKE_SIM_INTERFACES within the body of the class. Corresponding invocations of JEOD_DECLARE_SIM_INTERFACES Are made at file scope and in the context of the global namespace.

In a Trick environment, these macros gives the Trick input processor, the Trick checkpoint / checkpoint-restart facility, and the ICG-generated io_src file for the header full visibility of the class's contents. The intent is to provide the same capability outside the Trick.

9.10 jeod_integrator_interface.hh File Reference

Define the interface for accessing / updating elements of a simulation engine's integrator object.

```
#include "utils/sim_interface/include/config.hh"
#include "utils/sim_interface/include/jeod_class.hh"
#include "er7_utils/integration/core/include/integration_technique.hh"
#include "er7_utils/integration/core/include/integrator_interface.hh"
```

Data Structures

- class jeod::JeodIntegratorInterface

  A JeodIntegratorInterface extends the ER7 IntegratorInterface with the concept of a pointer to the simulation engine's integration object.

Namespaces

- Trick

  Namespace Trick furnishes several standard functions for use in the Trick environment.
- jeod

  Namespace jeod.

9.10.1 Detailed Description

Define the interface for accessing / updating elements of a simulation engine's integrator object.

9.11 jeod_trick_integrator.hh File Reference

Define the interface for accessing / updating elements of a Trick simulation integrator object.

```
#include "sim_services/Integrator/include/Integrator.hh"
#include "er7_utils/trick/integration/include/translate_trick_integ_type.↩
hh"
#include "jeod_class.hh"
#include "jeod_integrator_interface.hh"
```

Data Structures

- class jeod::TrickJeodIntegrator

  A TrickJeodIntegrator specializes the Trick::Integrator to provide the Trick side of the integration interface between Trick and JEOD.
- class jeod::JeodTrickIntegrator

  A JeodTrickIntegrator specializes the JeodIntegratorInterface for use with Trick as the simulation engine.

Namespaces

- jeod

  Namespace jeod.

9.11.1 Detailed Description

Define the interface for accessing / updating elements of a Trick simulation integrator object.

9.12 memory_attributes.hh File Reference

Define JEOD memory interface macros.

```
#include "config.hh"
#include "sim_services/MemoryManager/include/attributes.h"
```

Namespaces

- jeod

  Namespace jeod.

Macros

- #define JEOD_DECLARE_ATTRIBUTES(class_name)

  JEOD_DECLARE_ATTRIBUTES(class_name) This macro is obsolete.
- #define JEOD_ATTRIBUTES(type) JeodSimulationInterface::get_memory_interface().find_attributes(#type)

  Get a pointer to or construct the name of the attributes for the type.

9.12.1 Detailed Description

Define JEOD memory interface macros.

- Most of the memory interface between JEOD and the simulation engine is handled by the JeodMemory↩ Interface.
- The macros defined in this file represent the functionality that cannot be solved using c++ classes.
- The macros prefixed with JEOD_DECLARE are used in model files that use the memory model to allocate memory.
- The remaining macros are used internally by the JEOD memory model and should not be used in model files.

9.12.2 Macro Definition Documentation

9.12.2.1 JEOD_ATTRIBUTES

```
#define JEOD_ATTRIBUTES(
              type ) JeodSimulationInterface::get_memory_interface().find_attributes(#type)
```

Get a pointer to or construct the name of the attributes for the type.

Note

This is a primitive macro. Do not use it in model files.

Parameters

type	Data type.

Returns

Pointer to or symbolic name of the attributes for the type.

Definition at line 109 of file memory_attributes.hh.

9.12.2.2 JEOD_DECLARE_ATTRIBUTES

```
#define JEOD_DECLARE_ATTRIBUTES(
              class_name )
```

JEOD_DECLARE_ATTRIBUTES(class_name) This macro is obsolete.

Definition at line 99 of file memory_attributes.hh.

9.13 memory_interface.cc File Reference

Implement the MemoryInterface class.

```
#include "../include/memory_interface.hh"
```

Namespaces

- jeod

  Namespace jeod.

9.13.1 Detailed Description

Implement the MemoryInterface class.

9.14 memory_interface.hh File Reference

Define the MemoryInterface class, which abstractly defines the interface between the memory manager and the simulation engine.

```
#include <cstddef>
#include <string>
#include <typeinfo>
#include "utils/sim_interface/include/jeod_class.hh"
#include "memory_attributes.hh"
```

Data Structures

- class jeod::JeodMemoryInterface

  Abstract interface between the JEOD memory manager and the simulation engine.

Namespaces

- jeod

  Namespace jeod.

9.14.1 Detailed Description

Define the MemoryInterface class, which abstractly defines the interface between the memory manager and the simulation engine.

9.15 sim_interface_messages.cc File Reference

Implement the class SimInterfaceMessages.

```
#include "../include/sim_interface_messages.hh"
```

Namespaces

- jeod

  Namespace jeod.

Macros

- #define PATH "utils/sim_interface/"
- #define CLASS SimInterfaceMessages
- #define MAKE_MESSAGE_CODE(id) char const * CLASS::id = PATH #id

9.15.1 Detailed Description

Implement the class SimInterfaceMessages.

9.16 sim_interface_messages.hh File Reference

Define the class SimInterfaceMessages, the class that specifies the message IDs used in the sim interface model.

```
#include "jeod_class.hh"
```

Data Structures

- class jeod::SimInterfaceMessages

  Specifies the message IDs used in the sim_interface model.

Namespaces

- jeod

  Namespace jeod.

9.16.1 Detailed Description

Define the class SimInterfaceMessages, the class that specifies the message IDs used in the sim interface model.

9.17 simulation_interface.cc File Reference

Implement SimulationInterface methods.

```
#include <cstddef>
#include "utils/message/include/message_handler.hh"
#include "utils/memory/include/memory_manager.hh"
#include "../include/simulation_interface.hh"
#include "../include/sim_interface_messages.hh"
```

Namespaces

- jeod

  Namespace jeod.

9.17.1 Detailed Description

Implement SimulationInterface methods.

9.18 simulation_interface.hh File Reference

Define the abstract class JeodSimulationInterface.

```
#include <string>
#include "class_declarations.hh"
#include "jeod_class.hh"
#include "checkpoint_input_manager.hh"
#include "checkpoint_output_manager.hh"
#include "jeod_integrator_interface.hh"
```

Data Structures

- class jeod::JeodSimulationInterfaceInit

  Define configuration data needed to configure the dynamically-created message handler and memory manager.
- class jeod::JeodSimulationInterface

  This abstract class defines the basis for the interface between JEOD and a simulation engine.

Namespaces

- jeod

  Namespace jeod.

9.18.1 Detailed Description

Define the abstract class JeodSimulationInterface.

9.19 trick10_memory_interface.cc File Reference

Define JeodTrickMemoryInterface methods.

```
#include <cstddef>
#include <dlfcn.h>
#include <iomanip>
#include <sstream>
#include <iosfwd>
#include "sim_services/CheckPointAgent/include/ClassicCheckPointAgent.hh"
#include "sim_services/MemoryManager/include/MemoryManager.hh"
#include "utils/message/include/message_handler.hh"
#include "../include/simulation_interface.hh"
#include "../include/sim_interface_messages.hh"
#include "../include/trick10_memory_interface.hh"
```

Namespaces

- jeod

  Namespace jeod.

Variables

- Trick::MemoryManager * trick_MM

9.19.1 Detailed Description

Define JeodTrickMemoryInterface methods.

9.20 trick10_memory_interface.hh File Reference

Define the interface for registering / deregistering memory with Trick.

```
#include <cstddef>
#include <cstring>
#include <list>
#include <map>
#include <stdint.h>
#include <string>
#include "jeod_class.hh"
#include "memory_attributes.hh"
#include "memory_interface.hh"
#include "simulation_interface.hh"
#include "trick_memory_interface.hh"
```

Data Structures

- class jeod::JeodTrick10MemoryInterface

  A TrickMemoryInterface implements the two required methods needed to register and deregister memory with the simulation engine, Trick in this case.

Namespaces

- jeod

  Namespace jeod.
- Trick

  Namespace Trick furnishes several standard functions for use in the Trick environment.

9.20.1 Detailed Description

Define the interface for registering / deregistering memory with Trick.

9.21 trick_dynbody_integ_loop.cc File Reference

Define JeodDynbodyIntegrationLoop methods.

```
#include "../include/trick_dynbody_integ_loop.hh"
#include "../include/sim_interface_messages.hh"
#include "dynamics/dyn_manager/include/dyn_manager.hh"
#include "dynamics/dyn_body/include/dyn_body.hh"
#include "environment/time/include/time_manager.hh"
#include "utils/message/include/message_handler.hh"
#include "sim_services/Executive/include/exec_proto.h"
#include <cstddef>
#include <string>
#include <vector>
```

Namespaces

- jeod

  Namespace jeod.

Variables

- Trick::Integrator * trick_curr_integ

9.21.1 Detailed Description

Define JeodDynbodyIntegrationLoop methods.

9.22 trick_dynbody_integ_loop.hh File Reference

Define the class IntegrationGroupIntegLoopScheduler, which replaces the base Trick integration loop for multi-rate JEOD-based simulations.

```
#include "jeod_trick_integrator.hh"
#include "dynamics/dyn_manager/include/dynamics_integration_group.hh"
#include "utils/integration/include/jeod_integration_group.hh"
#include "utils/sim_interface/include/jeod_class.hh"
#include "sim_services/Integrator/include/IntegLoopScheduler.hh"
```

Data Structures

- class jeod::JeodDynbodyIntegrationLoop

  A Trick::IntegLoopScheduler that provides the ability to integrate a collection of Trick::SimObject instances over time, with the sim objects capable of being moved from one integration loop to another during run time.

Namespaces

- jeod

  Namespace jeod.
- Trick

  Namespace Trick furnishes several standard functions for use in the Trick environment.
- er7_utils

  Namespace er7_utils contains the state integration models used by JEOD.

9.22.1 Detailed Description

Define the class IntegrationGroupIntegLoopScheduler, which replaces the base Trick integration loop for multi-rate JEOD-based simulations.

9.23 trick_memory_interface.cc File Reference

Define JeodTrickMemoryInterface methods.

```
#include <cstddef>
#include <dlfcn.h>
#include <iomanip>
#include <sstream>
#include "utils/message/include/message_handler.hh"
#include "../include/simulation_interface.hh"
#include "../include/sim_interface_messages.hh"
#include "../include/trick_memory_interface.hh"
```

Namespaces

- jeod

  Namespace jeod.

9.23.1 Detailed Description

Define JeodTrickMemoryInterface methods.

9.24 trick_memory_interface.hh File Reference

Define the interface for registering / deregistering memory with Trick.

```
#include <cstddef>
#include <cstring>
#include <list>
#include <map>
#include <stdint.h>
#include <string>
#include "jeod_class.hh"
#include "memory_attributes.hh"
#include "memory_interface.hh"
#include "simulation_interface.hh"
```

Data Structures

- class jeod::JeodTrickMemoryInterface

  A TrickMemoryInterface implements the two required methods needed to register and deregister memory with the simulation engine, Trick in this case.
- struct jeod::JeodTrickMemoryInterface::ContainerListEntry

  Describes a Checkpointable object.
- struct jeod::JeodTrickMemoryInterface::AllocationMapEntry

  Describes a chunk of JEOD-allocated memory.

Namespaces

- jeod

  Namespace jeod.

9.24.1 Detailed Description

Define the interface for registering / deregistering memory with Trick.

9.25 trick_memory_interface_alloc.cc File Reference

Define JeodTrickMemoryInterface methods related to allocation/deallocation.

```
#include <cstddef>
#include <cstdlib>
#include <cstdio>
#include <cstring>
#include <dlfcn.h>
#include <iomanip>
#include <sstream>
#include <typeinfo>
#include "sim_services/MemoryManager/include/attributes.h"
#include "sim_services/MemoryManager/include/ADefParseContext.hh"
#include "sim_services/MemoryManager/include/MemoryManager.hh"
#include "utils/memory/include/memory_item.hh"
#include "utils/memory/include/memory_manager.hh"
#include "utils/memory/include/memory_type.hh"
#include "utils/message/include/message_handler.hh"
#include "../include/sim_interface_messages.hh"
#include "../include/simulation_interface.hh"
#include "../include/trick_memory_interface.hh"
```

Namespaces

- jeod

  Namespace jeod.

Variables

- Trick::MemoryManager * trick_MM

9.25.1 Detailed Description

Define JeodTrickMemoryInterface methods related to allocation/deallocation.

9.26 trick_memory_interface_attrib.cc File Reference

Define JeodTrickMemoryInterface methods related to attributes.

```
#include <cstddef>
#include <cstring>
#include <dlfcn.h>
#include "sim_services/MemoryManager/include/attributes.h"
#include "utils/memory/include/memory_type.hh"
#include "utils/message/include/message_handler.hh"
#include "utils/named_item/include/named_item.hh"
#include "../include/sim_interface_messages.hh"
#include "../include/trick_memory_interface.hh"
```

Namespaces

- jeod

  Namespace jeod.

9.26.1 Detailed Description

Define JeodTrickMemoryInterface methods related to attributes.

9.27 trick_memory_interface_chkpnt.cc File Reference

Define JeodTrick10MemoryInterface methods related to checkpoint/restart.

```
#include <cstddef>
#include <cstdlib>
#include <cstdio>
#include <cstring>
#include <dlfcn.h>
#include <iomanip>
#include <sstream>
#include <typeinfo>
#include "sim_services/MemoryManager/include/attributes.h"
#include "sim_services/MemoryManager/include/ADefParseContext.hh"
#include "sim_services/MemoryManager/include/MemoryManager.hh"
#include "utils/container/include/checkpointable.hh"
#include "utils/memory/include/memory_item.hh"
#include "utils/memory/include/memory_manager.hh"
#include "utils/memory/include/memory_type.hh"
#include "utils/message/include/message_handler.hh"
#include "../include/sim_interface_messages.hh"
#include "../include/simulation_interface.hh"
#include "../include/trick10_memory_interface.hh"
```

Namespaces

- jeod

  Namespace jeod.

Variables

- Trick::MemoryManager * trick_MM

9.27.1 Detailed Description

Define JeodTrick10MemoryInterface methods related to checkpoint/restart.

9.28 trick_memory_interface_xlate.cc File Reference

Define JeodTrickMemoryInterface methods related to name translation.

```
#include <cstddef>
#include <cstdlib>
#include <string>
#include <iosfwd>
#include "sim_services/CheckPointAgent/include/ClassicCheckPointAgent.hh"
#include "sim_services/MemoryManager/include/attributes.h"
#include "sim_services/MemoryManager/include/MemoryManager.hh"
#include "sim_services/MemoryManager/include/memorymanager_c_intf.h"
#include "sim_services/CheckPointRestart/include/CheckPointRestart_c_intf.↩
hh"
#include "utils/memory/include/memory_type.hh"
#include "../include/simulation_interface.hh"
#include "../include/sim_interface_messages.hh"
#include "../include/trick10_memory_interface.hh"
```

Namespaces

- jeod

  Namespace jeod.

Variables

- Trick::MemoryManager * trick_MM

9.28.1 Detailed Description

Define JeodTrickMemoryInterface methods related to name translation.

9.29 trick_message_handler.cc File Reference

Define member functions for the class TrickMessageHandler.

```
#include <cstdarg>
#include <cstdio>
#include "sim_services/Executive/include/exec_proto.h"
#include "sim_services/Message/include/message_proto.h"
#include "utils/memory/include/jeod_alloc.hh"
#include "../include/trick_message_handler.hh"
```

Namespaces

- jeod

  Namespace jeod.

Macros

- #define MAX_MSG_SIZE 4096

9.29.1 Detailed Description

Define member functions for the class TrickMessageHandler.

9.30 trick_message_handler.hh File Reference

Define the class TrickMessageHandler, the message handler designed for use in Trick-based simulations.

```
#include <cstdarg>
#include <string>
#include "utils/container/include/primitive_set.hh"
#include "utils/message/include/suppressed_code_message_handler.hh"
#include "utils/sim_interface/include/jeod_class.hh"
#include "class_declarations.hh"
```

Data Structures

- class jeod::TrickMessageHandler

  The MessageHandler class for designed for use in Trick-based simulations.

Namespaces

- jeod

  Namespace jeod.

9.30.1 Detailed Description

Define the class TrickMessageHandler, the message handler designed for use in Trick-based simulations.

9.31 trick_sim_interface.cc File Reference

Implement TrickSimInterface methods.

```
#include "sim_services/Executive/include/exec_proto.h"
#include "sim_services/Message/include/message_proto.h"
#include "sim_services/CommandLineArguments/include/command_line_protos.h"
#include "utils/memory/include/jeod_alloc.hh"
#include "utils/message/include/message_handler.hh"
#include "../include/sim_interface_messages.hh"
#include "../include/jeod_trick_integrator.hh"
#include "../include/trick_sim_interface.hh"
#include "../include/checkpoint_input_manager.hh"
#include "../include/checkpoint_output_manager.hh"
```

Namespaces

- jeod

  Namespace jeod.

9.31.1 Detailed Description

Implement TrickSimInterface methods.

9.32 trick_sim_interface.hh File Reference

Define the class JeodTrickSimInterface.

```
#include "utils/memory/include/memory_manager.hh"
#include "simulation_interface.hh"
#include "trick_memory_interface.hh"
#include "trick10_memory_interface.hh"
#include "trick_message_handler.hh"
#include "jeod_class.hh"
#include "utils/sim_interface/include/jeod_trick_integrator.hh"
```

Data Structures

- class jeod::BasicJeodTrickSimInterface

  The BasicJeodTrickSimInterface implements the required capabilities of the generic JeodSimulationInterface in a Trick simulation environment.
- class jeod::TrickMessageHandlerMixin

  The TrickMessageHandlerMixin implements the required capabilities of the generic JeodSimulationInterface in a Trick simulation environment.
- class jeod::JeodTrickSimInterface

  A JEODTrickSimInterface implements the required capabilities of the generic JeodSimulationInterface in a Trick simulation environment.

Namespaces

- jeod

  Namespace jeod.

9.32.1 Detailed Description

Define the class JeodTrickSimInterface.

Index