SISO-STD-004-2004

Standard for
Dynamic Link Compatible HLA API
Standard for the
HLA Interface Specification
(Version 1.3)

Reaffirmed
8 December 2014

Prepared by
High Level Architecture-Evolved
Product Support Group

SISO Inc. Board of Directors
P.O. Box 781238
Orlando, FL 32878-1238, USA

Revision History

Version	Section	Date (MM/DD/YYYY)	Description
2014		12/08/2014	Reaffirmed version. Updated document format.
2004		12/03/2004	Original published version developed by the Dynamic Link Compatible HLA API Product Development Group

Participants

At the time this product was submitted to the Standards Activity Committee (SAC) for approval to reaffirm the High Level Architecture-Evolved Product Support Group had the following membership and was assigned the following SAC Technical Area Director:

Product Support Group

Randy Saunders (Chair)
Björn Möller (Vice-Chair)
Katherine L. Morse (Secretary)

Simone Youngblood, SAC Technical Area Director

Adelantado, Martin
Andrews, William
Antelius, Frederik
Beavin, William
Bergenthal, Jeffery
Borah, Jake
Ceranowicz, Andy
Chase, Tram
Cramp, Anthony
Cutts, Dannie
Dillman, Brad
Dobrindt, Uwe
Drake, David
Dubois, Aaron
Fay, John
Gagliano, Michael
Gibson, Ralph
Granowetter, Len
Guillerit, Jean-Baptiste
Gustavson, Paul
Hazen, Mark
Henninger, Amy
Hill, Frank
Igarza, Jean-Louis
Jake Borah
Jones, Stephen
Karlsson, Mikael
Kogler, James
Lewis, Jennifer
Lightner, Gary
Lindo, Wayne
Little, Reed
Lowe, Paul
Lutz, Bob
Mauget, Regis
McCall, James (Mark)
Montgomery, Mike
Murray, Bob
Nandi, Shagoto
Oates, William
Petty, Mikel
Pokorny, Tim
Powell, Ed
Prignac, Laurent
Rodriguez, Felix
Ross, Peter
Rouget, Chris
Sandberg, Stefan
Scrudder, Roy
Shanks, Graham
Snively, Keith
Strassburger, Steffen
Stutzman, Marcy
Tapp, Martin
Tolk, Andreas
Verhage, Rene
Vrieler, Stefan
Wilson, Annette

When the Standards Activity Committee approved this product for reaffirmation on 18 November 2014, it had the following membership:

Standards Activity Committee

Jeff Abbott (Chair)
Marcy Stutzman (Vice Chair / Secretary)

Grant Bailey
Curt Blais
Peggy Gravitz
Kevin Gupton
Jean-Louis Igarza
Bob Lutz
Lana McGlynn
Thom McLean
William Oates
Simone Youngblood

When the Executive Committee approved this product for reaffirmation on 8 December 2014, it had the following membership:

Executive Committee

Michael O'Connor (Chair)
James Coolahan (Vice Chair)
Jane Bachman (Secretary)

Jeff Abbot	David Graham	Roy Scrudder
John Daly	Paul Gustavson	Robert Siegfried
John Diem	Shel Ocasio	Eric Whittington

Acknowledgments (2004)

The Dynamic Link Compatible HLA API Product Development Group (PDG) created this document as a community effort. The Simulation Interoperability Standards Organization's Standards Activity Committee chartered this PDG on December 4, 2002. The *Dynamic Link Compatible HLA API Standard for the HLA Interface Specification Version 1.*3 would not have been possible without the hard work and dedicated efforts of the following individuals:

Drafting Group

Editor: Steven R. Drake
Associate Editor: Len Granowetter
Associate Editor: Björn Möller
Associate Editor: Roger Wuerfel

Bill Helfinstine
Mikael Karlsson
Jeff Meyer
Keith Snively
Annette Wilson
Doug Wood

Product Development Group Officers

Chairperson: Steve Monson / Len Granowetter
Vice-Chairperson: Angus L.M. Thom McLean / Roger Wuerfel
Recording Secretary: Allison Griffin / Katia Sullivan
Technical Area Director: Reed Little

Reviewers Who Cast Ballots

John Stevens
Sidney Chartrand
Len Granowetter
Mike Bachmann
Bill Helfinstine
Jeff Fischer
Terry Schmidt
James Hollenbach
Jeff Meyer
Philip Moul
Stefano Cerutti
Frank Zawadal
Nils LaVine
Rosemarie Keener
John Fay
Reed Little
Annette Wilson
Roger Wuerfel
Doug Wood
Marc Williams
Thom McLean
Tomas Berg
Jim Kogler
Jane Bachman
Robert Reynolds
Randy Saunders
Chris Rouget
Anthony Cramp
Susan Solick
Emmet Beeker
Mikael Karlsson
Keith Snively
Doug Scheiding
Cam Tran
Uwe Dobrindt
Mark Crooks
Graham Shanks
Joanne Atherton
Csaba Attila Boer
Ray Drake
Steve Drake

Introduction

This Standard was originally published on 3 December 2004. It was recommended to be reaffirmed by the HLA-Evolved Product Support Group and subsequently approved by the Simulation Interoperability Standards Organization (SISO) Executive Committee on 8 December 2014. This version differs from the 2004 only in the document's format as it was brought into compliance with Standards Product template approved by the SISO Standards Activity Committee.

TABLE OF CONTENTS

LIST OF FIGURES

1. OVERVIEW

The Department of Defense High Level Architecture (HLA) promised a composable approach to simulation that would promote:

1. Interoperability among federating simulations and across functional M&S communities; and
2. Reuse of simulation components across federations, functional M&S communities, and runtime infrastructures.

Reuse of simulation components across runtime infrastructures has only been achieved through the voluntary efforts of RTI (Runtime Infrastructure) vendors as they attempted to comply with the de facto standard defined in the DMSO HLA 1.3 RTI implementation. Industry must develop and manage a standard to provide RTI link compatibility and provide formal regulation of the language-specific implementation of the HLA API.

1.1. Scope

This standard defines link compatible C++ and Java Application Programmer Interfaces (API) consistent with the U.S. DoD *High Level Architecture Interface Specification Version 1.3*, and is applicable to HLA Runtime Infrastructures and federates developed in compliance with that specification.

1.2. Purpose

Compliance with the application programmer interfaces defined in this specification will permit simulation developers to interchange link compatible HLA RTIs without recompiling federate source code or re-linking federate object code with Dynamic Link-Compatible (DLC) RTI libraries. The use of DLC compliant RTIs does not ensure that federations will have similar performance, but will enable federate developers to easily utilize RTIs provided by different vendors.

1.3. Objectives

The primary objective of this standard is to provide a mechanism to permit federates to utilize RTIs developed in compliance with the DoD High Level Architecture Version 1.3 and this specification, without recompiling or relinking federate code. It is not intended to facilitate functional compatibility. This standard is intended to establish the C++ and Java API specifications.

2. REFERENCES (NORMATIVE)

#	Document Number	Title	Date
1.		U.S. Department of Defense, *High Level Architecture Interface Specification Version 1.3*, Draft	11, 20 April 1998.
2.		U.S. Department of Defense, *Interpretations of the High Level Architecture Interface Specification Version 1.3*, Release 3	

3. DEFINITIONS, ACRONYMS, AND ABBREVIATIONS

3.1. Definitions

Term	Definition
Link Compatibility	In the case of a statically linked RTI, compiler-supplied libraries must successfully resolve all compiler symbols. In the case of a dynamically linked RTI, the loading and binding operations at runtime must be successful. In both cases, the RTI libraries must be identically named, and all compatible symbols

must have the same names. Correct execution of a federate or federation is not guaranteed by link compatibility.

A pre-requisite of link-compatibility among different RTI implementations is that they are each built on compatible platforms. The term "platform" includes operating system version, compiler version,. Application Binary Interface (ABI), and choice of compiler and linker flags. Switching from one RTI implementation to another without recompiling or relinking is possible when both implementations meet this specification, and when both implementations support the same platform (or compatible platforms).

Functional Compatibility — Although the performance of one RTI may vary from another, RTIs are functionally compatible if they are otherwise interchangeable. The HLA 1.3 Interface Specification falls short of fully specifying an interface to achieve functional compatibility.

3.2. Acronyms and Abbreviations

Acronym or Abbreviation	Meaning
ABI	Application Binary Interface
API	Application Programmer Interface
COTS	Commercial-off-the-shelf
DLC	Dynamic Link-Compatible
DMSO	Defense Modeling and Simulation Organization
DoD	Department of Defense
FED	Federation Execution Data
HLA	High Level Architecture
JLC	Java Link-Compatible
M&S	Modeling and Simulation
RTI	Runtime Infrastructure
SISO	Simulation Interoperability Standards Organization
URL	Universal Resource Locator

4. C++ IMPLEMENTATION REQUIREMENTS

4.1. Technical Approach

The general approach to achieving dynamic-link compatibility for C++ is to define a standard set of header files that all compliant RTI implementations build to. A federate compiles against the standard headers and dynamically links against any RTI that complies with this standard. The federate can then switch to a different RTI implementation by utilizing any DLC RTI library.

4.2. RTI Library

The implementation of the RTI API shall be provided in a library named librti13 followed by the extension appropriate for each platform, e.g. librti13.dll or librti13.so. The librti13 library shall contain all classes and method implementations except rti13::FedTimeFactory.

4.3. FEDTIME Library

The implementations of rti13::FedTime and rti13::FedTimeFactory shall be contained in a library named libfedtime13, followed by the extension appropriate for each platform, e.g. libfedtime13.dll or libfedtime13.so. The RTI API uses an abstract base class called rti13::FedTime and static members of the rti13::FedTimeFactory.

4.4. Completeness of Implementation

An RTI shall provide implementations for all functions and classes listed in the standard RTI header files, with the exception of the rti13::FederateAmbassador, rti13::FedTimeFactory and rti13::FedTime.

4.5. Tick and RTI Callbacks

All RTI implementations shall support a synchronous mechanism for invoking Federate Ambassador callbacks. In this mode, FederateAmbassador functions shall be invoked from within the federate thread, only during a call to rti13::RTIambassador::tick(). RTI implementations may also implement other methods of callback invocation.

4.6. Format of FED File Designator

All RTIs shall support the interpretation of the FED designator as a regular file name. (The FED designator is an argument to the rti13::RTIambassador::createFederationExecution() service).

4.7. Third Party Libraries

RTI implementations shall link directly with any required third-party libraries. This ensures that federate developers need to link only to librti13 and libfedtime13 in order to use the RTI, and ensures that a federate will be able to switch from one RTI implementation to another without having to re-link against the third party libraries required by the new implementation.

4.8. Use of C++ Standard Versus Legacy Versions of OSTREAM

On platforms that support both C++ standard and legacy versions of ostream, an RTI shall provide support for both versions. There are several functions within the RTI API that require the use of the ostream class that is part of the C++ standard library. Some platforms support both the C++ standard version of ostream (which is part of the std namespace), and the legacy version of ostream, which is in the global namespace. This enables a federate to choose which version of ostream to use, knowing that the RTI supports either case.

For example, the RTI API includes an output operator for rti13::Exceptions. A compliant RTI is required to implement two versions of this function: one for global ostream, and one for std::ostream:

```
std::ostream RTI_EXPORT & operator << (std::ostream &, rti13::Exception *);
ostream RTI_EXPORT & operator << (ostream &, rti13::Exception *);
```

When a federate includes RTI header files, the federate defines the preprocessor macro RTI_USES_STD_FSTREAM if and only if it is using std::ostream, and therefore wishes to use the std::ostream versions of these RTI operators.

4.9. Spurious Compiler Warnings

Certain C++ compilers will generate many innocuous warnings when compiling the RTI API header files, and disabling these warnings can only be accomplished through pragmas. Federate developers who would like to disable these warnings (as the RTI NG v6 header files automatically did), should add a definition for the preprocessor macro RTI_DISABLE_WARNINGS to their compile settings.

4.10. Backwards Compatibility

Due to API differences between this standard and the header files provided with the final release of the DMSO RTI (NGv6), applications built against RTI NG v6 (or RTIs that claim to be Dynamic-Link-Compatible

with RTI NG v6), cannot directly swap in an RTI implementation that meets this standard. However, this standard was developed in a manner that permits the use of a backwards-compatibility library that will allow old federates to use new RTIs indirectly. Such a wrapper library would implement the old API in terms of the new. Federates built against RTI NGv6 can swap in the wrapper library without recompiling or re-linking, and every service call that the federate makes is forwarded to the new RTI by the wrapper. RTI developers may choose to distribute a compatibility wrapper library to facilitate the use of their RTIs by legacy federates.

The API changes have also broken compile-compatibility with old RTIs. However, porting source code from a federate from the RTI NG v6 header files to the new standard header files is fairly trivial. For the most part, all that is necessary is replacing the RTI:: prefix on all RTI classes with the new prefix rti13::. In addition, federates that have relied on the macros in RTItypes.hh will need to switch to the new API's alternative macros and functions. For example, instead of using MAX_EXTENT or MIN_EXTENT, use rti13::Region::getMinExtent() or rti13::Region::getMaxExtent().

For those federates who wish to re-compile against this RTI standard without having to change code, a backwards-compatibility header file (still called RTI.hh) is provided in the standard. By including RTI.hh instead of the recommended rti13.h, most federate code that compiled against RTI NG v6 will compile against RTIs that meet this standard. (Exceptions are federates that were relying on some of the macros in RTItypes.hh that had been deprecated well before RTI NG v6, or on macros that provided RTI version numbers).

5. JAVA IMPLEMENTATION REQUIREMENTS

The Java Link Compatibility (JLC) API is a vendor-independent solution to the link compatibility problem, that is, the ability to switch between different RTIs without recompiling federates.

5.1. Technical Approach

Java link compatibility requires a solution that provides an implementation-independent way to get access to implementation-specific RTI classes.

The solution in the JLC API is to have a common RtiFactoryFactory class that returns an implementation-specific class that implements the standardized interface RtiFactory that in turn can provide the actual class instances that are needed for a specific RTI, e.g. RTIambassador.

One important aspect is how to choose the right RTI in case there are several RTIs with Java bindings available. It may also be useful to be able to instantiate several RTIs in the same federate, for example in a bridge federate. The following approach is chosen:

1. A specific RTI can be requested by providing the implementation-specific class name.

2. If no class name is provided, the RTI implementation to use is specified in stored settings, e.g. a Java properties file. If no stored settings are found, or if no default RTI is specified in the stored settings then the RtiFactoryFactory may choose an RTI freely.

5.1.1. Sample Use in a Federate

General case:

A flexible Federate that may use any JLC-compliant RTI. It uses the default RTI as specified in the stored settings.

```
import hla.rti.jlc.*;
RtiFactory rtiFactory = RtiFactoryFactory.getRtiFactory();
RTIambassador _rtiAmbassador = rtiFactory.createRtiAmbassador();
```

Special case:

A federate that wants to override this and use a specific RTI:

```
RtiFactory rtiFactory =
RtiFactoryFactory.getRtiFactory("com.rtibusters.rti.BustersRtiFactory");
RTIambassador _rtiAmbassador = rtiFactory.createRtiAmbassador();
```

5.2. Package Naming

The JLC classes shall reside in a package named hla.rti.jlc. The RTI shall also provide the HLA 1.3 Java API classes in the package hla.rti. Each RTI vendor shall provide its RTI implementation classes in a implementation-specific package.

5.3. Completeness of Implementation

The RTI shall provide implementations for the classes RtiFactoryFactory, NullFederateAmbassador, and EncodingHelpers in the hla.rti.jlc package. The RTI shall also provide a class that implements the RtiFactory interface defined in the hla.rti.jlc package. This class shall reside in an implementation-specific package different from hla.rti and hla.rti.jlc.

5.4. Tick and RTI Callbacks

All RTI implementations shall support a synchronous mechanism for invoking Federate Ambassador callbacks. In this mode, FederateAmbassador functions shall be invoked from within the federate thread, only during a call to RTIambassador.tick(). RTI implementations may also implement other methods of callback invocation.

5.5. Format of FED File Designator

All RTIs shall support the interpretation of the FED designator as a URL. (The FED designator is an argument to the createFederationExecution() service).

5.6. Factory Mechanism

The JLC API adds a Java package. The name of this new package is hla.rti.jlc.

The new hla.rti.jlc package contains:

1. class RtiFactoryFactory
2. interface RtiFactory

Each RTI developer shall provide a class that implements the RtiFactory interface. That class shall reside in the RTI's own package. For example, the company RtiBusters may provide an implementation of the RtiFactory interface in a class named BustersRtiFactory in the package "com.rtibusters.rti".

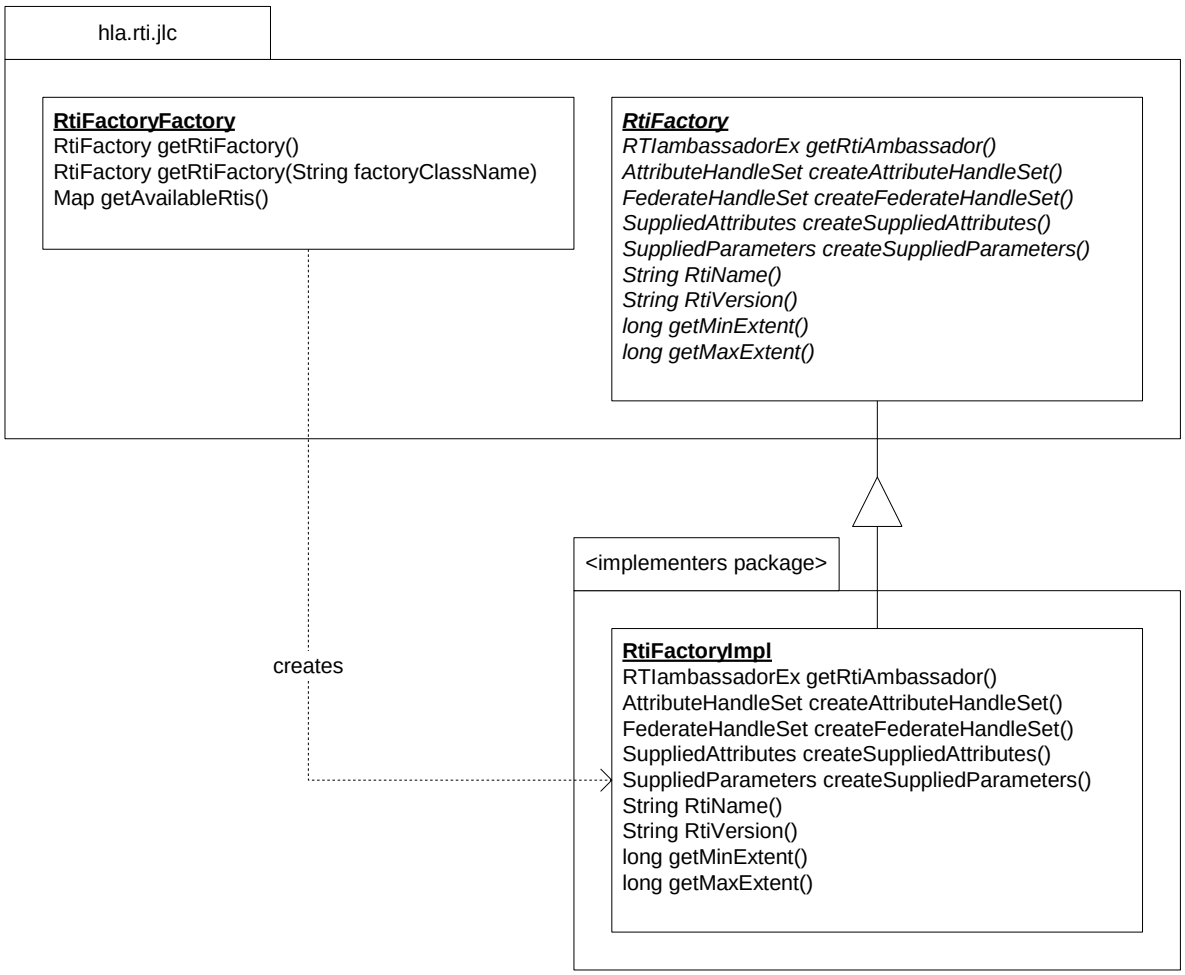

Figure 1 Java Class Diagram

5.6.1. RtiFactoryFactory

The JLC API shall contain a class named RtiFactoryFactory that provides methods for creating a new RtiFactory instance, either by specifying a specific RTI or by using stored user-editable settings. It shall also have a method for getting the available RTIs from the stored settings. A default RTI shall be specified using stored settings. This places the burden of creating and maintaining these settings on the user.

All RTI implementations shall provide an implementation of the RtiFactoryFactory class.

The RtiFactoryFactory class shall contain the following methods:

RtiFactory getRtiFactory()

Returns an RtiFactory instance for the default RTI as specified in the stored settings. If no stored settings are available, or if no default RTI is specified in the stored settings then the RtiFactoryFactory may choose an RTI freely.

RtiFactory getRtiFactory(String factoryClassName)

Returns an RtiFactory instance of the specified factory class. If no RtiFactory is associated with the specified name, the RTIinternalError exception shall be thrown.

Map getAvailableRtis()

Returns a Map with RTI names as keys and factory class names as mapped values. The names and values are read from the stored settings.

5.6.2. RtiFactory

The JLC API shall contain an interface named RtiFactory. The interface shall contain methods for creating the classes necessary to run a federate. It shall also have methods for getting the name and version of the specific RTI.

Each vendor shall provide a class that implements the RtiFactory interface. The name of this class shall be clearly documented. The full name of this class (including the package) may be used as parameter to the method RtiFactoryFactory.getRtiFactory( ).

All RTI developers shall provide the RtiFactory interface.

The RtiFactory interface shall define the following methods:

RTIambassadorEx getRtiAmbassador()
Returns an object instance that implements the RTIambassadorEx interface.

AttributeHandleSet createAttributeHandleSet()
Returns an object instance that implements the AttributeHandleSet interface.

FederateHandleSet createFederateHandleSet()
Returns an object instance that implements the FederateHandleSet interface.

SuppliedAttributes createSuppliedAttributes()
Returns an object instance that implements the SuppliedAttributes interface.

SuppliedParameters createSuppliedParamters()
Returns an object instance that implements the SuppliedParameters interface.

String RtiName()
Returns the name of the RTI.

String RtiVersion()
Returns the version of the RTI.

long getMinExtent()
Returns the minimum extent allowed in a range.

long getMaxExtent()
Returns the maximum extent allowed in a range.

5.6.3 Additional functionality

The JLC API shall add a new extended version of the RTIambassador interface to the hla.rti.jlc package. This new interface shall be called RTIambassadorEx. It shall add a new method tick(min, max), matching the same method in the C++ API.

```
package hla.rti.jlc;

import hla.rti.*;

public interface RTIambassadorEx extends RTIambassador
{
  public boolean tick(final double min, final double max)
        throws hla.rti.RTIinternalError, hla.rti.ConcurrentAccessAttempted;
}
```

To use this new functionality, the federate declares its RTIambassador reference to use the new interface RTIambassadorEx. Example:

```
import hla.rti.jlc.*;

RtiFactory rtiFactory = RtiFactoryFactory.getRtiFactory();
RTIambassadorEx _rtiAmbassador = rtiFactory.createRtiAmbassador();
```

Existing federates can continue to use the unchanged RTIambassador interface.

The RTI implemention shall return from the RtiFactory.getRtiAmbassador method an object instance that implements the RTIambassadorEx interface.

The JLC API shall add the following utility classes to the hla.rti.jlc package.

1. NullFederateAmbassador
   Provides empty implementations of all methods in FederateAmbassador.
2. EncodingHelpers
   Provides helper functions for encoding various data types.

The utility classes NullFederateAmbassador and EncodingHelpers shall be provided by the RTI implementer.

5.7. Backwards Compatibility

The JLC shall only include additions to the API. It shall not contain any changes of existing functionality or names. A federate that previously worked with a specific RTI will need no modifications. New federates can use the new functionality and still interoperate in a federation execution with old, unmodified federates (assuming that they use the same RTI).

6. RTI COMPLIANCE TESTING

Since this specification is not intended to ensure functional compatibility, compliance testing can be easily accomplished with a simple test federate that contains an empty main() function. The test federate should contain other functions that create all data types and explicitly call or declare each function specified in Appendix A or Appendix B of this document. Dynamic link compliance is demonstrated when the test federate loads successfully and the dynamic linker resolves all symbols.

For functional compatibility, refer to the HLA 1.3 Interface Specification

APPENDIX A C++ APPLICATION PROGRAMMER INTERFACE (NORMATIVE)

A.1 RTI.hh

```
//File RTI.hh

// This file exists solely to provide compile-time
// backwards-compatibility for federates built before
// the existence of the Dynamic-Link-Compatible RTI API.
// New federates should be including RTI13.h rather than
// this file, RTI.hh.

#ifndef RTI_BACKCOMPAT_hh
#define RTI_BACKCOMPAT_hh

#include "RTI13.h"

// The following alias allows old code to continue to reference RTI
// classes by the old names (qualified by RTI:: rather than rti13::).
namespace RTI = rti13;

// Map old macros to new static member functions of Region class
#define MAX_EXTENT (RTI::Region::getMaxExtent())
#define MIN_EXTENT (RTI::Region::getMinExtent())

#endif // RTI_BACKCOMPAT_hh
```

A.2 RTI13.h

```
//File RTI13.h

#ifndef RTI13_h
#define RTI13_h

// Identification of the API version number.
#define HLA_SPECIFICATION_NAME "1.3"
#define HLA_API_MAJOR_VERSION 7
#define HLA_API_MINOR_VERSION 0

#if defined(_WIN32)
#if defined(_MSC_VER) && defined(RTI_DISABLE_WARNINGS)
// disable warning about exceptions not being part of a method's signature
#pragma warning(disable: 4290)
// disable warnings about a "dllexport" class using a regular class
#pragma warning(disable: 4251)
#endif
//
// On Windows, BUILDING_RTI should be defined only when compiling
// the RTI DLL (i.e. by RTI developers).  BUILDING_FEDTIME should
// be defined only when building a libfedtime DLL.  STATIC_RTI
// should be defined when building a static (non-DLL) RTI library,
// or when building a federate that wants to statically link to
// an RTI library.  STATIC_FEDTIME should be defined when building
// a static (non-DLL) fedtime library, or when building a federate
// that wants to statically link to a fedtime library.
//
#if defined(STATIC_RTI)
#define RTI_EXPORT
#else
#if defined(BUILDING_RTI)
// define the proper qualifiers to import/export symbols from/to DLL
#define RTI_EXPORT __declspec(dllexport)
#else // !BUILDING_RTI
#define RTI_EXPORT __declspec(dllimport)
#endif // BUILDING_RTI
#endif // STATIC_RTI

#if defined(STATIC_FEDTIME)
#define RTI_EXPORT_FEDTIME
#else
#if defined(BUILDING_FEDTIME)
// define the proper qualifiers to import/export symbols from/to DLL
#define RTI_EXPORT_FEDTIME __declspec(dllexport)
#else // !BUILDING_FEDTIME
#define RTI_EXPORT_FEDTIME __declspec(dllimport)
#endif // BUILDING_FEDTIME
#endif // STATIC_FEDTIME

#else // !WIN32
// no special qualfifers are necessary on non-WIN32 platforms
#define RTI_EXPORT
#define RTI_EXPORT_FEDTIME
```

```
#endif

//
// On platforms that support both the standard C++ version of
// ostream (std::ostream), and the legacy ostream in the global
// namespace, an RTI library will contain two versions of each
// RTI function that relies on the ostream class.  This allows
// federates to work with either new or old ostreams.  When a
// federate includes RTI header files, it should define
// RTI_USES_STD_FSTREAM if and only if it is using std::ostream,
// and therefore wishes to use the std::ostream versions of these
// RTI functions.
//
#ifdef RTI_USES_STD_FSTREAM
#include <fstream>
#define RTI_STD std
#else
#include <fstream.h>
#define RTI_STD /* nothing */
#endif

#include "baseTypes13.h"
#include "RTItypes13.h"

namespace rti13
{
  struct RTIambPrivateRefs;
  struct RTIambPrivateData;

// Vendor-specific name and version of the RTI implementation
  const char *RTIname();
  const char *RTIversion(); // identical to MOM attributes of same name

  class RTI_EXPORT RTIambassador {
  public:
#include "RTIambServices13.h"
    RTIambPrivateData* privateData;
  private:
    RTIambPrivateRefs* privateRefs;
  };

  class RTI_EXPORT FederateAmbassador {
  public:
#include "federateAmbServices13.h"
  };
} // End of namespace rti13

RTI_STD::ostream RTI_EXPORT &
operator << (RTI_STD::ostream &, rti13::Exception *);

RTI_STD::ostream RTI_EXPORT &
operator << (RTI_STD::ostream &, rti13::Exception const &);

RTI_STD::ostream RTI_EXPORT &
operator << (RTI_STD::ostream &, const rti13::FedTime &);
```

```
#endif // RTI13_h
```

A.3 baseTypes13.h

```
//File baseTypes13.h
//Included in RTI13.h

#ifndef BASETYPES13_H_INCLUDED
#define BASETYPES13_H_INCLUDED

#ifndef NULL
#define NULL (0)
#endif

namespace rti13
{

typedef unsigned short UShort;
typedef short          Short;
#if defined(RTI_64_BIT_LONG)
typedef unsigned int   ULong;
typedef int            Long;
#else
typedef unsigned long  ULong;
typedef long           Long;
#endif
typedef double         Double;
typedef float          Float;

enum Boolean {
    RTI_FALSE = 0,
    RTI_TRUE};

class RTI_EXPORT Exception {
public:
  ULong _serial;
  char *_reason;
  const char *_name;
  Exception (const char *reason);
  Exception (ULong serial, const char *reason=NULL);
  Exception (const Exception &toCopy);
  virtual ~Exception ();
  Exception & operator = (const Exception &);
  virtual Exception * cloneSelf() const throw() = 0;
  virtual void        throwSelf() const         = 0;
};

#define RTI_EXCEPT(A)   \
class A : public Exception {       \
public: \
  static RTI_EXPORT const char *_ex;  \
  A (const char *reason) : Exception (reason)  { _name = _ex; }   \
  A (ULong serial, const char *reason=NULL)        \
    : Exception (serial, reason)  { _name = _ex; }        \
  A (A const & toCopy) : Exception(toCopy) { _name = _ex; } \
  Exception * cloneSelf() const throw() { return (new A(_reason)); } \
  void        throwSelf() const         { throw *this; } \
```

```
};

} // End of namespace rti13

#endif // BASETYPES13_H_INCLUDED
```

A.4 RTItypes13.h

```
//File RTItypes13.h
//Included in RTI13.h

#ifndef RTITYPES13_H_INCLUDED
#define RTITYPES13_H_INCLUDED

namespace rti13
{
RTI_EXCEPT(ArrayIndexOutOfBounds)
RTI_EXCEPT(AsynchronousDeliveryAlreadyDisabled)
RTI_EXCEPT(AsynchronousDeliveryAlreadyEnabled)
RTI_EXCEPT(AttributeAcquisitionWasNotRequested)
RTI_EXCEPT(AttributeAcquisitionWasNotCanceled)
RTI_EXCEPT(AttributeAlreadyBeingAcquired)
RTI_EXCEPT(AttributeAlreadyBeingDivested)
RTI_EXCEPT(AttributeAlreadyOwned)
RTI_EXCEPT(AttributeDivestitureWasNotRequested)
RTI_EXCEPT(AttributeNotDefined)
RTI_EXCEPT(AttributeNotKnown)
RTI_EXCEPT(AttributeNotOwned)
RTI_EXCEPT(AttributeNotPublished)
RTI_EXCEPT(ConcurrentAccessAttempted)
RTI_EXCEPT(CouldNotDiscover)
RTI_EXCEPT(CouldNotOpenFED)
RTI_EXCEPT(CouldNotRestore)
RTI_EXCEPT(DeletePrivilegeNotHeld)
RTI_EXCEPT(DimensionNotDefined)
RTI_EXCEPT(EnableTimeConstrainedPending)
RTI_EXCEPT(EnableTimeConstrainedWasNotPending)
RTI_EXCEPT(EnableTimeRegulationPending)
RTI_EXCEPT(EnableTimeRegulationWasNotPending)
RTI_EXCEPT(ErrorReadingFED)
RTI_EXCEPT(EventNotKnown)
RTI_EXCEPT(FederateAlreadyExecutionMember)
RTI_EXCEPT(FederateInternalError)
RTI_EXCEPT(FederateLoggingServiceCalls)
RTI_EXCEPT(FederateNotExecutionMember)
RTI_EXCEPT(FederateOwnsAttributes)
RTI_EXCEPT(FederateWasNotAskedToReleaseAttribute)
RTI_EXCEPT(FederatesCurrentlyJoined)
RTI_EXCEPT(FederationExecutionAlreadyExists)
RTI_EXCEPT(FederationExecutionDoesNotExist)
RTI_EXCEPT(FederationTimeAlreadyPassed)
RTI_EXCEPT(HandleValuePairMaximumExceeded)
RTI_EXCEPT(InteractionClassNotDefined)
RTI_EXCEPT(InteractionClassNotKnown)
RTI_EXCEPT(InteractionClassNotPublished)
RTI_EXCEPT(InteractionClassNotSubscribed)
RTI_EXCEPT(InteractionParameterNotDefined)
RTI_EXCEPT(InteractionParameterNotKnown)
RTI_EXCEPT(InvalidExtents)
RTI_EXCEPT(InvalidFederationTime)
RTI_EXCEPT(InvalidHandleValuePairSetContext)
```

```
RTI_EXCEPT(InvalidLookahead)
RTI_EXCEPT(InvalidOrderingHandle)
RTI_EXCEPT(InvalidRegionContext)
RTI_EXCEPT(InvalidResignAction)
RTI_EXCEPT(InvalidRetractionHandle)
RTI_EXCEPT(InvalidTransportationHandle)
RTI_EXCEPT(MemoryExhausted)
RTI_EXCEPT(NameNotFound)
RTI_EXCEPT(ObjectClassNotDefined)
RTI_EXCEPT(ObjectClassNotKnown)
RTI_EXCEPT(ObjectClassNotPublished)
RTI_EXCEPT(ObjectClassNotSubscribed)
RTI_EXCEPT(ObjectNotKnown)
RTI_EXCEPT(ObjectAlreadyRegistered)
RTI_EXCEPT(OwnershipAcquisitionPending)
RTI_EXCEPT(RegionInUse)
RTI_EXCEPT(RegionNotKnown)
RTI_EXCEPT(RestoreInProgress)
RTI_EXCEPT(RestoreNotRequested)
RTI_EXCEPT(RTIinternalError)
RTI_EXCEPT(SpaceNotDefined)
RTI_EXCEPT(SaveInProgress)
RTI_EXCEPT(SaveNotInitiated)
RTI_EXCEPT(SpecifiedSaveLabelDoesNotExist)
RTI_EXCEPT(SynchronizationPointLabelWasNotAnnounced)
RTI_EXCEPT(TimeAdvanceAlreadyInProgress)
RTI_EXCEPT(TimeAdvanceWasNotInProgress)
RTI_EXCEPT(TimeConstrainedAlreadyEnabled)
RTI_EXCEPT(TimeConstrainedWasNotEnabled)
RTI_EXCEPT(TimeRegulationAlreadyEnabled)
RTI_EXCEPT(TimeRegulationWasNotEnabled)
RTI_EXCEPT(UnableToPerformSave)
RTI_EXCEPT(ValueCountExceeded)
RTI_EXCEPT(ValueLengthExceeded)

enum ResignAction {
  RELEASE_ATTRIBUTES = 1,
  DELETE_OBJECTS,
  DELETE_OBJECTS_AND_RELEASE_ATTRIBUTES,
  NO_ACTION
};

class Region;

class FederateAmbassador;

typedef FederateAmbassador *FederateAmbassadorPtr;

typedef Long SpaceHandle;

typedef ULong ObjectClassHandle;

typedef ULong InteractionClassHandle;

typedef ULong ExtentIndex;
```

```
typedef ULong Handle;

typedef Handle AttributeHandle;

typedef Handle ParameterHandle;

typedef Handle ObjectHandle;

typedef Handle DimensionHandle;

typedef ULong FederateHandle;

typedef Handle TransportationHandle;

typedef TransportationHandle TransportType;

typedef Handle OrderingHandle;

typedef OrderingHandle OrderType;

typedef ULong FederateID;

typedef ULong UniqueID;

typedef Double TickTime;

typedef ULong RegionToken;

class RTI_EXPORT AttributeHandleValuePairSet {
// Instances of class HandleValuePairSet are the containers used to pass
// object attribute values and interaction parameter values between the
// Fedrate and the RTI.  These containers hold sets of attribute/parameter
// values indexed by their attribute/parameter handle.  Instances of this
// class are provided to the RTI in the Update Attribute Values and Send
// Interaction service invocations.  Instances of this class are provided
// to the Federate in the Reflect Attribute Values and Receive Interaction
// service invocations.  When instances of HandleValuePairSet are provided
// to the Federate by the RTI, the memory used to store attribute/parameter
// values is valid for use by the federate only within the scope of the
// Reflect Attribute Values or Receive Interaction service invocation.
// Symmetrically, for instances of HandleValuePairSet provided by the
// Federate to the RTI, the memory used to store attribute/parameter values
// is valid for use by the RTI only within the scope of the Update
// Attribute Values or Send Interaction service invocation.
public:
  virtual ~AttributeHandleValuePairSet() { ; }

  virtual ULong size() const = 0;

  virtual Handle getHandle(
    ULong i) const
    throw (
      ArrayIndexOutOfBounds) = 0;
```

```
virtual ULong getValueLength(
  ULong i) const
  throw (
    ArrayIndexOutOfBounds) = 0;

virtual void getValue(
  ULong i,
  char*      buff,
  ULong&     valueLength) const
  throw (
    ArrayIndexOutOfBounds) = 0;

virtual char *getValuePointer(
  ULong i,
  ULong&     valueLength) const
  throw (
    ArrayIndexOutOfBounds) = 0;

virtual TransportType getTransportType( ULong i) const
  throw (
    ArrayIndexOutOfBounds,
    InvalidHandleValuePairSetContext) = 0;

virtual OrderType getOrderType( ULong i) const
  throw (
    ArrayIndexOutOfBounds,
    InvalidHandleValuePairSetContext) = 0;

virtual Region *getRegion(
  ULong i) const
  throw (
    ArrayIndexOutOfBounds,
    InvalidHandleValuePairSetContext) = 0;

virtual void add(
  Handle      h,
  const char* buff,
  ULong       valueLength)
  throw (
    ValueLengthExceeded,
    ValueCountExceeded) = 0;

virtual void remove(        // not guaranteed safe while iterating
  Handle      h)
  throw (
    ArrayIndexOutOfBounds) = 0;

virtual void moveFrom(
  const AttributeHandleValuePairSet& ahvps,
  ULong&               i)
  throw (
    ValueCountExceeded,
    ArrayIndexOutOfBounds) = 0;

virtual void empty() = 0; // Empty the Set without deallocating space.
```

```
  virtual ULong start() const = 0;
  virtual ULong valid(ULong i) const = 0;
  virtual ULong next(ULong i) const = 0;
};


class RTI_EXPORT AttributeSetFactory {
public:
  static AttributeHandleValuePairSet* create(
    ULong count)
    throw (
      MemoryExhausted,
      ValueCountExceeded,
      HandleValuePairMaximumExceeded);
};

class RTI_EXPORT AttributeHandleSet {
public:
  virtual ~AttributeHandleSet() { ; }

  virtual ULong size() const = 0;

  virtual AttributeHandle getHandle(ULong i) const
    throw (
      ArrayIndexOutOfBounds) = 0;

  virtual void add(AttributeHandle h)
    throw (
      ArrayIndexOutOfBounds,
      AttributeNotDefined) = 0;

  virtual void remove(AttributeHandle h)
    throw (             // not guaranteed safe while iterating
      AttributeNotDefined) = 0;

  virtual void empty() = 0; // Empty the Set

  virtual Boolean isEmpty() const = 0;  //is set empty?
  virtual Boolean isMember(AttributeHandle h) const = 0;
};


class RTI_EXPORT AttributeHandleSetFactory {
public:
  static AttributeHandleSet* create(
    ULong count)
    throw(
      MemoryExhausted,
      ValueCountExceeded);
};

class RTI_EXPORT FederateHandleSet {
public:
```

```
  virtual ~FederateHandleSet() {}

  virtual ULong size() const = 0;

  virtual FederateHandle getHandle(ULong i) const
    throw (
      ArrayIndexOutOfBounds) = 0;

  virtual void add(FederateHandle h)
    throw (
      ValueCountExceeded) = 0;

  virtual void remove(FederateHandle h)
    throw (
      ArrayIndexOutOfBounds) = 0;

  virtual void empty() = 0; // Empty the set without deallocating space.

  virtual Boolean isMember(FederateHandle h) const = 0;
};

class RTI_EXPORT FederateHandleSetFactory {
public:
  static FederateHandleSet* create(ULong count)
    throw (
      MemoryExhausted,
      ValueCountExceeded);
};


class RTI_EXPORT ParameterHandleValuePairSet {
// Instances of class HandleValuePairSet are the containers used to pass
// object attribute values and interaction parameter values between the
// Fedrate and the RTI.  These containers hold sets of attribute/parameter
// values indexed by their attribute/parameter handle.  Instances of this
// class are provided to the RTI in the Update Attribute Values and Send
// Interaction service invocations.  Instances of this class are provided
// to the Federate in the Reflect Attribute Values and Receive Interaction
// service invocations.  When instances of HandleValuePairSet are provided
// to the Federate by the RTI, the memory used to store attribute/parameter
// values is valid for use by the federate only within the scope of the
// Reflect Attribute Values or Receive Interaction service invocation.
// Symmetrically, for instances of HandleValuePairSet provided by the
// Federate to the RTI, the memory used to store attribute/parameter values
// is valid for use by the RTI only within the scope of the Update
// Attribute Values or Send Interaction service invocation.
public:
  virtual ~ParameterHandleValuePairSet() { ; }

  virtual ULong size() const = 0;

  virtual Handle getHandle(
    ULong i) const
    throw (
      ArrayIndexOutOfBounds) = 0;
```

```
  virtual ULong getValueLength(
    ULong i) const
    throw (
      ArrayIndexOutOfBounds) = 0;

  virtual void getValue(
    ULong i,
    char*      buff,
    ULong&     valueLength) const
    throw (
      ArrayIndexOutOfBounds) = 0;

  virtual char *getValuePointer(
    ULong i,
    ULong&     valueLength) const
    throw (
      ArrayIndexOutOfBounds) = 0;

  virtual TransportType getTransportType(void) const
    throw ( InvalidHandleValuePairSetContext) = 0;

  virtual OrderType getOrderType(void) const
    throw ( InvalidHandleValuePairSetContext) = 0;

  virtual Region *getRegion(void) const
    throw ( InvalidHandleValuePairSetContext) = 0;

  virtual void add(
    Handle      h,
    const char* buff,
    ULong       valueLength)
    throw (
      ValueLengthExceeded,
      ValueCountExceeded) = 0;

  virtual void remove(        // not guaranteed safe while iterating
    Handle      h)
    throw (
      ArrayIndexOutOfBounds) = 0;

  virtual void moveFrom(
    const ParameterHandleValuePairSet& phvps,
    ULong&               i)
    throw (
      ValueCountExceeded,
      ArrayIndexOutOfBounds) = 0;

  virtual void empty() = 0; // Empty the Set without deallocating space.

  virtual ULong start() const = 0;
  virtual ULong valid(ULong i) const = 0;
  virtual ULong next(ULong i) const = 0;
};
```

```
class RTI_EXPORT ParameterSetFactory {
public:
  static ParameterHandleValuePairSet* create(ULong count)
    throw (
      MemoryExhausted,
      ValueCountExceeded,
      HandleValuePairMaximumExceeded);
};

class RTI_EXPORT Region {
public:

  virtual ~Region() { }

  virtual ULong getRangeLowerBound(
    ExtentIndex     theExtent,
    DimensionHandle theDimension) const
    throw (
      ArrayIndexOutOfBounds) = 0;

  virtual ULong getRangeUpperBound(
    ExtentIndex     theExtent,
    DimensionHandle theDimension) const
    throw (
      ArrayIndexOutOfBounds) = 0;

  virtual void setRangeLowerBound(
    ExtentIndex     theExtent,
    DimensionHandle theDimension,
    ULong           theLowerBound)
    throw (
      ArrayIndexOutOfBounds) = 0;

  virtual void setRangeUpperBound(
    ExtentIndex     theExtent,
    DimensionHandle theDimension,
    ULong           theUpperBound)
    throw (
      ArrayIndexOutOfBounds) = 0;

  virtual SpaceHandle getSpaceHandle() const
    throw (
      ) = 0;

  virtual ULong getNumberOfExtents() const
    throw (
      ) = 0;

  virtual ULong getRangeLowerBoundNotificationLimit(
    ExtentIndex     theExtent,
    DimensionHandle theDimension) const
    throw (
      ArrayIndexOutOfBounds) = 0;

  virtual ULong getRangeUpperBoundNotificationLimit(
```

```
    ExtentIndex     theExtent,
    DimensionHandle theDimension) const
    throw (
      ArrayIndexOutOfBounds) = 0;

  static ULong getMaxExtent()
    throw ();

  static ULong getMinExtent()
    throw ();

};

class RTI_EXPORT FedTime {
public:
  virtual ~FedTime();

  virtual void setZero() = 0;

  virtual Boolean isZero() = 0;

  virtual void setEpsilon() = 0;

  virtual void setPositiveInfinity() = 0;

  virtual Boolean isPositiveInfinity() = 0;

  virtual FedTime& operator+= (const FedTime&)
    throw (
      InvalidFederationTime) = 0;

  virtual FedTime& operator-= (const FedTime&)
    throw (
      InvalidFederationTime) = 0;

  virtual Boolean operator<= (const FedTime&) const
    throw (
      InvalidFederationTime) = 0;

  virtual Boolean operator< (const FedTime&) const
    throw (
      InvalidFederationTime) = 0;

   virtual Boolean operator>= (const FedTime&) const
    throw (
      InvalidFederationTime) = 0;

  virtual Boolean operator> (const FedTime&) const
    throw (
      InvalidFederationTime) = 0;

  virtual Boolean operator== (const FedTime&) const
    throw (
      InvalidFederationTime) = 0;
```

```
  virtual FedTime& operator= (const FedTime&)
    throw (
      InvalidFederationTime) = 0;

  //return bytes needed to encode
  virtual int encodedLength() const = 0;

  //encode into suppled buffer
  virtual void encode(char *buff) const = 0;

  virtual int getPrintableLength() const = 0;

  virtual void getPrintableString(char*) const = 0;

};

class RTI_EXPORT_FEDTIME FedTimeFactory {
public:
  static FedTime* makeZero()
    throw (
      MemoryExhausted);

  static FedTime* decode(const char *buf)
    throw (
      MemoryExhausted);
};

struct EventRetractionHandle_s {
  UniqueID              theSerialNumber;
  FederateHandle        sendingFederate;
};
typedef struct EventRetractionHandle_s EventRetractionHandle;

} // End of namepsace rti13

#endif      /*RTITYPES13_H_INCLUDED*/
```

A.5 RTIambServices13.h

```
//File RTIambServices13.h
//Included in RTI13.h


//                RTI Parameter Passing Memory Conventions
//
// C1  In parameter by value.
// C2  Out parameter by pointer value.
// C3  Function return by value.
// C4  In parameter by const pointer value.  Caller provides memory.
//     Caller may free memory or overwrite it upon completion of
//     the call.  Callee must copy during the call anything it
//     wishes to save beyond completion of the call.  Parameter
//     type must define const accessor methods.
// C5  Out parameter by pointer value.  Caller provides reference to object.
//     Callee constructs an instance on the heap (new) and returns.
//     The caller destroys the instance (delete) at its leisure.
// C6  Function return by pointer value.  Callee constructs an instance on
//     the heap (new) and returns a reference.  The caller destroys the
//     instance (delete) at its leisure.
//

typedef FederateAmbassador *FederateAmbassadorPtr;

////////////////////////////////////
// Federation Management Services //
////////////////////////////////////

// 4.2
virtual
void createFederationExecution (
  const char *executionName, // supplied C4
  const char *FED)           // supplied C4
throw (
  FederationExecutionAlreadyExists,
  CouldNotOpenFED,
  ErrorReadingFED,
  ConcurrentAccessAttempted,
  RTIinternalError);

// 4.3
virtual
void destroyFederationExecution (
  const char *executionName) // supplied C4
throw (
  FederatesCurrentlyJoined,
  FederationExecutionDoesNotExist,
  ConcurrentAccessAttempted,
  RTIinternalError);

// 4.4
virtual
FederateHandle                                          // returned C3
```

```
joinFederationExecution (
  const char                  *yourName,                    // supplied C4
  const char                  *executionName,               // supplied C4
        FederateAmbassadorPtr   federateAmbassadorReference) // supplied C1
throw (
  FederateAlreadyExecutionMember,
  FederationExecutionDoesNotExist,
  CouldNotOpenFED,
  ErrorReadingFED,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 4.5
virtual
void resignFederationExecution (
  ResignAction theAction) // supplied C1
throw (
  FederateOwnsAttributes,
  FederateNotExecutionMember,
  InvalidResignAction,
  ConcurrentAccessAttempted,
  RTIinternalError);

// 4.6
virtual
void registerFederationSynchronizationPoint (
  const char *label,  // supplied C4
  const char *theTag) // supplied C4
throw (
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

virtual
void registerFederationSynchronizationPoint (
  const char                *label,    // supplied C4
  const char                *theTag,   // supplied C4
  const FederateHandleSet&   syncSet)  // supplied C4
throw (
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 4.9
virtual
void synchronizationPointAchieved (
  const char *label) // supplied C4
throw (
  SynchronizationPointLabelWasNotAnnounced,
```

```
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 4.11
virtual
void requestFederationSave (
  const char     *label,   // supplied C4
  const FedTime&  theTime) // supplied C4
throw (
  FederationTimeAlreadyPassed,
  InvalidFederationTime,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

virtual
void requestFederationSave (
  const char *label)     // supplied C4
  throw (
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 4.13
virtual
void federateSaveBegun ()
throw (
  SaveNotInitiated,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  RestoreInProgress,
  RTIinternalError);

// 4.14
virtual
void federateSaveComplete ()
throw (
  SaveNotInitiated,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  RestoreInProgress,
  RTIinternalError);

virtual
void federateSaveNotComplete ()
throw (
  SaveNotInitiated,
  FederateNotExecutionMember,
```

```
  ConcurrentAccessAttempted,
  RestoreInProgress,
  RTIinternalError);

// 4.16
virtual
void requestFederationRestore (
  const char *label) // supplied C4
throw (
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 4.20
virtual
void federateRestoreComplete ()
  throw (
    RestoreNotRequested,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RTIinternalError);

virtual
void federateRestoreNotComplete ()
throw (
  RestoreNotRequested,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RTIinternalError);

////////////////////////////////////
// Declaration Management Services //
////////////////////////////////////

// 5.2
virtual
void publishObjectClass (
        ObjectClassHandle   theClass,      // supplied C1
  const AttributeHandleSet& attributeList) // supplied C4
throw (
  ObjectClassNotDefined,
  AttributeNotDefined,
  OwnershipAcquisitionPending,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 5.3
virtual
```

```
void unpublishObjectClass (
  ObjectClassHandle theClass) // supplied C1
throw (
  ObjectClassNotDefined,
  ObjectClassNotPublished,
  OwnershipAcquisitionPending,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 5.4
virtual
void publishInteractionClass (
  InteractionClassHandle theInteraction) // supplied C1
throw (
  InteractionClassNotDefined,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 5.5
virtual
void unpublishInteractionClass (
  InteractionClassHandle theInteraction) // supplied C1
throw (
  InteractionClassNotDefined,
  InteractionClassNotPublished,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 5.6
virtual
void subscribeObjectClassAttributes (
        ObjectClassHandle   theClass,      // supplied C1
  const AttributeHandleSet& attributeList, // supplied C4
        Boolean        active = RTI_TRUE)
throw (
  ObjectClassNotDefined,
  AttributeNotDefined,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 5.7
virtual
void unsubscribeObjectClass (
```

```
  ObjectClassHandle theClass) // supplied C1
throw (
  ObjectClassNotDefined,
  ObjectClassNotSubscribed,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 5.8
virtual
void subscribeInteractionClass (
  InteractionClassHandle theClass, // supplied C1
  Boolean           active = RTI_TRUE)
throw (
  InteractionClassNotDefined,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  FederateLoggingServiceCalls,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 5.9
virtual
void unsubscribeInteractionClass (
  InteractionClassHandle theClass) // supplied C1
throw (
  InteractionClassNotDefined,
  InteractionClassNotSubscribed,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

///////////////////////////////
// Object Management Services //
///////////////////////////////

// 6.2
virtual
ObjectHandle                          // returned C3
registerObjectInstance (
        ObjectClassHandle  theClass,  // supplied C1
  const char              *theObject) // supplied C4
throw (
  ObjectClassNotDefined,
  ObjectClassNotPublished,
  ObjectAlreadyRegistered,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
```

```
  RTIinternalError);

virtual
ObjectHandle                          // returned C3
registerObjectInstance (
        ObjectClassHandle theClass)  // supplied C1
throw (
  ObjectClassNotDefined,
  ObjectClassNotPublished,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 6.4
virtual
EventRetractionHandle                                  // returned C3
updateAttributeValues (
        ObjectHandle                  theObject,     // supplied C1
  const AttributeHandleValuePairSet& theAttributes, // supplied C4
  const FedTime&                      theTime,       // supplied C4
  const char                         *theTag)        // supplied C4
throw (
  ObjectNotKnown,
  AttributeNotDefined,
  AttributeNotOwned,
  InvalidFederationTime,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

virtual
void updateAttributeValues (
        ObjectHandle                  theObject,     // supplied C1
  const AttributeHandleValuePairSet& theAttributes, // supplied C4
  const char                         *theTag)        // supplied C4
throw (
  ObjectNotKnown,
  AttributeNotDefined,
  AttributeNotOwned,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 6.6
virtual
EventRetractionHandle                                   // returned C3
sendInteraction (
        InteractionClassHandle       theInteraction, // supplied C1
  const ParameterHandleValuePairSet& theParameters,  // supplied C4
```

```
  const FedTime&                     theTime,        // supplied C4
  const char                        *theTag)         // supplied C4
throw (
  InteractionClassNotDefined,
  InteractionClassNotPublished,
  InteractionParameterNotDefined,
  InvalidFederationTime,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

virtual
void sendInteraction (
        InteractionClassHandle       theInteraction, // supplied C1
  const ParameterHandleValuePairSet& theParameters,  // supplied C4
  const char                        *theTag)         // supplied C4
throw (
  InteractionClassNotDefined,
  InteractionClassNotPublished,
  InteractionParameterNotDefined,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 6.8
virtual
EventRetractionHandle                 // returned C3
deleteObjectInstance (
        ObjectHandle    theObject,    // supplied C1
  const FedTime&        theTime,      // supplied C4
  const char           *theTag)       // supplied C4
throw (
  ObjectNotKnown,
  DeletePrivilegeNotHeld,
  InvalidFederationTime,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

virtual
void deleteObjectInstance (
        ObjectHandle    theObject,    // supplied C1
  const char           *theTag)       // supplied C4
throw (
  ObjectNotKnown,
  DeletePrivilegeNotHeld,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
```

```
  RestoreInProgress,
  RTIinternalError);

// 6.10
virtual
void localDeleteObjectInstance (
  ObjectHandle    theObject)       // supplied C1
throw (
  ObjectNotKnown,
  FederateOwnsAttributes,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 6.11
virtual
void changeAttributeTransportationType (
        ObjectHandle             theObject,     // supplied C1
  const AttributeHandleSet&      theAttributes, // supplied C4
        TransportationHandle     theType)       // supplied C1
throw (
  ObjectNotKnown,
  AttributeNotDefined,
  AttributeNotOwned,
  InvalidTransportationHandle,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 6.12
virtual
void changeInteractionTransportationType (
  InteractionClassHandle theClass, // supplied C1
  TransportationHandle   theType)  // supplied C1
throw (
  InteractionClassNotDefined,
  InteractionClassNotPublished,
  InvalidTransportationHandle,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 6.15
virtual
void requestObjectAttributeValueUpdate (
        ObjectHandle        theObject,     // supplied C1
  const AttributeHandleSet& theAttributes) // supplied C4
throw (
  ObjectNotKnown,
```

```
  AttributeNotDefined,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

virtual
void requestClassAttributeValueUpdate (
        ObjectClassHandle   theClass,      // supplied C1
  const AttributeHandleSet& theAttributes) // supplied C4
throw (
  ObjectClassNotDefined,
  AttributeNotDefined,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

///////////////////////////////////
// Ownership Management Services //
///////////////////////////////////

// 7.2
virtual
void unconditionalAttributeOwnershipDivestiture (
        ObjectHandle                theObject,     // supplied C1
  const AttributeHandleSet&         theAttributes) // supplied C4
throw (
  ObjectNotKnown,
  AttributeNotDefined,
  AttributeNotOwned,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 7.3
virtual
void negotiatedAttributeOwnershipDivestiture (
        ObjectHandle                theObject,     // supplied C1
  const AttributeHandleSet&         theAttributes, // supplied C4
  const char                       *theTag)        // supplied C4
throw (
  ObjectNotKnown,
  AttributeNotDefined,
  AttributeNotOwned,
  AttributeAlreadyBeingDivested,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);
```

```
// 7.7
virtual
void attributeOwnershipAcquisition (
        ObjectHandle        theObject,         // supplied C1
  const AttributeHandleSet& desiredAttributes, // supplied C4
  const char               *theTag)            // supplied C4
throw (
  ObjectNotKnown,
  ObjectClassNotPublished,
  AttributeNotDefined,
  AttributeNotPublished,
  FederateOwnsAttributes,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 7.8
virtual
void attributeOwnershipAcquisitionIfAvailable (
        ObjectHandle        theObject,         // supplied C1
  const AttributeHandleSet& desiredAttributes) // supplied C4
throw (
  ObjectNotKnown,
  ObjectClassNotPublished,
  AttributeNotDefined,
  AttributeNotPublished,
  FederateOwnsAttributes,
  AttributeAlreadyBeingAcquired,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 7.11
virtual
AttributeHandleSet*                          // returned C6
attributeOwnershipReleaseResponse (
        ObjectHandle        theObject,     // supplied C1
  const AttributeHandleSet& theAttributes) // supplied C4
throw (
  ObjectNotKnown,
  AttributeNotDefined,
  AttributeNotOwned,
  FederateWasNotAskedToReleaseAttribute,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 7.12
```

```
virtual
void cancelNegotiatedAttributeOwnershipDivestiture (
        ObjectHandle        theObject,     // supplied C1
  const AttributeHandleSet& theAttributes) // supplied C4
throw (
  ObjectNotKnown,
  AttributeNotDefined,
  AttributeNotOwned,
  AttributeDivestitureWasNotRequested,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 7.13
virtual
void cancelAttributeOwnershipAcquisition (
        ObjectHandle        theObject,     // supplied C1
  const AttributeHandleSet& theAttributes) // supplied C4
throw (
  ObjectNotKnown,
  AttributeNotDefined,
  AttributeAlreadyOwned,
  AttributeAcquisitionWasNotRequested,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 7.15
virtual
void queryAttributeOwnership (
  ObjectHandle    theObject,    // supplied C1
  AttributeHandle theAttribute) // supplied C1
throw (
  ObjectNotKnown,
  AttributeNotDefined,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 7.17
virtual
Boolean                          // returned C3
isAttributeOwnedByFederate (
  ObjectHandle    theObject,     // supplied C1
  AttributeHandle theAttribute)  // supplied C1
throw (
  ObjectNotKnown,
  AttributeNotDefined,
  FederateNotExecutionMember,
```

```
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

//////////////////////////////
// Time Management Services //
//////////////////////////////

// 8.2
virtual
void enableTimeRegulation (
  const FedTime& theFederateTime,  // supplied C4
  const FedTime& theLookahead)     // supplied C4
throw (
  TimeRegulationAlreadyEnabled,
  EnableTimeRegulationPending,
  TimeAdvanceAlreadyInProgress,
  InvalidFederationTime,
  InvalidLookahead,
  ConcurrentAccessAttempted,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 8.4
virtual
void disableTimeRegulation ()
throw (
  TimeRegulationWasNotEnabled,
  ConcurrentAccessAttempted,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 8.5
virtual
void enableTimeConstrained ()
throw (
  TimeConstrainedAlreadyEnabled,
  EnableTimeConstrainedPending,
  TimeAdvanceAlreadyInProgress,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 8.7
virtual
void disableTimeConstrained ()
throw (
  TimeConstrainedWasNotEnabled,
```

```
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 8.8
virtual
void timeAdvanceRequest (
 const  FedTime& theTime) // supplied C4
throw (
  InvalidFederationTime,
  FederationTimeAlreadyPassed,
  TimeAdvanceAlreadyInProgress,
  EnableTimeRegulationPending,
  EnableTimeConstrainedPending,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 8.9
virtual
void timeAdvanceRequestAvailable (
const FedTime& theTime) // supplied C4
  throw (
    InvalidFederationTime,
    FederationTimeAlreadyPassed,
    TimeAdvanceAlreadyInProgress,
    EnableTimeRegulationPending,
    EnableTimeConstrainedPending,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 8.10
virtual
void nextEventRequest (
  const FedTime& theTime) // supplied C4
throw (
  InvalidFederationTime,
  FederationTimeAlreadyPassed,
  TimeAdvanceAlreadyInProgress,
  EnableTimeRegulationPending,
  EnableTimeConstrainedPending,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 8.11
```

```
virtual
void nextEventRequestAvailable (
  const FedTime& theTime) // supplied C4
throw (
  InvalidFederationTime,
  FederationTimeAlreadyPassed,
  TimeAdvanceAlreadyInProgress,
  EnableTimeRegulationPending,
  EnableTimeConstrainedPending,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 8.12
virtual
void flushQueueRequest (
  const FedTime& theTime) // supplied C4
throw (
  InvalidFederationTime,
  FederationTimeAlreadyPassed,
  TimeAdvanceAlreadyInProgress,
  EnableTimeRegulationPending,
  EnableTimeConstrainedPending,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 8.14
virtual
void enableAsynchronousDelivery()
  throw (
    AsynchronousDeliveryAlreadyEnabled,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 8.15
virtual
void disableAsynchronousDelivery()
  throw (
    AsynchronousDeliveryAlreadyDisabled,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError);

// 8.16
virtual
```

```
void queryLBTS (
  FedTime& theTime) // returned C5
throw (
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 8.17
virtual
void queryFederateTime (
  FedTime& theTime) // returned C5
throw (
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 8.18
virtual
void queryMinNextEventTime (
  FedTime& theTime) // returned C5
throw (
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 8.19
virtual
void modifyLookahead (
  const FedTime& theLookahead) // supplied C4
throw (
  InvalidLookahead,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 8.20
virtual
void queryLookahead (
   FedTime& theTime) // returned C5
throw (
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 8.21
```

```
virtual
void retract (
  EventRetractionHandle theHandle) // supplied C1
throw (
  InvalidRetractionHandle,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 8.23
virtual
void changeAttributeOrderType (
        ObjectHandle        theObject,     // supplied C1
  const AttributeHandleSet& theAttributes, // supplied C4
        OrderingHandle      theType)       // supplied C1
throw (
  ObjectNotKnown,
  AttributeNotDefined,
  AttributeNotOwned,
  InvalidOrderingHandle,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 8.24
virtual
void changeInteractionOrderType (
  InteractionClassHandle theClass, // supplied C1
  OrderingHandle         theType)  // supplied C1
throw (
  InteractionClassNotDefined,
  InteractionClassNotPublished,
  InvalidOrderingHandle,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

///////////////////////////////////
// Data Distribution Management //
///////////////////////////////////

// 9.2
virtual
Region*                           // returned C6
createRegion (
  SpaceHandle theSpace,           // supplied C1
  ULong       numberOfExtents)    // supplied C1
throw (
  SpaceNotDefined,
```

```
  InvalidExtents,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 9.3
virtual
void notifyAboutRegionModification (
  Region &theRegion)  // supplied C4
throw (
  RegionNotKnown,
  InvalidExtents,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 9.4
virtual
void deleteRegion (
  Region *theRegion) // supplied C1
throw (
  RegionNotKnown,
  RegionInUse,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 9.5
virtual
ObjectHandle                                    // returned C3
registerObjectInstanceWithRegion (
        ObjectClassHandle theClass,             // supplied C1
  const char             *theObject,            // supplied C4
        AttributeHandle   theAttributes[],      // supplied C4
        Region           *theRegions[],         // supplied C4
        ULong             theNumberOfHandles)   // supplied C1
throw (
  ObjectClassNotDefined,
  ObjectClassNotPublished,
  AttributeNotDefined,
  AttributeNotPublished,
  RegionNotKnown,
  InvalidRegionContext,
  ObjectAlreadyRegistered,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);
```

```
virtual
ObjectHandle                                // returned C3
registerObjectInstanceWithRegion (
  ObjectClassHandle theClass,               // supplied C1
  AttributeHandle   theAttributes[],        // supplied C4
  Region           *theRegions[],           // supplied C4
  ULong             theNumberOfHandles)     // supplied C1
throw (
  ObjectClassNotDefined,
  ObjectClassNotPublished,
  AttributeNotDefined,
  AttributeNotPublished,
  RegionNotKnown,
  InvalidRegionContext,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 9.6
virtual
void associateRegionForUpdates (
        Region              &theRegion,     // supplied C4
        ObjectHandle         theObject,     // supplied C1
  const AttributeHandleSet &theAttributes) // supplied C4
throw (
  ObjectNotKnown,
  AttributeNotDefined,
  InvalidRegionContext,
  RegionNotKnown,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 9.7
virtual
void unassociateRegionForUpdates (
  Region       &theRegion,     // supplied C4
  ObjectHandle  theObject)     // supplied C1
throw (
  ObjectNotKnown,
  InvalidRegionContext,
  RegionNotKnown,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 9.8
virtual
```

```
void subscribeObjectClassAttributesWithRegion (
        ObjectClassHandle   theClass,      // supplied C1
        Region             &theRegion,     // supplied C4
  const AttributeHandleSet &attributeList, // supplied C4
        Boolean        active = RTI_TRUE)
throw (
  ObjectClassNotDefined,
  AttributeNotDefined,
  RegionNotKnown,
  InvalidRegionContext,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 9.9
virtual
void unsubscribeObjectClassWithRegion (
  ObjectClassHandle theClass,          // supplied C1
  Region           &theRegion)         // supplied C4
throw (
  ObjectClassNotDefined,
  RegionNotKnown,
  ObjectClassNotSubscribed,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 9.10
virtual
void subscribeInteractionClassWithRegion (
  InteractionClassHandle theClass,        // supplied C1
  Region                &theRegion,       // supplied C4
  Boolean           active = RTI_TRUE)
throw (
  InteractionClassNotDefined,
  RegionNotKnown,
  InvalidRegionContext,
  FederateLoggingServiceCalls,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 9.11
virtual
void unsubscribeInteractionClassWithRegion (
  InteractionClassHandle theClass,  // supplied C1
  Region                &theRegion) // supplied C4
throw (
  InteractionClassNotDefined,
```

```
  InteractionClassNotSubscribed,
  RegionNotKnown,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 9.12
virtual
EventRetractionHandle                                 // returned C3
sendInteractionWithRegion (
        InteractionClassHandle       theInteraction, // supplied C1
  const ParameterHandleValuePairSet &theParameters,  // supplied C4
  const FedTime&                     theTime,        // supplied C4
  const char                        *theTag,         // supplied C4
  const Region                      &theRegion)      // supplied C4
throw (
  InteractionClassNotDefined,
  InteractionClassNotPublished,
  InteractionParameterNotDefined,
  InvalidFederationTime,
  RegionNotKnown,
  InvalidRegionContext,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

virtual
void sendInteractionWithRegion (
        InteractionClassHandle       theInteraction, // supplied C1
  const ParameterHandleValuePairSet &theParameters,  // supplied C4
  const char                        *theTag,         // supplied C4
  const Region                      &theRegion)      // supplied C4
throw (
  InteractionClassNotDefined,
  InteractionClassNotPublished,
  InteractionParameterNotDefined,
  RegionNotKnown,
  InvalidRegionContext,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 9.13
virtual
void requestClassAttributeValueUpdateWithRegion (
        ObjectClassHandle   theClass,      // supplied C1
  const AttributeHandleSet &theAttributes, // supplied C4
  const Region             &theRegion)     // supplied C4
throw (
```

```
  ObjectClassNotDefined,
  AttributeNotDefined,
  RegionNotKnown,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

//////////////////////////
// RTI Support Services //
//////////////////////////

// 10.2
virtual
ObjectClassHandle      // returned C3
getObjectClassHandle (
  const char *theName) // supplied C4
throw (
  NameNotFound,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  RTIinternalError);

// 10.3
virtual
char *                          // returned C6
getObjectClassName (
  ObjectClassHandle theHandle) // supplied C1
throw (
  ObjectClassNotDefined,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  RTIinternalError);

// 10.4
virtual
AttributeHandle                         // returned C3
getAttributeHandle (
  const char             *theName,    // supplied C4
        ObjectClassHandle whichClass) // supplied C1
throw (
  ObjectClassNotDefined,
  NameNotFound,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  RTIinternalError);

// 10.5
virtual
char *                           // returned C6
getAttributeName (
  AttributeHandle   theHandle,  // supplied C1
  ObjectClassHandle whichClass) // supplied C1
throw (
```

```
  ObjectClassNotDefined,
  AttributeNotDefined,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  RTIinternalError);

// 10.6
virtual
InteractionClassHandle      // returned C3
getInteractionClassHandle (
  const char *theName)      // supplied C4
throw (
  NameNotFound,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  RTIinternalError);

// 10.7
virtual
char *                                // returned C6
getInteractionClassName (
  InteractionClassHandle theHandle) // supplied C1
throw (
  InteractionClassNotDefined,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  RTIinternalError);

// 10.8
virtual
ParameterHandle                                // returned C3
getParameterHandle (
  const char *theName,                         // supplied C4
        InteractionClassHandle whichClass) // supplied C1
throw (
  InteractionClassNotDefined,
  NameNotFound,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  RTIinternalError);

// 10.9
virtual
char *                                  // returned C6
getParameterName (
  ParameterHandle        theHandle,  // supplied C1
  InteractionClassHandle whichClass) // supplied C1
throw (
  InteractionClassNotDefined,
  InteractionParameterNotDefined,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  RTIinternalError);

// 10.10
```

```
virtual
ObjectHandle                 // returned C3
getObjectInstanceHandle (
  const char *theName)       // supplied C4
throw (
    ObjectNotKnown,
    FederateNotExecutionMember,
    ConcurrentAccessAttempted,
    RTIinternalError);

// 10.11
virtual
char *                      // returned C6
getObjectInstanceName (
  ObjectHandle theHandle)  // supplied C1
throw (
  ObjectNotKnown,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  RTIinternalError);

// 10.12
virtual
SpaceHandle                 // returned C3
getRoutingSpaceHandle (
  const char *theName)     // supplied C4
throw (
  NameNotFound,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  RTIinternalError);

// 10.13
virtual
char *                           // returned C6
getRoutingSpaceName (
  SpaceHandle theHandle) // supplied C4
throw (
  SpaceNotDefined,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  RTIinternalError);

// 10.14
virtual
DimensionHandle                    // returned C3
getDimensionHandle (
  const char         *theName,    // supplied C4
        SpaceHandle   whichSpace) // supplied C1
throw (
  SpaceNotDefined,
  NameNotFound,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  RTIinternalError);
```

```
// 10.15
virtual
char *                          // returned C6
getDimensionName (
  DimensionHandle theHandle,  // supplied C1
  SpaceHandle     whichSpace) // supplied C1
throw (
  SpaceNotDefined,
  DimensionNotDefined,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  RTIinternalError);

// 10.16
virtual
SpaceHandle                       // returned C3
getAttributeRoutingSpaceHandle (
  AttributeHandle   theHandle,   // supplied C1
  ObjectClassHandle whichClass)  // supplied C1
throw (
  ObjectClassNotDefined,
  AttributeNotDefined,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  RTIinternalError);

// 10.17
virtual
ObjectClassHandle            // returned C3
getObjectClass (
  ObjectHandle theObject)    // supplied C1
throw (
  ObjectNotKnown,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  RTIinternalError);

// 10.18
virtual
SpaceHandle                                // returned C3
getInteractionRoutingSpaceHandle (
  InteractionClassHandle   theHandle)   // supplied C1
throw (
  InteractionClassNotDefined,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  RTIinternalError);

// 10.19
virtual
TransportationHandle      // returned C3
getTransportationHandle (
  const char *theName)    // supplied C4
throw (
```

```
  NameNotFound,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  RTIinternalError);

// 10.20
virtual
char *                              // returned C6
getTransportationName (
  TransportationHandle theHandle) // supplied C1
throw (
  InvalidTransportationHandle,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  RTIinternalError);

// 10.21
virtual
OrderingHandle         // returned C3
getOrderingHandle (
  const char *theName) // supplied C4
throw (
  NameNotFound,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  RTIinternalError);

// 10.22
virtual
char *                       // returned C6
getOrderingName (
  OrderingHandle theHandle) // supplied C1
throw (
  InvalidOrderingHandle,
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  RTIinternalError);

// 10.23
virtual
void enableClassRelevanceAdvisorySwitch()
throw(
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 10.24
virtual
void disableClassRelevanceAdvisorySwitch()
throw(
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
```

```
  RestoreInProgress,
  RTIinternalError);

// 10.25
virtual
void enableAttributeRelevanceAdvisorySwitch()
throw(
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 10.26
virtual
void disableAttributeRelevanceAdvisorySwitch()
throw(
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 10.27
virtual
void enableAttributeScopeAdvisorySwitch()
throw(
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 10.28
virtual
void disableAttributeScopeAdvisorySwitch()
throw(
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 10.29
virtual
void enableInteractionRelevanceAdvisorySwitch()
throw(
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

// 10.30
virtual
```

```
void disableInteractionRelevanceAdvisorySwitch()
throw(
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError);

//
virtual
Boolean // returned C3
tick ()
throw (
  SpecifiedSaveLabelDoesNotExist,
  ConcurrentAccessAttempted,
  RTIinternalError);

virtual
Boolean             // returned C3
tick (
  TickTime minimum, // supplied C1
  TickTime maximum) // supplied C1
throw (
  SpecifiedSaveLabelDoesNotExist,
  ConcurrentAccessAttempted,
  RTIinternalError);

RTIambassador()
throw (
  MemoryExhausted,
  RTIinternalError);

virtual
~RTIambassador()
throw (RTIinternalError);

virtual
RegionToken
getRegionToken(
  Region *)
throw(
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  RegionNotKnown,
  RTIinternalError);

virtual
Region *
getRegion(
  RegionToken)
throw(
  FederateNotExecutionMember,
  ConcurrentAccessAttempted,
  RegionNotKnown,
  RTIinternalError);
```

A.6 federateAmbServices13.h

```
//File federateAmbServices13.h
//Included in RTI13.h

//               RTI Parameter Passing Memory Conventions
//
// C1  In parameter by value.
// C2  Out parameter by pointer value.
// C3  Function return by value.
// C4  In parameter by const pointer value.  Caller provides memory.
//     Caller may free memory or overwrite it upon completion of
//     the call.  Callee must copy during the call anything it
//     wishes to save beyond completion of the call.  Parameter
//     type must define const accessor methods.
// C5  Out parameter by pointer value.  Caller provides reference to object.
//     Callee constructs an instance on the heap (new) and returns.
//     The caller destroys the instance (delete) at its leisure.
// C6  Function return by pointer value.  Callee constructs an instance on
//     the heap (new) and returns a reference.  The caller destroys the
//     instance (delete) at its leisure.
//

////////////////////////////////////
// Federation Management Services //
////////////////////////////////////

// 4.7
virtual void synchronizationPointRegistrationSucceeded (
  const char *label) // supplied C4)
throw (
  FederateInternalError) = 0;

virtual void synchronizationPointRegistrationFailed (
  const char *label) // supplied C4)
throw (
  FederateInternalError) = 0;

// 4.8
virtual void announceSynchronizationPoint (
  const char *label, // supplied C4
  const char *tag)   // supplied C4
throw (
  FederateInternalError) = 0;

// 4.10
virtual void federationSynchronized (
  const char *label) // supplied C4)
throw (
  FederateInternalError) = 0;

// 4.12
virtual void initiateFederateSave (
  const char *label) // supplied C4
throw (
```

```
  UnableToPerformSave,
  FederateInternalError) = 0;

// 4.15
virtual void federationSaved ()
throw (
  FederateInternalError) = 0;

virtual void federationNotSaved ()
throw (
  FederateInternalError) = 0;

// 4.17
virtual void requestFederationRestoreSucceeded (
  const char *label) // supplied C4
throw (
  FederateInternalError) = 0;

virtual void requestFederationRestoreFailed (
  const char *label,  // supplied C4
  const char *reason) // supplied C4
throw (
  FederateInternalError) = 0;

// 4.18
virtual void federationRestoreBegun ()
throw (
  FederateInternalError) = 0;

// 4.19
virtual void initiateFederateRestore (
  const char *label,   // supplied C4
        FederateHandle handle)  // supplied C1
throw (
  SpecifiedSaveLabelDoesNotExist,
  CouldNotRestore,
  FederateInternalError) = 0;

// 4.21
virtual void federationRestored ()
throw (
  FederateInternalError) = 0;

virtual void federationNotRestored ()
throw (
  FederateInternalError) = 0;

//////////////////////////////////////
// Declaration Management Services //
//////////////////////////////////////

// 5.10
virtual void startRegistrationForObjectClass (
        ObjectClassHandle   theClass)      // supplied C1
throw (
```

```
  ObjectClassNotPublished,
  FederateInternalError) = 0;

// 5.11
virtual void stopRegistrationForObjectClass (
        ObjectClassHandle   theClass)      // supplied C1
throw (
  ObjectClassNotPublished,
  FederateInternalError) = 0;

// 5.12
virtual void turnInteractionsOn (
  InteractionClassHandle theHandle) // supplied C1
throw (
  InteractionClassNotPublished,
  FederateInternalError) = 0;

// 5.13
virtual void turnInteractionsOff (
  InteractionClassHandle theHandle) // supplied C1
throw (
  InteractionClassNotPublished,
  FederateInternalError) = 0;

////////////////////////////////
// Object Management Services //
////////////////////////////////

// 6.3
virtual void discoverObjectInstance (
        ObjectHandle          theObject,      // supplied C1
        ObjectClassHandle     theObjectClass, // supplied C1
  const char *                theObjectName)  // supplied C4
throw (
  CouldNotDiscover,
  ObjectClassNotKnown,
  FederateInternalError) = 0;

// 6.5
virtual void reflectAttributeValues (
        ObjectHandle                  theObject,     // supplied C1
  const AttributeHandleValuePairSet& theAttributes, // supplied C4
  const FedTime&                      theTime,       // supplied C1
  const char                         *theTag,        // supplied C4
        EventRetractionHandle         theHandle)     // supplied C1
throw (
  ObjectNotKnown,
  AttributeNotKnown,
  FederateOwnsAttributes,
  InvalidFederationTime,
  FederateInternalError) = 0;

virtual void reflectAttributeValues (
        ObjectHandle                  theObject,     // supplied C1
  const AttributeHandleValuePairSet& theAttributes, // supplied C4
```

```
  const char                         *theTag)        // supplied C4
throw (
  ObjectNotKnown,
  AttributeNotKnown,
  FederateOwnsAttributes,
  FederateInternalError) = 0;

// 6.7
virtual void receiveInteraction (
        InteractionClassHandle       theInteraction, // supplied C1
  const ParameterHandleValuePairSet& theParameters,  // supplied C4
  const FedTime&                     theTime,        // supplied C4
  const char                         *theTag,        // supplied C4
        EventRetractionHandle        theHandle)      // supplied C1
throw (
  InteractionClassNotKnown,
  InteractionParameterNotKnown,
  InvalidFederationTime,
  FederateInternalError) = 0;

virtual void receiveInteraction (
        InteractionClassHandle       theInteraction, // supplied C1
  const ParameterHandleValuePairSet& theParameters,  // supplied C4
  const char                         *theTag)        // supplied C4
throw (
  InteractionClassNotKnown,
  InteractionParameterNotKnown,
  FederateInternalError) = 0;

// 6.9
virtual void removeObjectInstance (
        ObjectHandle          theObject, // supplied C1
  const FedTime&              theTime,   // supplied C4
  const char                  *theTag,   // supplied C4
        EventRetractionHandle theHandle) // supplied C1
throw (
  ObjectNotKnown,
  InvalidFederationTime,
  FederateInternalError) = 0;

virtual void removeObjectInstance (
        ObjectHandle          theObject, // supplied C1
  const char                  *theTag)   // supplied C4
throw (
  ObjectNotKnown,
  FederateInternalError) = 0;

// 6.13
virtual void attributesInScope (
        ObjectHandle        theObject,     // supplied C1
  const AttributeHandleSet& theAttributes) // supplied C4
throw (
  ObjectNotKnown,
  AttributeNotKnown,
  FederateInternalError) = 0;
```

```
// 6.14
virtual void attributesOutOfScope (
        ObjectHandle        theObject,     // supplied C1
  const AttributeHandleSet& theAttributes) // supplied C4
throw (
  ObjectNotKnown,
  AttributeNotKnown,
  FederateInternalError) = 0;

// 6.16
virtual void provideAttributeValueUpdate (
        ObjectHandle        theObject,     // supplied C1
  const AttributeHandleSet& theAttributes) // supplied C4
throw (
  ObjectNotKnown,
  AttributeNotKnown,
  AttributeNotOwned,
  FederateInternalError) = 0;

// 6.17
virtual void turnUpdatesOnForObjectInstance (
        ObjectHandle        theObject,     // supplied C1
  const AttributeHandleSet& theAttributes) // supplied C4
throw (
  ObjectNotKnown,
  AttributeNotOwned,
  FederateInternalError) = 0;

// 6.18
virtual void turnUpdatesOffForObjectInstance (
        ObjectHandle        theObject,      // supplied C1
  const AttributeHandleSet& theAttributes) // supplied C4
throw (
  ObjectNotKnown,
  AttributeNotOwned,
  FederateInternalError) = 0;

///////////////////////////////////
// Ownership Management Services //
///////////////////////////////////

// 7.4
virtual void requestAttributeOwnershipAssumption (
        ObjectHandle        theObject,         // supplied C1
  const AttributeHandleSet& offeredAttributes, // supplied C4
  const char               *theTag)            // supplied C4
throw (
  ObjectNotKnown,
  AttributeNotKnown,
  AttributeAlreadyOwned,
  AttributeNotPublished,
  FederateInternalError) = 0;

// 7.5
```

```
virtual void attributeOwnershipDivestitureNotification (
        ObjectHandle        theObject,          // supplied C1
  const AttributeHandleSet& releasedAttributes) // supplied C4
throw (
  ObjectNotKnown,
  AttributeNotKnown,
  AttributeNotOwned,
  AttributeDivestitureWasNotRequested,
  FederateInternalError) = 0;

// 7.6
virtual void attributeOwnershipAcquisitionNotification (
        ObjectHandle        theObject,         // supplied C1
  const AttributeHandleSet& securedAttributes) // supplied C4
throw (
  ObjectNotKnown,
  AttributeNotKnown,
  AttributeAcquisitionWasNotRequested,
  AttributeAlreadyOwned,
  AttributeNotPublished,
  FederateInternalError) = 0;

// 7.9
virtual void attributeOwnershipUnavailable (
        ObjectHandle        theObject,         // supplied C1
  const AttributeHandleSet& theAttributes) // supplied C4
throw (
  ObjectNotKnown,
  AttributeNotKnown,
  AttributeNotDefined,
  AttributeAlreadyOwned,
  AttributeAcquisitionWasNotRequested,
  FederateInternalError) = 0;

// 7.10
virtual void requestAttributeOwnershipRelease (
        ObjectHandle        theObject,           // supplied C1
  const AttributeHandleSet& candidateAttributes, // supplied C4
  const char               *theTag)              // supplied C4
throw (
  ObjectNotKnown,
  AttributeNotKnown,
  AttributeNotOwned,
  FederateInternalError) = 0;

// 7.14
virtual void confirmAttributeOwnershipAcquisitionCancellation (
        ObjectHandle        theObject,         // supplied C1
  const AttributeHandleSet& theAttributes) // supplied C4
throw (
  ObjectNotKnown,
  AttributeNotKnown,
  AttributeNotDefined,
  AttributeAlreadyOwned,
  AttributeAcquisitionWasNotCanceled,
```

```
  FederateInternalError) = 0;

// 7.16
virtual void informAttributeOwnership (
  ObjectHandle    theObject,    // supplied C1
  AttributeHandle theAttribute, // supplied C1
  FederateHandle  theOwner)     // supplied C1
throw (
  ObjectNotKnown,
  AttributeNotKnown,
  FederateInternalError) = 0;

virtual void attributeIsNotOwned (
  ObjectHandle    theObject,    // supplied C1
  AttributeHandle theAttribute) // supplied C1
throw (
  ObjectNotKnown,
  AttributeNotKnown,
  FederateInternalError) = 0;

virtual void attributeOwnedByRTI (
  ObjectHandle    theObject,    // supplied C1
  AttributeHandle theAttribute) // supplied C1
throw (
  ObjectNotKnown,
  AttributeNotKnown,
  FederateInternalError) = 0;

//////////////////////////////
// Time Management Services //
//////////////////////////////

// 8.3
virtual void timeRegulationEnabled (
 const  FedTime& theFederateTime) // supplied C4
throw (
  InvalidFederationTime,
  EnableTimeRegulationWasNotPending,
  FederateInternalError) = 0;

// 8.6
virtual void timeConstrainedEnabled (
  const FedTime& theFederateTime) // supplied C4
throw (
  InvalidFederationTime,
  EnableTimeConstrainedWasNotPending,
  FederateInternalError) = 0;

// 8.13
virtual void timeAdvanceGrant (
  const FedTime& theTime) // supplied C4
throw (
  InvalidFederationTime,
  TimeAdvanceWasNotInProgress,
  FederationTimeAlreadyPassed,
```

```
  FederateInternalError) = 0;

// 8.22
virtual void requestRetraction (
  EventRetractionHandle theHandle) // supplied C1
throw (
  EventNotKnown,
  FederateInternalError) = 0;


virtual ~FederateAmbassador()
throw (FederateInternalError) { ; }
```

A.7 NullFederateAmbassador13.h

```
//File NullFederateAmbassador13.h

#ifndef NullFederateAmbassador13_h
#define NullFederateAmbassador13_h

#include <RTI13.h>

//----------------------------------------------------------------
//                RTI Parameter Passing Memory Conventions
//
// C1  In parameter by value.
// C2  Out parameter by reference.
// C3  Function return by value.
// C4  In parameter by const reference.  Caller provides memory.
//     Caller may free memory or overwrite it upon completion of
//     the call.  Callee must copy during the call anything it
//     wishes to save beyond completion of the call.  Parameter
//     type must define const accessor methods.
// C5  Out parameter by reference.  Caller provides reference to object.
//     Callee constructs an instance on the heap (new) and returns.
//     The caller destroys the instance (delete) at its leisure.
// C6  Function return by reference.  Callee constructs an instance on
//     the heap (new) and returns a reference.  The caller destroys the
//     instance (delete) at its leisure.
//----------------------------------------------------------------

#ifndef NULL_AMBASSADOR_EXPORT
#define NULL_AMBASSADOR_EXPORT
#endif

namespace rti13
{

class NULL_AMBASSADOR_EXPORT
   NullFederateAmbassador : public FederateAmbassador
{
public:

NullFederateAmbassador() {}
virtual ~NullFederateAmbassador()
throw (FederateInternalError) {}

////////////////////////////////////
// Federation Management Services //
////////////////////////////////////

virtual void synchronizationPointRegistrationSucceeded (
  const char *label) // supplied C4)
throw (
  FederateInternalError) {}

virtual void synchronizationPointRegistrationFailed (
  const char *label) // supplied C4)
```

```
throw (
  FederateInternalError) {}

virtual void announceSynchronizationPoint (
  const char *label, // supplied C4
  const char *tag)   // supplied C4
throw (
  FederateInternalError) {}

virtual void federationSynchronized (
  const char *label) // supplied C4)
throw (
  FederateInternalError) {}

virtual void initiateFederateSave (
  const char *label) // supplied C4
throw (
  UnableToPerformSave,
  FederateInternalError) {}

virtual void federationSaved ()
throw (
  FederateInternalError) {}

virtual void federationNotSaved ()
throw (
  FederateInternalError) {}

virtual void requestFederationRestoreSucceeded (
  const char *label) // supplied C4
throw (
  FederateInternalError) {}

virtual void requestFederationRestoreFailed (
  const char *label,
  const char *reason) // supplied C4
throw (
  FederateInternalError) {}

virtual void federationRestoreBegun ()
throw (
  FederateInternalError) {}

virtual void initiateFederateRestore (
  const char               *label,   // supplied C4
        FederateHandle handle)  // supplied C1
throw (
  SpecifiedSaveLabelDoesNotExist,
  CouldNotRestore,
  FederateInternalError) {}

virtual void federationRestored ()
throw (
  FederateInternalError) {}
```

```
virtual void federationNotRestored ()
throw (
  FederateInternalError) {}

////////////////////////////////////
// Declaration Management Services //
////////////////////////////////////

virtual void startRegistrationForObjectClass (
        ObjectClassHandle   theClass)      // supplied C1
throw (
  ObjectClassNotPublished,
  FederateInternalError) {}

virtual void stopRegistrationForObjectClass (
        ObjectClassHandle   theClass)      // supplied C1
throw (
  ObjectClassNotPublished,
  FederateInternalError) {}

virtual void turnInteractionsOn (
  InteractionClassHandle theHandle) // supplied C1
throw (
  InteractionClassNotPublished,
  FederateInternalError) {}

virtual void turnInteractionsOff (
  InteractionClassHandle theHandle) // supplied C1
throw (
  InteractionClassNotPublished,
  FederateInternalError) {}

////////////////////////////////
// Object Management Services //
////////////////////////////////

virtual void discoverObjectInstance (
        ObjectHandle          theObject,      // supplied C1
        ObjectClassHandle     theObjectClass, // supplied C1
  const char*                      theObjectName)  // supplied C4
throw (
  CouldNotDiscover,
  ObjectClassNotKnown,
  FederateInternalError) {}

virtual void reflectAttributeValues (
        ObjectHandle                  theObject,     // supplied C1
  const AttributeHandleValuePairSet& theAttributes, // supplied C4
  const FedTime&                     theTime,       // supplied C1
  const char                             *theTag,        // supplied C4
        EventRetractionHandle        theHandle)     // supplied C1
throw (
  ObjectNotKnown,
  AttributeNotKnown,
  FederateOwnsAttributes,
```

```
  InvalidFederationTime,
  FederateInternalError) {}

virtual void reflectAttributeValues (
        ObjectHandle                 theObject,     // supplied C1
  const AttributeHandleValuePairSet& theAttributes, // supplied C4
  const char                             *theTag)        // supplied C4
throw (
  ObjectNotKnown,
  AttributeNotKnown,
  FederateOwnsAttributes,
  FederateInternalError) {}

// 4.6
virtual void receiveInteraction (
        InteractionClassHandle       theInteraction, // supplied C1
  const ParameterHandleValuePairSet& theParameters,  // supplied C4
  const FedTime&                     theTime,        // supplied C4
  const char                             *theTag,         // supplied C4
        EventRetractionHandle        theHandle)      // supplied C1
throw (
  InteractionClassNotKnown,
  InteractionParameterNotKnown,
  InvalidFederationTime,
  FederateInternalError) {}

virtual void receiveInteraction (
        InteractionClassHandle       theInteraction, // supplied C1
  const ParameterHandleValuePairSet& theParameters,  // supplied C4
  const char                             *theTag)         // supplied C4
throw (
  InteractionClassNotKnown,
  InteractionParameterNotKnown,
  FederateInternalError) {}

virtual void removeObjectInstance (
        ObjectHandle          theObject, // supplied C1
  const FedTime&              theTime,   // supplied C4
  const char                      *theTag,    // supplied C4
        EventRetractionHandle theHandle) // supplied C1
throw (
  ObjectNotKnown,
  InvalidFederationTime,
  FederateInternalError) {}

virtual void removeObjectInstance (
        ObjectHandle          theObject, // supplied C1
  const char                      *theTag)    // supplied C4
throw (
  ObjectNotKnown,
  FederateInternalError) {}

virtual void attributesInScope (
        ObjectHandle        theObject,     // supplied C1
  const AttributeHandleSet& theAttributes) // supplied C4
```

```
throw (
  ObjectNotKnown,
  AttributeNotKnown,
  FederateInternalError) {}

virtual void attributesOutOfScope (
        ObjectHandle        theObject,     // supplied C1
  const AttributeHandleSet& theAttributes) // supplied C4
throw (
  ObjectNotKnown,
  AttributeNotKnown,
  FederateInternalError) {}

virtual void provideAttributeValueUpdate (
        ObjectHandle        theObject,     // supplied C1
  const AttributeHandleSet& theAttributes) // supplied C4
throw (
  ObjectNotKnown,
  AttributeNotKnown,
  AttributeNotOwned,
  FederateInternalError) {}

virtual void turnUpdatesOnForObjectInstance (
        ObjectHandle        theObject,     // supplied C1
  const AttributeHandleSet& theAttributes) // supplied C4
throw (
  ObjectNotKnown,
  AttributeNotOwned,
  FederateInternalError) {}

virtual void turnUpdatesOffForObjectInstance (
        ObjectHandle        theObject,      // supplied C1
  const AttributeHandleSet& theAttributes) // supplied C4
throw (
  ObjectNotKnown,
  AttributeNotOwned,
  FederateInternalError) {}

///////////////////////////////////
// Ownership Management Services //
///////////////////////////////////

virtual void requestAttributeOwnershipAssumption (
        ObjectHandle        theObject,         // supplied C1
  const AttributeHandleSet& offeredAttributes, // supplied C4
  const char                    *theTag)            // supplied C4
throw (
  ObjectNotKnown,
  AttributeNotKnown,
  AttributeAlreadyOwned,
  AttributeNotPublished,
  FederateInternalError) {}

virtual void attributeOwnershipDivestitureNotification (
        ObjectHandle        theObject,          // supplied C1
```

```
  const AttributeHandleSet& releasedAttributes) // supplied C4
throw (
  ObjectNotKnown,
  AttributeNotKnown,
  AttributeNotOwned,
  AttributeDivestitureWasNotRequested,
  FederateInternalError) {}

virtual void attributeOwnershipAcquisitionNotification (
        ObjectHandle        theObject,         // supplied C1
  const AttributeHandleSet& securedAttributes) // supplied C4
throw (
  ObjectNotKnown,
  AttributeNotKnown,
  AttributeAcquisitionWasNotRequested,
  AttributeAlreadyOwned,
  AttributeNotPublished,
  FederateInternalError) {}

virtual void attributeOwnershipUnavailable (
        ObjectHandle        theObject,         // supplied C1
  const AttributeHandleSet& theAttributes) // supplied C4
throw (
  ObjectNotKnown,
  AttributeNotKnown,
  AttributeAlreadyOwned,
  AttributeAcquisitionWasNotRequested,
  FederateInternalError) {}

virtual void requestAttributeOwnershipRelease (
        ObjectHandle        theObject,           // supplied C1
  const AttributeHandleSet& candidateAttributes, // supplied C4
  const char                    *theTag)             // supplied C4
throw (
  ObjectNotKnown,
  AttributeNotKnown,
  AttributeNotOwned,
  FederateInternalError) {}

virtual void confirmAttributeOwnershipAcquisitionCancellation (
        ObjectHandle        theObject,         // supplied C1
  const AttributeHandleSet& theAttributes) // supplied C4
throw (
  ObjectNotKnown,
  AttributeNotKnown,
  AttributeAlreadyOwned,
  AttributeAcquisitionWasNotCanceled,
  FederateInternalError) {}

virtual void informAttributeOwnership (
  ObjectHandle    theObject,    // supplied C1
  AttributeHandle theAttribute, // supplied C1
  FederateHandle  theOwner)     // supplied C1
throw (
  ObjectNotKnown,
```

```
  AttributeNotKnown,
  FederateInternalError) {}

virtual void attributeIsNotOwned (
  ObjectHandle    theObject,    // supplied C1
  AttributeHandle theAttribute) // supplied C1
throw (
  ObjectNotKnown,
  AttributeNotKnown,
  FederateInternalError) {}

virtual void attributeOwnedByRTI (
  ObjectHandle    theObject,    // supplied C1
  AttributeHandle theAttribute) // supplied C1
throw (
  ObjectNotKnown,
  AttributeNotKnown,
  FederateInternalError) {}

//////////////////////////////
// Time Management Services //
//////////////////////////////

virtual void timeRegulationEnabled (
 const  FedTime& theFederateTime) // supplied C4
throw (
  InvalidFederationTime,
  EnableTimeRegulationWasNotPending,
  FederateInternalError) {}

virtual void timeConstrainedEnabled (
  const FedTime& theFederateTime) // supplied C4
throw (
  InvalidFederationTime,
  EnableTimeConstrainedWasNotPending,
  FederateInternalError) {}

virtual void timeAdvanceGrant (
  const FedTime& theTime) // supplied C4
throw (
  InvalidFederationTime,
  TimeAdvanceWasNotInProgress,
  FederateInternalError) {}

virtual void requestRetraction (
  EventRetractionHandle theHandle) // supplied C1
throw (
  EventNotKnown,
  FederateInternalError) {}
};

} // End of namespace rti13

#endif // NullFederateAmbassador13_h
```

APPENDIX B JAVA APPLICATION PROGRAMMER INTERFACE (NORMATIVE)

B.1 EncodingHelpers.java

```
package hla.rti.jlc;

public class EncodingHelpers
{
   /**
    * Encodes a boolean value as a big-endian 32-bit integer in a 4-byte
    * buffer and returns the buffer.
    *
    * @param value Value to encode
    *
    * @return Buffer with encoded value
    */
   public static byte[] encodeBoolean(boolean value)
   {
      byte[] buffer = new byte[4];
      encodeBoolean(value, buffer, 0);
      return buffer;
   }

   /**
    * Encodes a boolean value as a big-endian 32-bit integer into the
    * specified buffer at the specified offset.
    *
    * @param value  Value to encode
    * @param buffer Buffer to store value in
    * @param offset Offset in buffer
    */
   public static void encodeBoolean(boolean value, byte[] buffer, int offset)
   {
      encodeInt(value ? 1 : 0, buffer, offset);
   }

   /**
    * Decodes a big-endian 32-bit integer in a buffer into a boolean.
    *
    * @param buffer Buffer containing encoded value
    *
    * @return Decoded value
    */
   public static boolean decodeBoolean(byte[] buffer)
   {
      return decodeBoolean(buffer, 0);
   }

   /**
    * Decodes a big-endian 32-bit integer in a 4-byte buffer at the
    * specified offset int a boolean value.
    *
    * @param buffer Buffer containing encoded value
    * @param offset Offset in buffer
    *
    * @return Decoded value
```

```
 */
public static boolean decodeBoolean(byte[] buffer, int offset)
{
   return decodeInt(buffer, offset) != 0;
}

/**
 * Encodes a byte in a buffer and returns the buffer.
 *
 * @param value Value to encode
 *
 * @return Buffer with encoded value
 */
public static byte[] encodeByte(int value)
{
   byte[] buffer = new byte[4];
   encodeByte(value, buffer, 0);
   return buffer;
}

/**
 * Encodes a byte into a buffer at the specified offset.
 *
 * @param value  Value to encode
 * @param buffer Buffer to store value in
 * @param offset Offset in buffer
 */
public static void encodeByte(int value, byte[] buffer, int offset)
{
   buffer[offset] = (byte)value;
}

/**
 * Decodes a byte from a buffer.
 *
 * @param buffer Buffer containing encoded value
 *
 * @return Decoded value.
 */
public static byte decodeByte(byte[] buffer)
{
   return decodeByte(buffer, 0);
}

/**
 * Decodes a byte from a buffer at the specified offset.
 *
 * @param buffer Buffer containing encoded value
 * @param offset Offset in buffer
 *
 * @return Decoded value.
 */
public static byte decodeByte(byte[] buffer, int offset)
{
   return (byte)(buffer[offset] & 0xff);
}
```

```
/**
 * Encodes a char value as a big-endian 16-bit integer into a 2-byte
 * buffer and returns the buffer.
 *
 * @param value Value to encode
 *
 * @return Buffer with encoded value
 */
public static byte[] encodeChar(char value)
{
   byte[] buffer = new byte[2];
   encodeChar(value, buffer, 0);
   return buffer;
}

/**
 * Encodes a char value as a big-endian 16-bit integer into a buffer at
 * the specified offset.
 *
 * @param value  Value to encode
 * @param buffer Buffer to encode value in
 * @param offset Offset in buffer
 */
public static void encodeChar(char value, byte[] buffer, int offset)
{
   int pos = offset;
   buffer[pos++] = (byte)((value >>>  8) & 0xFF);
   buffer[pos++] = (byte)((value >>>  0) & 0xFF);
}

/**
 * Decodes a big-endian 16-bit integer in a buffer into a char value.
 *
 * @param buffer Buffer containing encoded value
 *
 * @return Decoded value.
 */
public static char decodeChar(byte[] buffer)
{
   return decodeChar(buffer, 0);
}

/**
 * Decodes a big-endian 16-bit integer in a buffer at the specified
 * offset into a char value.
 *
 * @param buffer Buffer containing encoded value
 * @param offset Offset in buffer
 *
 * @return Decoded value.
 */
public static char decodeChar(byte[] buffer, int offset)
{
   int value = 0;
   int pos = offset;
   value += (buffer[pos++] & 0xff) <<  8;
   value += (buffer[pos++] & 0xff) <<  0;
```

```
   return (char)value;
}

/**
 * Converts a double value to a long and stores that long value as a
 * big-endian 64-bit integer in an 8-byte buffer and returns that
 * buffer.
 *
 * @param value Value to encode
 *
 * @return Buffer with encoded value
 */
public static byte[] encodeDouble(double value)
{
   byte[] buffer = new byte[8];
   encodeDouble(value, buffer, 0);
   return buffer;
}

/**
 * Converts a double value to a long and stores that long value as a
 * big-endian 64-bit integer into buffer at specified offset.
 *
 * @param value  Value to encode
 * @param buffer Buffer to store value in
 * @param offset Offset in buffer
 */
public static void encodeDouble(double value, byte[] buffer, int offset)
{
   encodeLong(Double.doubleToLongBits(value), buffer, offset);
}

/**
 * Decodes a big-endian 64-bit integer stored in buffer into a long
 * value and converts that to a double value, which is returned.
 *
 * @param buffer Buffer containing encoded value
 *
 * @return Decoded value
 */
public static double decodeDouble(byte[] buffer)
{
   return decodeDouble(buffer, 0);
}

/**
 * Decodes a big-endian 64-bit integer stored in buffer at specified
 * offset into a long value and converts that to a double value, which
 * is returned.
 *
 * @param buffer Buffer containing encoded value
 * @param offset Offset in buffer
 *
 * @return Decoded value
 */
public static double decodeDouble(byte[] buffer, int offset)
```

```
{
   return Double.longBitsToDouble(decodeLong(buffer, offset));
}

/**
 * Converts a float value to an int and stores that int value as a
 * big-endian 32-bit integer in a 4-byte buffer and returns that
 * buffer.
 *
 * @param value Value to encode
 *
 * @return Buffer with encoded value
 */
public static byte[] encodeFloat(float value)
{
   byte[] buffer = new byte[4];
   encodeFloat(value, buffer, 0);
   return buffer;
}

/**
 * Converts a float value to an int and stores that int value as a
 * big-endian 32-bit integer in the specified buffer at a specified
 * offset.
 *
 * @param value  Value to encode
 * @param buffer Buffer to store value in
 * @param offset Offset in buffer
 */
public static void encodeFloat(float value, byte[] buffer, int offset)
{
   encodeInt(Float.floatToIntBits(value), buffer, offset);
}

/**
 * Decodes a big-endian 32-bit integer stored in buffer into an int
 * value and converts that to a float value, which is returned.
 *
 * @param buffer Buffer containing encoded value
 *
 * @return Decoded value
 */
public static float decodeFloat(byte[] buffer)
{
   return decodeFloat(buffer, 0);
}

/**
 * Decodes a big-endian 32-bit integer stored in buffer at specified
 * offset into an int value and converts that to a float value, which is
 * returned.
 *
 * @param buffer Buffer containing encoded value
 * @param offset Offset in buffer
 *
 * @return Decoded value
 */
```

```
public static float decodeFloat(byte[] buffer, int offset)
{
   return Float.intBitsToFloat(decodeInt(buffer, offset));
}

/**
 * Encodes an int as a big-endian 32-bit integer into a 4-byte buffer
 * and returns the buffer.
 *
 * @param value Value to encode
 *
 * @return Buffer with encoded value
 */
public static byte[] encodeInt(int value)
{
   byte[] buffer = new byte[4];
   encodeInt(value, buffer, 0);
   return buffer;
}

/**
 * Encodes an int as a big-endian 32-bit integer into buffer at the
 * specified offset.
 *
 * @param value  Value to encode
 * @param buffer Buffer to store value in
 * @param offset Offset in buffer
 */
public static void encodeInt(int value, byte[] buffer, int offset)
{
   int pos = offset;
   buffer[pos++] = (byte)((value >>> 24) & 0xFF);
   buffer[pos++] = (byte)((value >>> 16) & 0xFF);
   buffer[pos++] = (byte)((value >>>  8) & 0xFF);
   buffer[pos++] = (byte)((value >>>  0) & 0xFF);
}

/**
 * Decodes a big-endian 32-bit integer stored in buffer into an int
 * value.
 *
 * @param buffer Buffer containing encoded value
 *
 * @return Decoded value
 */
public static int decodeInt(byte[] buffer)
{
   return decodeInt(buffer, 0);
}

/**
 * Decodes a big-endian 32-bit integer stored in buffer at specified
 * offset into an int value.
 *
 * @param buffer Buffer containing encoded value
 * @param offset Offset in buffer
 *
```

```
 * @return Decoded value
 */
public static int decodeInt(byte[] buffer, int offset)
{
   int value = 0;
   int pos = offset;
   value += (buffer[pos++] & 0xff) << 24;
   value += (buffer[pos++] & 0xff) << 16;
   value += (buffer[pos++] & 0xff) <<  8;
   value += (buffer[pos++] & 0xff) <<  0;

   return value;
}

/**
 * Encodes a long value as a big-endian 64-bit integer into a 8-byte
 * buffer and returns the buffer.
 *
 * @param value Value to encode
 *
 * @return Buffer with encoded value
 */
public static byte[] encodeLong(long value)
{
   byte[] buffer = new byte[8];
   encodeLong(value, buffer, 0);
   return buffer;
}

/**
 * Encodes a long value as a big-endian 64-bit integer into a buffer at
 * specified offset.
 *
 * @param value  Value to encode
 * @param buffer Buffer to store value in
 * @param offset Offset in buffer
 */
public static void encodeLong(long value, byte[] buffer, int offset)
{
   int pos = offset;
   buffer[pos++] = (byte)((value >>> 56) & 0xFF);
   buffer[pos++] = (byte)((value >>> 48) & 0xFF);
   buffer[pos++] = (byte)((value >>> 40) & 0xFF);
   buffer[pos++] = (byte)((value >>> 32) & 0xFF);
   buffer[pos++] = (byte)((value >>> 24) & 0xFF);
   buffer[pos++] = (byte)((value >>> 16) & 0xFF);
   buffer[pos++] = (byte)((value >>>  8) & 0xFF);
   buffer[pos++] = (byte)((value >>>  0) & 0xFF);
}

/**
 * Decodes a big-endian 64-bit integer stored in buffer into a long
 * value.
 *
 * @param buffer Buffer containing encoded value
 *
 * @return Decoded value
```

```
 */
public static long decodeLong(byte[] buffer)
{
   return decodeLong(buffer, 0);
}

/**
 * Decodes a big-endian 64-bit integer stored in buffer at specified
 * offset into a long value.
 *
 * @param buffer Buffer containing encoded value
 * @param offset Offset in buffer
 *
 * @return Decoded value
 */
public static long decodeLong(byte[] buffer, int offset)
{
   long value = 0;
   int pos = offset;
   value += ((long)buffer[pos++] & 0xff) << 56;
   value += ((long)buffer[pos++] & 0xff) << 48;
   value += ((long)buffer[pos++] & 0xff) << 40;
   value += ((long)buffer[pos++] & 0xff) << 32;
   value += ((long)buffer[pos++] & 0xff) << 24;
   value += ((long)buffer[pos++] & 0xff) << 16;
   value += ((long)buffer[pos++] & 0xff) <<  8;
   value += ((long)buffer[pos++] & 0xff) <<  0;

   return value;
}

/**
 * Encodes a short value as a big-endian 16-bit integer into a 2-byte
 * buffer and returns that buffer.
 *
 * @param value Value to encode
 *
 * @return Buffer with encoded value
 */
public static byte[] encodeShort(short value)
{
   byte[] buffer = new byte[2];
   encodeShort(value, buffer, 0);
   return buffer;
}

/**
 * Encodes a short value as a big-endian 16-bit integer into a buffer at
 * the specified offset.
 *
 * @param value  Value to encode
 * @param buffer Buffer to store value in
 * @param offset Offset in buffer
 */
public static void encodeShort(short value, byte[] buffer, int offset)
{
   int pos = offset;
```

```
   buffer[pos++] = (byte)((value >>>  8) & 0xFF);
   buffer[pos++] = (byte)((value >>>  0) & 0xFF);
}

/**
 * Decodes a big-endian 16-bit integer stored in buffer into a short
 * value.
 *
 * @param buffer Buffer containing encoded value
 *
 * @return Decoded value
 */
public static short decodeShort(byte[] buffer)
{
   return decodeShort(buffer, 0);
}

/**
 * Decodes a big-endian 16-bit integer stored in buffer at specified
 * offset into a short value.
 *
 * @param buffer Buffer containing encoded value
 * @param offset Offset in buffer
 *
 * @return Decoded value
 */
public static short decodeShort(byte[] buffer, int offset)
{
   short value = 0;
   int pos = offset;
   value += (buffer[pos++] & 0xff) <<  8;
   value += (buffer[pos++] & 0xff) <<  0;

   return value;
}

/**
 * Appends a null character to the string and returns a buffer
 * containing the encoded string. The purpose of the null
 * character is compatibility with federates written in C++.
 *
 * @param str String to encode
 *
 * @return Buffer with encoded string
 */
public static byte[] encodeString(String str)
{
   return (str + "\0").getBytes();
}

/**
 * Decodes a string from a buffer and strips the final null character.
 *
 * @param buffer Buffer with encoded string
 *
 * @return Decoded string
 */
```

```
   public static String decodeString(byte[] buffer)
   {
      return new String(buffer, 0, buffer.length - 1);
   }
}
```

B.2 NullFederateAmbassador.java

```
package hla.rti.jlc;

import hla.rti.*;

/**
 * Provides empty implementations for all methods in FederateAmbassador.
 */
public class NullFederateAmbassador implements FederateAmbassador
{
   //4.7
   public void synchronizationPointRegistrationFailed(
      String synchronizationPointLabel)
      throws
      FederateInternalError
   {
   }

   //4.7
   public void synchronizationPointRegistrationSucceeded(
      String synchronizationPointLabel)
      throws
      FederateInternalError
   {
   }

   //4.8
   public void announceSynchronizationPoint(
      String synchronizationPointLabel,
      byte[] userSuppliedTag)
      throws
      FederateInternalError
   {
   }

   //4.10
   public void federationSynchronized(
      String synchronizationPointLabel)
      throws
      FederateInternalError
   {
   }

   //4.12
   public void initiateFederateSave(
      String label)
      throws
      UnableToPerformSave,
      FederateInternalError
   {
```

```
}

// 4.15
public void federationSaved()
   throws
   FederateInternalError
{
}

// 4.15
public void federationNotSaved()
   throws
   FederateInternalError
{
}

// 4.17
public void requestFederationRestoreSucceeded(
   String label)
   throws
   FederateInternalError
{
}

// 4.17
public void requestFederationRestoreFailed(
   String label,
   String reason)
   throws
   FederateInternalError
{
}

// 4.18
public void federationRestoreBegun()
   throws
   FederateInternalError
{
}

// 4.19
public void initiateFederateRestore(
   String label,
   int federateHandle)
   throws
   SpecifiedSaveLabelDoesNotExist,
   CouldNotRestore,
   FederateInternalError
{
}

// 4.21
public void federationRestored()
   throws
   FederateInternalError
{
}
```

```
// 4.21
public void federationNotRestored()
   throws
   FederateInternalError
{
}

// 5.10
public void startRegistrationForObjectClass(
   int theClass)
   throws
   ObjectClassNotPublished,
   FederateInternalError
{
}

// 5.11
public void stopRegistrationForObjectClass(
   int theClass)
   throws
   ObjectClassNotPublished,
   FederateInternalError
{
}

// 5.12
public void turnInteractionsOn(
   int theHandle)
   throws
   InteractionClassNotPublished,
   FederateInternalError
{
}

// 5.13
public void turnInteractionsOff(
   int theHandle)
   throws
   InteractionClassNotPublished,
   FederateInternalError
{
}

// 6.3
public void discoverObjectInstance(
   int theObject,
   int theObjectClass,
   String objectName)
   throws
   CouldNotDiscover,
   ObjectClassNotKnown,
   FederateInternalError
{
}

// 6.5
```

```
public void reflectAttributeValues(
   int theObject,
   ReflectedAttributes theAttributes,
   byte[] userSuppliedTag)
   throws
   ObjectNotKnown,
   AttributeNotKnown,
   FederateOwnsAttributes,
   FederateInternalError
{
}

// 6.5
public void reflectAttributeValues(
   int theObject,
   ReflectedAttributes theAttributes,
   byte[] userSuppliedTag,
   LogicalTime theTime,
   EventRetractionHandle retractionHandle)
   throws
   ObjectNotKnown,
   AttributeNotKnown,
   FederateOwnsAttributes,
   InvalidFederationTime,
   FederateInternalError
{
}

// 6.7
public void receiveInteraction(
   int interactionClass,
   ReceivedInteraction theInteraction,
   byte[] userSuppliedTag)
   throws
   InteractionClassNotKnown,
   InteractionParameterNotKnown,
   FederateInternalError
{
}

// 6.7
public void receiveInteraction(
   int interactionClass,
   ReceivedInteraction theInteraction,
   byte[] userSuppliedTag,
   LogicalTime theTime,
   EventRetractionHandle eventRetractionHandle)
   throws
   InteractionClassNotKnown,
   InteractionParameterNotKnown,
   InvalidFederationTime,
   FederateInternalError
{
}

// 6.9
public void removeObjectInstance(
```

```
   int theObject,
   byte[] userSuppliedTag)
   throws
   ObjectNotKnown,
   FederateInternalError
{
}

// 6.9
public void removeObjectInstance(
   int theObject,
   byte[] userSuppliedTag,
   LogicalTime theTime,
   EventRetractionHandle retractionHandle)
   throws
   ObjectNotKnown,
   InvalidFederationTime,
   FederateInternalError
{
}

// 6.13
public void attributesInScope(
   int theObject,
   AttributeHandleSet theAttributes)
   throws
   ObjectNotKnown,
   AttributeNotKnown,
   FederateInternalError
{
}

// 6.14
public void attributesOutOfScope(
   int theObject,
   AttributeHandleSet theAttributes)
   throws
   ObjectNotKnown,
   AttributeNotKnown,
   FederateInternalError
{
}

// 6.16
public void provideAttributeValueUpdate(
   int theObject,
   AttributeHandleSet theAttributes)
   throws
   ObjectNotKnown,
   AttributeNotKnown,
   AttributeNotOwned,
   FederateInternalError
{
}

// 6.17
public void turnUpdatesOnForObjectInstance(
```

```
   int theObject,
   AttributeHandleSet theAttributes)
   throws
   ObjectNotKnown,
   AttributeNotOwned,
   FederateInternalError
{
}

// 6.18
public void turnUpdatesOffForObjectInstance(
   int theObject,
   AttributeHandleSet theAttributes)
   throws
   ObjectNotKnown,
   AttributeNotOwned,
   FederateInternalError
{
}

// 7.4
public void requestAttributeOwnershipAssumption(
   int theObject,
   AttributeHandleSet offeredAttributes,
   byte[] userSuppliedTag)
   throws
   ObjectNotKnown,
   AttributeNotKnown,
   AttributeAlreadyOwned,
   AttributeNotPublished,
   FederateInternalError
{
}

// 7.5
public void attributeOwnershipDivestitureNotification(
   int theObject,
   AttributeHandleSet releasedAttributes)
   throws
   ObjectNotKnown,
   AttributeNotKnown,
   AttributeNotOwned,
   AttributeDivestitureWasNotRequested,
   FederateInternalError
{
}

// 7.6
public void attributeOwnershipAcquisitionNotification(
   int theObject,
   AttributeHandleSet securedAttributes)
   throws
   ObjectNotKnown,
   AttributeNotKnown,
   AttributeAcquisitionWasNotRequested,
   AttributeAlreadyOwned,
   AttributeNotPublished,
```

```
   FederateInternalError
{
}

// 7.9
public void attributeOwnershipUnavailable(
   int theObject,
   AttributeHandleSet theAttributes)
   throws
   ObjectNotKnown,
   AttributeNotKnown,
   AttributeAlreadyOwned,
   AttributeAcquisitionWasNotRequested,
   FederateInternalError
{
}

// 7.10
public void requestAttributeOwnershipRelease(
   int theObject,
   AttributeHandleSet candidateAttributes,
   byte[] userSuppliedTag)
   throws
   ObjectNotKnown,
   AttributeNotKnown,
   AttributeNotOwned,
   FederateInternalError
{
}

// 7.14
public void confirmAttributeOwnershipAcquisitionCancellation(
   int theObject,
   AttributeHandleSet theAttributes)
   throws
   ObjectNotKnown,
   AttributeNotKnown,
   AttributeAlreadyOwned,
   AttributeAcquisitionWasNotCanceled,
   FederateInternalError
{
}

// 7.16
public void informAttributeOwnership(
   int theObject,
   int theAttribute,
   int theOwner)
   throws
   ObjectNotKnown,
   AttributeNotKnown,
   FederateInternalError
{
}

// 7.16
public void attributeIsNotOwned(
```

```
   int theObject,
   int theAttribute)
   throws
   ObjectNotKnown,
   AttributeNotKnown,
   FederateInternalError
{
}

// 7.16
public void attributeOwnedByRTI(
   int theObject,
   int theAttribute)
   throws
   ObjectNotKnown,
   AttributeNotKnown,
   FederateInternalError
{
}

// 8.3
public void timeRegulationEnabled(
   LogicalTime theFederateTime)
   throws
   InvalidFederationTime,
   EnableTimeRegulationWasNotPending,
   FederateInternalError
{
}

// 8.6
public void timeConstrainedEnabled(
   LogicalTime theFederateTime)
   throws
   InvalidFederationTime,
   EnableTimeConstrainedWasNotPending,
   FederateInternalError
{
}

// 8.13
public void timeAdvanceGrant(
   LogicalTime theTime)
   throws
   InvalidFederationTime,
   TimeAdvanceWasNotInProgress,
   FederateInternalError
{
}

// 8.22
public void requestRetraction(
   EventRetractionHandle theHandle)
   throws
   EventNotKnown,
   FederateInternalError
{
```

```
   }
}
```

B.3 RTIambassadorEx.java

```
package hla.rti.jlc;

import hla.rti.*;

public interface RTIambassadorEx extends RTIambassador
{
   public boolean tick( final double min, final double max )
     throws hla.rti.RTIinternalError,
              hla.rti.ConcurrentAccessAttempted;
}
```

B.4 RtiFactory.java

```
package hla.rti.jlc;

import hla.rti.*;

public interface RtiFactory
{
   RTIambassadorEx createRtiAmbassador()
     throws hla.rti.RTIinternalError;

   AttributeHandleSet createAttributeHandleSet();
   FederateHandleSet createFederateHandleSet();
   SuppliedAttributes createSuppliedAttributes();
   SuppliedParameters createSuppliedParameters();

   String RtiName();
   String RtiVersion();

   long getMinExtent();
   long getMaxExtent();
}
```

B.5 RtiFactoryFactory.java

```
package hla.rti.jlc;

import hla.rti.RTIinternalError;

import java.util.Properties;
import java.util.Map;
import java.util.HashMap;
import java.io.File;
import java.io.InputStream;
import java.io.FileInputStream;
import java.io.IOException;

public class RtiFactoryFactory
{
   public static RtiFactory getRtiFactory(String factoryClassName)
      throws RTIinternalError
```

```
   {
      try {
         Class cls = Class.forName(factoryClassName);
         return (RtiFactory)cls.newInstance();
      } catch (ClassNotFoundException e) {
         throw new RTIinternalError("Cannot find class " + factoryClassName);
      } catch (InstantiationException e) {
         throw new RTIinternalError("Cannot instantiate class " +
factoryClassName);
      } catch (IllegalAccessException e) {
         throw new RTIinternalError("Cannot access class " +
factoryClassName);
      }
   }

   public static RtiFactory getRtiFactory() throws RTIinternalError
   {
      String userHomeDir = System.getProperty("user.home");
      File propertiesFile = new File(userHomeDir, "RTI-list.properties");

      if (propertiesFile.exists()) {
         Properties properties = new Properties();
         try {
            InputStream is = new FileInputStream(propertiesFile);
            properties.load(is);
            is.close();
         } catch (IOException e) {
            throw new RTIinternalError("Error reading Link Compatibility
settings file");
         }

         String defaultRTI = properties.getProperty("Default");
         if (defaultRTI != null) {
            String factoryClassName = properties.getProperty(defaultRTI +
".factory");
            if (factoryClassName == null) {
               throw new RTIinternalError("Cannot find factory class setting
for default RTI");
            }

            return getRtiFactory(factoryClassName);
         }
      }

      // Provide a reasonable default if no setting found
      return getRtiFactory("com.rtibusters.rti.BustersRtiFactory");
   }

   public static Map getAvailableRtis() throws RTIinternalError
   {
      String userHomeDir = System.getProperty("user.home");
      File propertiesFile = new File(userHomeDir, "RTI-list.properties");

      if (!propertiesFile.exists()) {
         throw new RTIinternalError("Cannot find file " + propertiesFile);
      }
```

```
      Properties properties = new Properties();
      try {
         InputStream is = new FileInputStream(propertiesFile);
         properties.load(is);
         is.close();
      } catch (IOException e) {
         throw new RTIinternalError("Error reading Link Compatibility
settings file");
      }

      Map map = new HashMap();
      int index = 1;
      while (true) {
         String rtiName = properties.getProperty(index + ".name");
         String rtiFactory = properties.getProperty(index + ".factory");
         if (rtiName == null || rtiFactory == null) {
            break;
         }
         map.put(rtiName, rtiFactory);
         index++;
      }

      return map;
   }
}
```

B.6 ArrayIndexOutOfBounds.java

```
package hla.rti;

public final class ArrayIndexOutOfBounds extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public ArrayIndexOutOfBounds(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public ArrayIndexOutOfBounds(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.7 AsynchronousDeliveryAlreadyDisabled.java

```
package hla.rti;

/**
 * Public exception class AsynchronousDeliveryAlreadyDisabled
*/
public final class AsynchronousDeliveryAlreadyDisabled extends RTIexception {

  /**
```

```
   * @param reason    String to be carried with exception
   */
  public AsynchronousDeliveryAlreadyDisabled(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public AsynchronousDeliveryAlreadyDisabled(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.8 AsynchronousDeliveryAlreadyEnabled.java

```
package hla.rti;

/**
 * Public exception class AsynchronousDeliveryAlreadyEnabled
*/
public final class AsynchronousDeliveryAlreadyEnabled extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public AsynchronousDeliveryAlreadyEnabled(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public AsynchronousDeliveryAlreadyEnabled(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.9 AttributeAcquisitionWasNotCanceled.java

```
package hla.rti;

/**
 * Public exception class AttributeAcquisitionWasNotCanceled
*/

public final class AttributeAcquisitionWasNotCanceled extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public AttributeAcquisitionWasNotCanceled(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
```

```
   */
  public AttributeAcquisitionWasNotCanceled(String reason, int serial) {

      super(reason, serial);
  }
}
```

B.10 AttributeAcquisitionWasNotRequested.java

```
package hla.rti;

/**
 * Public exception class AttributeAcquisitionWasNotRequested
*/

public final class AttributeAcquisitionWasNotRequested extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public AttributeAcquisitionWasNotRequested(String reason) {

      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public AttributeAcquisitionWasNotRequested(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.11 AttributeAlreadyBeingAcquired.java

```
package hla.rti;

/**
 * Public exception class AttributeAlreadyBeingAcquired
*/

public final class AttributeAlreadyBeingAcquired extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public AttributeAlreadyBeingAcquired(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public AttributeAlreadyBeingAcquired(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.12 AttributeAlreadyBeingDivested.java

```
package hla.rti;

/**
 * Public exception class AttributeAlreadyBeingDivested
*/

public final class AttributeAlreadyBeingDivested extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public AttributeAlreadyBeingDivested(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public AttributeAlreadyBeingDivested(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.13 AttributeAlreadyOwned.java

```
package hla.rti;

/**
 * Public exception class AttributeAlreadyOwned
*/

public final class AttributeAlreadyOwned extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public AttributeAlreadyOwned(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public AttributeAlreadyOwned(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.14 AttributeDivestitureWasNotRequested.java

```
package hla.rti;

/**
 * Public exception class AttributeDivestitureWasNotRequested
```

```
*/

public final class AttributeDivestitureWasNotRequested extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public AttributeDivestitureWasNotRequested(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public AttributeDivestitureWasNotRequested(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.15 AttributeHandleSet.java

```
package hla.rti;

/**
 *
 *
 */
public interface AttributeHandleSet {


/**
 * Add the handle to the set. Won't squawk if handle already member.
 *
 * @param handle int
 * @exception hla.rti.AttributeNotDefined if handle out of range
 */
public void add ( int handle) throws AttributeNotDefined;
/**
 * Classic clone
 *
 * @return java.lang.Object
 */
public Object clone();
/**
 * Empties set of its members.
 *
 */
public void empty ( );
/**
 * Classic equals.
 *
 * @return boolean: true if set of same type and same members.
 * @param obj java.lang.Object
 */
public boolean equals(Object obj);
public HandleIterator handles ( );
/**
```

```
 * Classic hashCode
 *
 * @return int: hash code
 */
public int hashCode();
/**
 *
 * @return boolean: true if set empty.
 */
public boolean isEmpty();
/**
 *
 * @return boolean: true if handle is a member
 * @param handle int: an attribute handle
 * @exception hla.rti.AttributeNotDefined if handle out of range
 */
public boolean isMember(int handle) throws AttributeNotDefined;
/**
 * Remove the handle from the set. Won't squawk if handle not a member.
 *
 * @param handle int
 * @exception hla.rti.AttributeNotDefined if handle out of range
 */
public void remove (int handle ) throws AttributeNotDefined;
/**
 *
 * @return int: number of members
 */
public int size();
/**
 *
 * @return java.lang.String
 */
public String toString();
}
```

B.16 AttributeHandleSetFactory.java

```
/* File AttributeHandleSetFactory.java */
package hla.rti;


public interface AttributeHandleSetFactory {
   /**
   * This method was created by a SmartGuide.
   * @return hla.rti.AttributeHandleSet newly created
   */
   public AttributeHandleSet create();
}
```

B.17 AttributeNotDefined.java

```
package hla.rti;

/**
 * Public exception class AttributeNotDefined
*/
```

```
public final class AttributeNotDefined extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public AttributeNotDefined(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public AttributeNotDefined(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.18 AttributeNotKnown.java

```
package hla.rti;

/**
 * Public exception class AttributeNotKnown
*/

public final class AttributeNotKnown extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public AttributeNotKnown(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public AttributeNotKnown(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.19 AttributeNotOwned.java

```
package hla.rti;

/**
 * Public exception class AttributeNotOwned
*/

public final class AttributeNotOwned extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public AttributeNotOwned(String reason) {
      super(reason, 0);
```

```
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public AttributeNotOwned(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.20 AttributeNotPublished.java

```
package hla.rti;

/**
 * Public exception class AttributeNotPublished
*/

public final class AttributeNotPublished extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public AttributeNotPublished(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public AttributeNotPublished(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.21 ConcurrentAccessAttempted.java

```
package hla.rti;

public final class ConcurrentAccessAttempted extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public ConcurrentAccessAttempted(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public ConcurrentAccessAttempted(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.22 CouldNotDecode.java

```
package hla.rti;

public final class CouldNotDecode extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public CouldNotDecode(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public CouldNotDecode(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.23 CouldNotDiscover.java

```
package hla.rti;

/**
 * Public exception class CouldNotDiscover
*/

public final class CouldNotDiscover extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public CouldNotDiscover(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public CouldNotDiscover(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.24 CouldNotOpenFED.java

```
package hla.rti;

/**
 * Public exception class CouldNotOpenFED
*/
public final class CouldNotOpenFED extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
```

```
  public CouldNotOpenFED(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public CouldNotOpenFED(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.25 CouldNotRestore.java

```
package hla.rti;

/**
 * Public exception class CouldNotRestore
*/

public final class CouldNotRestore extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public CouldNotRestore(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public CouldNotRestore(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.26 DeletePrivilegeNotHeld.java

```
package hla.rti;

/**
 * Public exception class DeletePrivilegeNotHeld
*/

public final class DeletePrivilegeNotHeld extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public DeletePrivilegeNotHeld(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
```

```
  public DeletePrivilegeNotHeld(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.27 DimensionNotDefined.java

```
package hla.rti;

/**
 * Public exception class DimensionNotDefined
*/

public final class DimensionNotDefined extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public DimensionNotDefined(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public DimensionNotDefined(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.28 EnableTimeConstrainedPending.java

```
package hla.rti;

/**
 * Public exception class EnableTimeConstrainedPending
*/

public final class EnableTimeConstrainedPending extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public EnableTimeConstrainedPending(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public EnableTimeConstrainedPending(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.29 EnableTimeConstrainedWasNotPending.java

```
package hla.rti;

/**
 * Public exception class EnableTimeConstrainedWasNotPending
*/

public final class EnableTimeConstrainedWasNotPending extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public EnableTimeConstrainedWasNotPending(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public EnableTimeConstrainedWasNotPending(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.30 EnableTimeRegulationPending.java

```
package hla.rti;

/**
 * Public exception class EnableTimeRegulationPending
*/

public final class EnableTimeRegulationPending extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public EnableTimeRegulationPending(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public EnableTimeRegulationPending(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.31 EnableTimeRegulationWasNotPending.java

```
package hla.rti;

/**
 * Public exception class EnableTimeRegulationWasNotPending
*/

public final class EnableTimeRegulationWasNotPending extends RTIexception {
```

```
  /**
   * @param reason    String to be carried with exception
   */
  public EnableTimeRegulationWasNotPending(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public EnableTimeRegulationWasNotPending(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.32 ErrorReadingFED.java

```
package hla.rti;

/**
 * Public exception class ErrorReadingFED
*/
public final class ErrorReadingFED extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public ErrorReadingFED(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public ErrorReadingFED(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.33 EventNotKnown.java

```
package hla.rti;

/**
 * Public exception class EventNotKnown
*/

public final class EventNotKnown extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public EventNotKnown(String reason) {
      super(reason, 0);
  }
```

```
  /**
   * @param serial    serial number also printed with the exception
   */
  public EventNotKnown(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.34 EventRetractionHandle.java

```
package hla.rti;

/**
 * The user can do nothing with these but employ them as keys.
 * Implementers should provide equals, hashCode and toString
 * rather than settling for the defaults.
 *
 */
public interface EventRetractionHandle {

}
```

B.35 FederateAlreadyExecutionMember.java

```
package hla.rti;

/**
 * Public exception class FederateAlreadyExecutionMember
*/
public final class FederateAlreadyExecutionMember extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public FederateAlreadyExecutionMember(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public FederateAlreadyExecutionMember(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.36 FederateAmbassador.java

```
package hla.rti;


/**
* Federate must implement this interface.
*/

public interface FederateAmbassador {
```

```
////////////////////////////////////
// Federation Management Services //
////////////////////////////////////

  //4.7
  public void synchronizationPointRegistrationFailed(
    String synchronizationPointLabel)
  throws
    FederateInternalError;

  //4.7
  public void synchronizationPointRegistrationSucceeded(
    String synchronizationPointLabel)
  throws
    FederateInternalError;

  //4.8
  public void announceSynchronizationPoint(
    String synchronizationPointLabel,
    byte[] userSuppliedTag)
  throws
    FederateInternalError;

  //4.10
  public void federationSynchronized(
    String synchronizationPointLabel)
  throws
    FederateInternalError;

  //4.12
  public void initiateFederateSave(
    String label)
  throws
    UnableToPerformSave,
    FederateInternalError;

  // 4.15
  public void federationSaved ()
  throws
    FederateInternalError;

  // 4.15
  public void federationNotSaved ()
  throws
    FederateInternalError;

  // 4.17
  public void requestFederationRestoreSucceeded (
    String label)
  throws
    FederateInternalError;

  // 4.17
  public void requestFederationRestoreFailed (
    String label,
    String reason)
  throws
```

```
    FederateInternalError;

  // 4.18
  public void federationRestoreBegun ()
  throws
    FederateInternalError;

  // 4.19
  public void initiateFederateRestore (
    String label,
    int    federateHandle)
  throws
    SpecifiedSaveLabelDoesNotExist,
    CouldNotRestore,
    FederateInternalError;

  // 4.21
  public void federationRestored ()
  throws
    FederateInternalError;

  // 4.21
  public void federationNotRestored ()
  throws
    FederateInternalError;

////////////////////////////////////
// Declaration Management Services //
////////////////////////////////////

  // 5.10
  public void startRegistrationForObjectClass (
    int theClass)
  throws
    ObjectClassNotPublished,
    FederateInternalError;

  // 5.11
  public void stopRegistrationForObjectClass (
    int theClass)
  throws
    ObjectClassNotPublished,
    FederateInternalError;

  // 5.12
  public void turnInteractionsOn (
    int theHandle)
  throws
    InteractionClassNotPublished,
    FederateInternalError;

  // 5.13
  public void turnInteractionsOff (
    int theHandle)
  throws
    InteractionClassNotPublished,
    FederateInternalError;
```

```
////////////////////////////////
// Object Management Services //
////////////////////////////////

  // 6.3
  public void discoverObjectInstance (
    int    theObject,
    int    theObjectClass,
    String objectName)
  throws
    CouldNotDiscover,
    ObjectClassNotKnown,
    FederateInternalError;

  // 6.5
  public void reflectAttributeValues (
    int                 theObject,
    ReflectedAttributes theAttributes,
    byte[]              userSuppliedTag)
  throws
    ObjectNotKnown,
    AttributeNotKnown,
    FederateOwnsAttributes,
    FederateInternalError;

  // 6.5
  public void reflectAttributeValues (
    int                 theObject,
    ReflectedAttributes theAttributes,
    byte[]              userSuppliedTag,
    LogicalTime         theTime,
    EventRetractionHandle retractionHandle)
  throws
    ObjectNotKnown,
    AttributeNotKnown,
    FederateOwnsAttributes,
    InvalidFederationTime,
    FederateInternalError;

  // 6.7
  public void receiveInteraction (
    int                 interactionClass,
    ReceivedInteraction theInteraction,
    byte[]              userSuppliedTag)
  throws
    InteractionClassNotKnown,
    InteractionParameterNotKnown,
    FederateInternalError;

  // 6.7
  public void receiveInteraction (
    int                 interactionClass,
    ReceivedInteraction theInteraction,
    byte[]              userSuppliedTag,
    LogicalTime         theTime,
    EventRetractionHandle eventRetractionHandle)
```

```
throws
  InteractionClassNotKnown,
  InteractionParameterNotKnown,
  InvalidFederationTime,
  FederateInternalError;

// 6.9
public void removeObjectInstance (
  int    theObject,
  byte[] userSuppliedTag)
throws
  ObjectNotKnown,
  FederateInternalError;

// 6.9
public void removeObjectInstance (
  int            theObject,
  byte[]         userSuppliedTag,
  LogicalTime    theTime,
  EventRetractionHandle retractionHandle)
throws
  ObjectNotKnown,
  InvalidFederationTime,
  FederateInternalError;

// 6.13
public void attributesInScope (
  int                theObject,
  AttributeHandleSet theAttributes)
throws
  ObjectNotKnown,
  AttributeNotKnown,
  FederateInternalError;

// 6.14
public void attributesOutOfScope (
  int                theObject,
  AttributeHandleSet theAttributes)
throws
  ObjectNotKnown,
  AttributeNotKnown,
  FederateInternalError;

// 6.16
public void provideAttributeValueUpdate (
  int                theObject,
  AttributeHandleSet theAttributes)
throws
  ObjectNotKnown,
  AttributeNotKnown,
  AttributeNotOwned,
  FederateInternalError;

// 6.17
public void turnUpdatesOnForObjectInstance (
  int                theObject,
  AttributeHandleSet theAttributes)
```

```
  throws
    ObjectNotKnown,
    AttributeNotOwned,
    FederateInternalError;

  // 6.18
  public void turnUpdatesOffForObjectInstance (
    int                theObject,
    AttributeHandleSet theAttributes)
  throws
    ObjectNotKnown,
    AttributeNotOwned,
    FederateInternalError;

///////////////////////////////////
// Ownership Management Services //
///////////////////////////////////

  // 7.4
  public void requestAttributeOwnershipAssumption (
    int                theObject,
    AttributeHandleSet offeredAttributes,
    byte[]             userSuppliedTag)
  throws
    ObjectNotKnown,
    AttributeNotKnown,
    AttributeAlreadyOwned,
    AttributeNotPublished,
    FederateInternalError;

  // 7.5
  public void attributeOwnershipDivestitureNotification (
    int                theObject,
    AttributeHandleSet releasedAttributes)
  throws
    ObjectNotKnown,
    AttributeNotKnown,
    AttributeNotOwned,
    AttributeDivestitureWasNotRequested,
    FederateInternalError;

  // 7.6
  public void attributeOwnershipAcquisitionNotification (
    int                theObject,
    AttributeHandleSet securedAttributes)
  throws
    ObjectNotKnown,
    AttributeNotKnown,
    AttributeAcquisitionWasNotRequested,
    AttributeAlreadyOwned,
    AttributeNotPublished,
    FederateInternalError;

  // 7.9
  public void attributeOwnershipUnavailable (
    int                theObject,
    AttributeHandleSet theAttributes)
```

```
throws
  ObjectNotKnown,
  AttributeNotKnown,
  AttributeAlreadyOwned,
  AttributeAcquisitionWasNotRequested,
  FederateInternalError;

// 7.10
public void requestAttributeOwnershipRelease (
  int                theObject,
  AttributeHandleSet candidateAttributes,
  byte[]             userSuppliedTag)
throws
  ObjectNotKnown,
  AttributeNotKnown,
  AttributeNotOwned,
  FederateInternalError;

// 7.14
public void confirmAttributeOwnershipAcquisitionCancellation (
     int                theObject,
     AttributeHandleSet theAttributes)
throws
  ObjectNotKnown,
  AttributeNotKnown,
  AttributeAlreadyOwned,
  AttributeAcquisitionWasNotCanceled,
  FederateInternalError;

// 7.16
public void informAttributeOwnership (
  int theObject,
  int theAttribute,
  int theOwner)
throws
  ObjectNotKnown,
  AttributeNotKnown,
  FederateInternalError;

// 7.16
public void attributeIsNotOwned (
  int theObject,
  int theAttribute)
throws
  ObjectNotKnown,
  AttributeNotKnown,
  FederateInternalError;

// 7.16
public void attributeOwnedByRTI (
  int theObject,
  int theAttribute)
throws
  ObjectNotKnown,
  AttributeNotKnown,
  FederateInternalError;
```

```
//////////////////////////////
// Time Management Services //
//////////////////////////////

  // 8.3
  public void timeRegulationEnabled (
    LogicalTime    theFederateTime)
  throws
    InvalidFederationTime,
    EnableTimeRegulationWasNotPending,
    FederateInternalError;

  // 8.6
  public void timeConstrainedEnabled (
    LogicalTime    theFederateTime)
  throws
    InvalidFederationTime,
    EnableTimeConstrainedWasNotPending,
    FederateInternalError;

  // 8.13
  public void timeAdvanceGrant (
    LogicalTime    theTime)
  throws
    InvalidFederationTime,
    TimeAdvanceWasNotInProgress,
    FederateInternalError;

  // 8.22
  public void requestRetraction (
    EventRetractionHandle theHandle)
  throws
    EventNotKnown,
    FederateInternalError;
}
```

B.37 FederateHandleSet.java

```
package hla.rti;

/**
 * Interface for a set of federate handles.
 *
 */
public interface FederateHandleSet {

/**
 * Add the handle to the set. Won't squawk if handle already member.
 * @param handle int
 */
public void add ( int handle);
/**
 * Classic clone
 *
 * @return java.lang.Object
 */
public Object clone();
```

```
/**
 * Empties set of its members.
 *
 */
public void empty ( );
/**
 * Classic equals.
 *
 * @return boolean: true if set of same type and same members.
 * @param obj java.lang.Object
 */
public boolean equals(Object obj);
public HandleIterator handles ( );
/**
 * Classic hashCode
 *
 * @return int: hash code
 */
public int hashCode();
/**
 *
 * @return boolean: true if set empty.
 */
public boolean isEmpty();
/**
 *
 * @return boolean: true if handle is a meber
 * @param handle int: an attribute handle
 */
public boolean isMember(int handle);
/**
 * Remove the handle from the set. Won't squawk if handle not a member.
 *
 * @param handle int
 */
public void remove (int handle );
/**
 *
 * @return int: number of members
 */
public int size();
/**
 *
 * @return java.lang.String
 */
public String toString();
}
```

B.38 FederateHandleSetFactory.java

```
package hla.rti;

/**
 *
 *
 */
public interface FederateHandleSetFactory {
```

```
/**
 *
 * @return hla.rti.FederateHandleSet newly created
 * @exception hla.rti.FederationExecutionDoesNotExist if called before FED
read
 */
public FederateHandleSet create()
throws FederationExecutionDoesNotExist;
}
```

B.39 FederateInternalError.java

```
package hla.rti;

/**
 * Public exception class FederateInternalError
*/
public final class FederateInternalError extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public FederateInternalError(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public FederateInternalError(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.40 FederateLoggingServiceCalls.java

```
package hla.rti;

/**
 * Public exception class FederateLoggingServiceCalls
*/

public final class FederateLoggingServiceCalls extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public FederateLoggingServiceCalls(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public FederateLoggingServiceCalls(String reason, int serial) {
      super(reason, serial);
```

```
  }
}
```

B.41 FederateNotExecutionMember.java

```
package hla.rti;

/**
 * Public exception class FederateNotExecutionMember
*/
public final class FederateNotExecutionMember extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public FederateNotExecutionMember(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public FederateNotExecutionMember(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.42 FederateNotSubscribed.java

```
package hla.rti;


public final class FederateNotSubscribed extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public FederateNotSubscribed(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public FederateNotSubscribed(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.43 FederateOwnsAttributes.java

```
package hla.rti;

/**
 * Public exception class FederateOwnsAttributes
*/
public final class FederateOwnsAttributes extends RTIexception {
```

```
  /**
   * @param reason    String to be carried with exception
   */
  public FederateOwnsAttributes(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public FederateOwnsAttributes(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.44 FederatesCurrentlyJoined.java

```
package hla.rti;

/**
 * Public exception class FederatesCurrentlyJoined
*/
public final class FederatesCurrentlyJoined extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public FederatesCurrentlyJoined(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public FederatesCurrentlyJoined(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.45 FederateWasNotAskedToReleaseAttribute.java

```
package hla.rti;

/**
 * Public exception class FederateWasNotAskedToReleaseAttribute
*/

public final class FederateWasNotAskedToReleaseAttribute extends RTIexception
{

  /**
   * @param reason    String to be carried with exception
   */
  public FederateWasNotAskedToReleaseAttribute(String reason) {
      super(reason, 0);
  }
```

```
  /**
   * @param serial    serial number also printed with the exception
   */
  public FederateWasNotAskedToReleaseAttribute(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.46 FederationExecutionAlreadyExists.java

```
package hla.rti;

/**
 * Public exception class FederationExecutionAlreadyExists
*/
public final class FederationExecutionAlreadyExists extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public FederationExecutionAlreadyExists(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public FederationExecutionAlreadyExists(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.47 FederationExecutionDoesNotExist.java

```
package hla.rti;

/**
 * Public exception class FederationExecutionDoesNotExist
*/
public final class FederationExecutionDoesNotExist extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public FederationExecutionDoesNotExist(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public FederationExecutionDoesNotExist(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.48 FederationTimeAlreadyPassed.java

```
package hla.rti;

/**
 * Public exception class FederationTimeAlreadyPassed
*/

public final class FederationTimeAlreadyPassed extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public FederationTimeAlreadyPassed(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public FederationTimeAlreadyPassed(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.49 HandleIterator.java

```
package hla.rti;

/**
 * This iterator interface is intended to be used with HandleSets.
 * It differs from the usual Java Enumerator. The typical idiom
 * is:
 *   for (HandleIterator i = handleSet.handles(), int h = i.first();
              i.isValid();
              h = i.next(); {
            ...
        }
 * The handle value h will iterate through all the values in the set.
 * This interface was generated by a SmartGuide.
 *
 */
public interface HandleIterator {

/**
 * Call this to get the first valid handle. Resets the iterator.
 * @return int: first valid handle in set, or -1
 */
public int first();
/**
 * Should be checked before using return from first() or next()
 * @return boolean: true if currently reported handle is valid.
 */
public boolean isValid();
/**
 * @return int: next valid handle in set, or -1
 */
```

```
public int next();
}
```

B.50 IllegalTimeArithmetic.java

```
package hla.rti;

/**
 * Public exception class InvalidFederationTime
*/

public final class IllegalTimeArithmetic extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public IllegalTimeArithmetic(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public IllegalTimeArithmetic(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.51 InteractionClassNotDefined.java

```
package hla.rti;

/**
 * Public exception class InteractionClassNotDefined
*/

public final class InteractionClassNotDefined extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public InteractionClassNotDefined(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public InteractionClassNotDefined(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.52 InteractionClassNotKnown.java

```
package hla.rti;
```

```
/**
 * Public exception class InteractionClassNotKnown
*/

public final class InteractionClassNotKnown extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public InteractionClassNotKnown(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public InteractionClassNotKnown(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.53 InteractionClassNotPublished.java

```
package hla.rti;

/**
 * Public exception class InteractionClassNotPublished
*/

public final class InteractionClassNotPublished extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public InteractionClassNotPublished(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public InteractionClassNotPublished(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.54 InteractionClassNotSubscribed.java

```
package hla.rti;

/**
 * Public exception class InteractionClassNotSubscribed
*/

public final class InteractionClassNotSubscribed extends RTIexception {

  /**
```

```
   * @param reason    String to be carried with exception
   */
  public InteractionClassNotSubscribed(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public InteractionClassNotSubscribed(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.55 InteractionParameterNotDefined.java

```
package hla.rti;

/**
 * Public exception class InteractionParameterNotDefined
*/

public final class InteractionParameterNotDefined extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public InteractionParameterNotDefined(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public InteractionParameterNotDefined(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.56 InteractionParameterNotKnown.java

```
package hla.rti;

/**
 * Public exception class InteractionParameterNotKnown
*/

public final class InteractionParameterNotKnown extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public InteractionParameterNotKnown(String reason) {
      super(reason, 0);
  }

  /**
```

```
   * @param serial    serial number also printed with the exception
   */
  public InteractionParameterNotKnown(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.57 InvalidExtents.java

```
package hla.rti;

/**
 * Public exception class InvalidExtents
*/

public final class InvalidExtents extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public InvalidExtents(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public InvalidExtents(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.58 InvalidFederationTime.java

```
package hla.rti;

/**
 * Public exception class InvalidFederationTime
*/

public final class InvalidFederationTime extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public InvalidFederationTime(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public InvalidFederationTime(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.59 InvalidLookahead.java

```
package hla.rti;

/**
 * Public exception class InvalidLookahead
*/

public final class InvalidLookahead extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public InvalidLookahead(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public InvalidLookahead(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.60 InvalidOrderingHandle.java

```
package hla.rti;

/**
 * Public exception class InvalidOrderingHandle
*/

public final class InvalidOrderingHandle extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public InvalidOrderingHandle(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public InvalidOrderingHandle(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.61 InvalidRegionContext.java

```
package hla.rti;

/**
 * Public exception class InvalidRegionContext
*/
```

```
public final class InvalidRegionContext extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public InvalidRegionContext(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public InvalidRegionContext(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.62 InvalidResignAction.java

```
package hla.rti;

/**
 * Public exception class InvalidResignAction
*/
public final class InvalidResignAction extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public InvalidResignAction(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public InvalidResignAction(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.63 InvalidRetractionHandle.java

```
package hla.rti;

/**
 * Public exception class InvalidRetractionHandle
*/

public final class InvalidRetractionHandle extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public InvalidRetractionHandle(String reason) {
      super(reason, 0);
```

```
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public InvalidRetractionHandle(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.64 InvalidTransportationHandle.java

```
package hla.rti;

/**
 * Public exception class InvalidTransportationHandle
*/

public final class InvalidTransportationHandle extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public InvalidTransportationHandle(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public InvalidTransportationHandle(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.65 LogicalTime.java

```
package hla.rti;

public interface LogicalTime {

      public void decreaseBy(LogicalTimeInterval subtrahend)
      throws IllegalTimeArithmetic;
      public void encode(byte[] buffer, int offset);
      public int encodedLength();
      public void increaseBy(LogicalTimeInterval addend)
      throws IllegalTimeArithmetic;
      public boolean isEqualTo(LogicalTime value);
      public boolean isFinal();
      public boolean isGreaterThan(LogicalTime value);
      public boolean isGreaterThanOrEqualTo(LogicalTime value);
      public boolean isInitial();
      public boolean isLessThan(LogicalTime value);
      public boolean isLessThanOrEqualTo(LogicalTime value);
      public void setFinal();
      public void setInitial();
      public void setTo(LogicalTime value);
```

```
      public LogicalTimeInterval subtract(LogicalTime subtrahend);
}
```

B.66 LogicalTimeFactory.java

```
package hla.rti;

public interface LogicalTimeFactory {

      public LogicalTime decode(byte[] buffer, int offset)
      throws CouldNotDecode;
      public LogicalTime makeInitial();
}
```

B.67 LogicalTimeInterval.java

```
package hla.rti;

public interface LogicalTimeInterval {

      public void encode(byte[] buffer, int offset);
      public int encodedLength();
      public boolean isEpsilon();
      public boolean isEqualTo(LogicalTimeInterval value);
      public boolean isGreaterThan(LogicalTimeInterval value);
      public boolean isGreaterThanOrEqualTo(LogicalTimeInterval value);
      public boolean isLessThan(LogicalTimeInterval value);
      public boolean isLessThanOrEqualTo(LogicalTimeInterval value);
      public boolean isZero();
      public void setEpsilon();
      public void setTo(LogicalTimeInterval value);
      public void setZero();
}
```

B.68 LogicalTimeIntervalFactory.java

```
package hla.rti;

public interface LogicalTimeIntervalFactory {

      public LogicalTimeInterval decode(byte[] buffer, int offset)
      throws CouldNotDecode;
      public LogicalTimeInterval makeZero();
}
```

B.69 MobileFederateServices.java

```
package hla.rti;

/**
 * Conveys the interfaces for all services that a federate
 * must supply and which may not execute in the federate's
 * space.
 *
 */
public final class MobileFederateServices {
      public hla.rti.LogicalTimeFactory _timeFactory;
```

```
      public hla.rti.LogicalTimeIntervalFactory _intervalFactory;

/**
 * This method was created by a SmartGuide.
 * @param timeFactory hla.rti.LogicalTimeFactory
 * @param intervalFactory hla.rti.LogicalTimeIntervalFactory
 */
public MobileFederateServices (LogicalTimeFactory timeFactory,
LogicalTimeIntervalFactory intervalFactory) {
      _timeFactory = timeFactory;
      _intervalFactory = intervalFactory;
}
}
```

B.70 NameNotFound.java

```
package hla.rti;

/**
 * Public exception class NameNotFound
*/

public final class NameNotFound extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public NameNotFound(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public NameNotFound(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.71 ObjectAlreadyRegistered.java

```
package hla.rti;

/**
 * Public exception class ObjectAlreadyRegistered
*/

public final class ObjectAlreadyRegistered extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public ObjectAlreadyRegistered(String reason) {
      super(reason, 0);
  }

  /**
```

```
   * @param serial    serial number also printed with the exception
   */
  public ObjectAlreadyRegistered(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.72 ObjectClassNotDefined.java

```
package hla.rti;

/**
 * Public exception class ObjectClassNotDefined
*/

public final class ObjectClassNotDefined extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public ObjectClassNotDefined(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public ObjectClassNotDefined(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.73 ObjectClassNotKnown.java

```
package hla.rti;

/**
 * Public exception class ObjectClassNotKnown
*/

public final class ObjectClassNotKnown extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public ObjectClassNotKnown(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public ObjectClassNotKnown(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.74 ObjectClassNotPublished.java

```
package hla.rti;

/**
 * Public exception class ObjectClassNotPublished
*/

public final class ObjectClassNotPublished extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public ObjectClassNotPublished(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public ObjectClassNotPublished(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.75 ObjectClassNotSubscribed.java

```
package hla.rti;

/**
 * Public exception class ObjectClassNotSubscribed
*/

public final class ObjectClassNotSubscribed extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public ObjectClassNotSubscribed(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public ObjectClassNotSubscribed(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.76 ObjectNotKnown.java

```
package hla.rti;

/**
 * Public exception class ObjectNotKnown
*/
```

```
public final class ObjectNotKnown extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public ObjectNotKnown(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public ObjectNotKnown(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.77 OwnershipAcquisitionPending.java

```
package hla.rti;

public final class OwnershipAcquisitionPending extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public OwnershipAcquisitionPending(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public OwnershipAcquisitionPending(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.78 ReceivedInteraction.java

```
package hla.rti;

/**
 * This packages the information supplied to the federate for
 * receiveInteraction. The parameters are conceptually an array
 * with an initial capacity and the ability to grow.
 * You enumerate by stepping index from 0 to size()-1.
 *
 */
public interface ReceivedInteraction {

/**
 * Return order type
 * @return int order type
 */
public int getOrderType ();
```

```
/**
 * Return parameter handle at index position.
 * @return int parameter handle
 * @param index int
 * @exception hla.rti.ArrayIndexOutOfBounds
 */
public int getParameterHandle ( int index) throws ArrayIndexOutOfBounds;
/**
 * Return Region out of which interaction was received.
 */
public Region getRegion ();
/**
 * Return transport type
 * @return int transport type
 */
public int getTransportType ();
/**
 * Return copy of value at index position.
 * @return byte[] copy (clone) of value
 * @param index int
 * @exception hla.rti.ArrayIndexOutOfBounds
 */
public byte[] getValue ( int index) throws ArrayIndexOutOfBounds;
/**
 * Return length of value at index position.
 * @return int value length
 * @param index int
 * @exception hla.rti.ArrayIndexOutOfBounds
 */
public int getValueLength ( int index) throws ArrayIndexOutOfBounds;
/**
 * Get the reference of the value at position index (not a clone)
 * @return byte[] the reference
 * @param index int
 * @exception hla.rti.ArrayIndexOutOfBounds
 */
public byte[] getValueReference ( int index) throws ArrayIndexOutOfBounds;
/**
 * @return int Number of parameter handle-value pairs
 */
public int size ( );
}
```

B.79 ReflectedAttributes.java

```
package hla.rti;

/**
 * This packages the attributes supplied to the federate for
 * reflectAttributeValues. This is conceptually an array
 * with an initial capacity and the ability to grow.
 * You enumerate by stepping index from 0 to size()-1.
 *
 */
public interface ReflectedAttributes {

/**
```

```
 * Return attribute handle at index position.
 * @return int attribute handle
 * @param index int
 * @exception hla.rti.ArrayIndexOutOfBounds
 */
public int getAttributeHandle ( int index) throws ArrayIndexOutOfBounds;
/**
 * Return order handle at index position.
 * @return int order type
 * @param index int
 * @exception hla.rti.ArrayIndexOutOfBounds
 */
public int getOrderType ( int index) throws ArrayIndexOutOfBounds;
/**
 * Return Region handle at index position.
 * @return int region handle
 * @param index int
 * @exception hla.rti.ArrayIndexOutOfBounds
 */
public Region getRegion ( int index) throws ArrayIndexOutOfBounds;
/**
 * Return transport handle at index position.
 * @return int transport type
 * @param index int
 * @exception hla.rti.ArrayIndexOutOfBounds
 */
public int getTransportType ( int index) throws ArrayIndexOutOfBounds;
/**
 * Return copy of value at index position.
 * @return byte[] copy (clone) of value
 * @param index int
 * @exception hla.rti.ArrayIndexOutOfBounds
 */
public byte[] getValue ( int index) throws ArrayIndexOutOfBounds;
/**
 * Return length of value at index position.
 * @return int value length
 * @param index int
 * @exception hla.rti.ArrayIndexOutOfBounds
 */
public int getValueLength ( int index) throws ArrayIndexOutOfBounds;
/**
 * Get the reference of the value at position index (not a clone)
 * @return byte[] the reference
 * @param index int
 * @exception hla.rti.ArrayIndexOutOfBounds
 */
public byte[] getValueReference ( int index) throws ArrayIndexOutOfBounds;
/**
 * @return int Number of attribute handle-value pairs
 */
public int size ( );
}
```

B.80 Region.java

```
package hla.rti;
```

```
/**
 *
 * Represents a Region in federate's space.
 * A federate creates a Region by calling RTIambassador.createRegion.
 * The federate mdifies the Region by invoking Region methods
 * on it. The federate modifies a Region by first modifying
 * its local instance, then supplying the modified instance
 * to RTIambassador.notifyOfRegionModification.
 *
 * The Region is conceptually an array, with the extents addressed
 * by index running from 0 to getNumberOfExtents()-1.
 */
public interface Region {

/**
 * @return long Number of extents in this Region
 */
public long getNumberOfExtents ( );
/**
 * @return long Lower bound of extent along indicated dimension
 * @param extentIndex int
 * @param dimensionHandle int
 * @exception hla.rti.ArrayIndexOutOfBounds
 */
public long getRangeLowerBound (int extentIndex, int dimensionHandle) throws
ArrayIndexOutOfBounds;
/**
 * @return long Upper bound of extent along indicated dimension
 * @param extentIndex int
 * @param dimensionHandle int
 * @exception hla.rti.ArrayIndexOutOfBounds
 */
public long getRangeUpperBound (int extentIndex, int dimensionHandle) throws
ArrayIndexOutOfBounds;
/**
 * @return int Handle of routing space of which this Region is a subset
 */
public int getSpaceHandle ( );
/**
 * Modify lower bound of extent along indicated dimension.
 * @param extentIndex int
 * @param dimensionHandle int
 * @param newLowerBound long
 * @exception hla.rti.ArrayIndexOutOfBounds
 */
public void setRangeLowerBound ( int extentIndex, int dimensionHandle, long
newLowerBound) throws ArrayIndexOutOfBounds;
/**
 * Modify upper bound of extent along indicated dimension.
 * @param extentIndex int
 * @param dimensionHandle int
 * @param newUpperBound long
 * @exception hla.rti.ArrayIndexOutOfBounds The exception description.
 */
public void setRangeUpperBound (int extentIndex, int dimensionHandle, long
newUpperBound) throws ArrayIndexOutOfBounds;
```

```
}
```

B.81 RegionInUse.java

```
package hla.rti;

public final class RegionInUse extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public RegionInUse(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public RegionInUse(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.82 RegionNotKnown.java

```
package hla.rti;

/**
 * Public exception class RegionNotKnown
*/

public final class RegionNotKnown extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public RegionNotKnown(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public RegionNotKnown(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.83 ResignAction.java

```
package hla.rti;

/**
 * Holder for the enumeration values for possible resign actions.
*/

public final class ResignAction {
```

```
  static public final int RELEASE_ATTRIBUTES = 1;
  static public final int DELETE_OBJECTS = 2;
  static public final int DELETE_OBJECTS_AND_RELEASE_ATTRIBUTES = 3;
  static public final int NO_ACTION = 4;

}
```

B.84 RestoreInProgress.java

```
package hla.rti;

/**
 * Public exception class RestoreInProgress
*/
public final class RestoreInProgress extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public RestoreInProgress(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public RestoreInProgress(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.85 RestoreNotRequested.java

```
package hla.rti;

/**
 * Public exception class RestoreNotRequested
*/

public final class RestoreNotRequested extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public RestoreNotRequested(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public RestoreNotRequested(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.86 RTIambassador.java

```
package hla.rti;

/**
Memory Management Conventions for Parameters

All Java parameters, including object references, are passed by value.
Therefore there is no need to specify further conventions for primitive
types.

Unless otherwise noted, reference parameters adhere to the following
convention:
The referenced object is created (or acquired) by the caller. The callee must
copy during the call anything it wishes to save beyond the completion of the
call.

Unless otherwise noted, a reference returned from a method represents a new
object created by the callee. The caller is free to modify the object whose
reference is returned.


*/

/**
 * The RTI presents this interface to the federate.
 * RTI implementer must implement this.
*/

public interface RTIambassador {

////////////////////////////////////
// Federation Management Services //
////////////////////////////////////

  //4.2
  public void createFederationExecution (
    String       executionName,
    java.net.URL fed)
  throws
    FederationExecutionAlreadyExists,
    CouldNotOpenFED,
    ErrorReadingFED,
    RTIinternalError,
    ConcurrentAccessAttempted;

  //4.3
  public void destroyFederationExecution (String executionName)
  throws
    FederatesCurrentlyJoined,
    FederationExecutionDoesNotExist,
    RTIinternalError,
    ConcurrentAccessAttempted;

  //4.4
  public int joinFederationExecution(
    String             federateType,
    String             federationExecutionName,
```

```
  FederateAmbassador federateReference)
throws
  FederateAlreadyExecutionMember,
  FederationExecutionDoesNotExist,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

//4.4
public int joinFederationExecution(
  String                federateType,
  String                federationExecutionName,
  FederateAmbassador    federateReference,
  MobileFederateServices serviceReferences)
throws
  FederateAlreadyExecutionMember,
  FederationExecutionDoesNotExist,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

//4.5
public void resignFederationExecution (
  int          resignAction)
throws
  FederateOwnsAttributes,
  FederateNotExecutionMember,
  InvalidResignAction,
  RTIinternalError,
  ConcurrentAccessAttempted;

//4.6
public void registerFederationSynchronizationPoint (
  String synchronizationPointLabel,
  byte[] userSuppliedTag)
throws
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

//4.6
public void registerFederationSynchronizationPoint (
  String            synchronizationPointLabel,
  byte[]            userSuppliedTag,
  FederateHandleSet synchronizationSet)
throws
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

//4.9
```

```
public void synchronizationPointAchieved (
  String synchronizationPointLabel)
throws
  SynchronizationLabelNotAnnounced,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 4.11
public void requestFederationSave (
  String         label,
  LogicalTime    theTime)
throws
  FederationTimeAlreadyPassed,
  InvalidFederationTime,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 4.11
public void requestFederationSave (
  String label)
throws
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 4.13
public void federateSaveBegun ()
throws
  SaveNotInitiated,
  FederateNotExecutionMember,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 4.14
public void federateSaveComplete ()
throws
  SaveNotInitiated,
  FederateNotExecutionMember,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 4.14
public void federateSaveNotComplete ()
throws
  SaveNotInitiated,
  FederateNotExecutionMember,
  RestoreInProgress,
```

```
    RTIinternalError,
    ConcurrentAccessAttempted;

  // 4.16
  public void requestFederationRestore (
    String label)
  throws
    FederateNotExecutionMember,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError,
    ConcurrentAccessAttempted;

  // 4.20
  public void federateRestoreComplete ()
  throws
    RestoreNotRequested,
    FederateNotExecutionMember,
    SaveInProgress,
    RTIinternalError,
    ConcurrentAccessAttempted;

  // 4.20
  public void federateRestoreNotComplete ()
  throws
    RestoreNotRequested,
    FederateNotExecutionMember,
    SaveInProgress,
    RTIinternalError,
    ConcurrentAccessAttempted;

////////////////////////////////////
// Declaration Management Services //
////////////////////////////////////

  // 5.2
  public void publishObjectClass (
    int                 theClass,
    AttributeHandleSet  attributeList)
  throws
    ObjectClassNotDefined,
    AttributeNotDefined,
    OwnershipAcquisitionPending,
    FederateNotExecutionMember,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError,
    ConcurrentAccessAttempted;

  // 5.3
  public void unpublishObjectClass (
    int theClass)
  throws
    ObjectClassNotDefined,
    ObjectClassNotPublished,
    OwnershipAcquisitionPending,
    FederateNotExecutionMember,
```

```
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 5.4
public void publishInteractionClass (
  int theInteraction)
throws
  InteractionClassNotDefined,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 5.5
public void unpublishInteractionClass (
  int theInteraction)
throws
  InteractionClassNotDefined,
  InteractionClassNotPublished,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 5.6
public void subscribeObjectClassAttributes (
  int                 theClass,
  AttributeHandleSet  attributeList)
throws
  ObjectClassNotDefined,
  AttributeNotDefined,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 5.6
public void subscribeObjectClassAttributesPassively (
  int                 theClass,
  AttributeHandleSet  attributeList)
throws
  ObjectClassNotDefined,
  AttributeNotDefined,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 5.7
public void unsubscribeObjectClass (
  int theClass)
```

```
  throws
    ObjectClassNotDefined,
    ObjectClassNotSubscribed,
    FederateNotExecutionMember,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError,
    ConcurrentAccessAttempted;

  // 5.8
  public void subscribeInteractionClass (
    int     theClass)
  throws
    InteractionClassNotDefined,
    FederateNotExecutionMember,
    FederateLoggingServiceCalls,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError,
    ConcurrentAccessAttempted;

  // 5.8
  public void subscribeInteractionClassPassively (
    int     theClass)
  throws
    InteractionClassNotDefined,
    FederateNotExecutionMember,
    FederateLoggingServiceCalls,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError,
    ConcurrentAccessAttempted;

  // 5.9
  public void unsubscribeInteractionClass (
    int theClass)
  throws
    InteractionClassNotDefined,
    InteractionClassNotSubscribed,
    FederateNotExecutionMember,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError,
    ConcurrentAccessAttempted;

////////////////////////////////
// Object Management Services //
////////////////////////////////

  // 6.2
  public int
  registerObjectInstance (
       int theClass)
  throws
    ObjectClassNotDefined,
    ObjectClassNotPublished,
    FederateNotExecutionMember,
```

```
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 6.2
public int
registerObjectInstance (
  int    theClass,
  String theObject)
throws
  ObjectClassNotDefined,
  ObjectClassNotPublished,
  ObjectAlreadyRegistered,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 6.4
public void updateAttributeValues (
  int                theObject,
  SuppliedAttributes theAttributes,
  byte[]             userSuppliedTag)
throws
  ObjectNotKnown,
  AttributeNotDefined,
  AttributeNotOwned,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 6.4
public EventRetractionHandle
updateAttributeValues (
  int                theObject,
  SuppliedAttributes theAttributes,
  byte[]             userSuppliedTag,
  LogicalTime        theTime)
throws
  ObjectNotKnown,
  AttributeNotDefined,
  AttributeNotOwned,
  InvalidFederationTime,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 6.6
public void sendInteraction (
  int                 theInteraction,
  SuppliedParameters  theParameters,
```

```
  byte[]              userSuppliedTag)
throws
  InteractionClassNotDefined,
  InteractionClassNotPublished,
  InteractionParameterNotDefined,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 6.6
public EventRetractionHandle
sendInteraction (
  int                 theInteraction,
  SuppliedParameters theParameters,
  byte[]              userSuppliedTag,
  LogicalTime         theTime)
throws
  InteractionClassNotDefined,
  InteractionClassNotPublished,
  InteractionParameterNotDefined,
  InvalidFederationTime,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 6.8
public void deleteObjectInstance (
  int    ObjectHandle,
  byte[] userSuppliedTag)
throws
  ObjectNotKnown,
  DeletePrivilegeNotHeld,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

//6.8
public EventRetractionHandle
deleteObjectInstance (
  int             ObjectHandle,
  byte[]          userSuppliedTag,
  LogicalTime     theTime)
throws
  ObjectNotKnown,
  DeletePrivilegeNotHeld,
  InvalidFederationTime,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;
```

```
// 6.10
public void localDeleteObjectInstance (
  int    ObjectHandle)
throws
  ObjectNotKnown,
  FederateOwnsAttributes,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 6.11
public void changeAttributeTransportationType (
  int                theObject,
  AttributeHandleSet theAttributes,
  int                theType)
throws
  ObjectNotKnown,
  AttributeNotDefined,
  AttributeNotOwned,
  InvalidTransportationHandle,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 6.12
public void changeInteractionTransportationType (
  int theClass,
  int theType)
throws
  InteractionClassNotDefined,
  InteractionClassNotPublished,
  InvalidTransportationHandle,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 6.15
public void requestObjectAttributeValueUpdate (
  int                 theObject,
  AttributeHandleSet  theAttributes)
throws
  ObjectNotKnown,
  AttributeNotDefined,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 6.15
```

```
  public void requestClassAttributeValueUpdate (
    int                theClass,
    AttributeHandleSet theAttributes)
  throws
    ObjectClassNotDefined,
    AttributeNotDefined,
    FederateNotExecutionMember,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError,
    ConcurrentAccessAttempted;

///////////////////////////////////
// Ownership Management Services //
///////////////////////////////////

  // 7.2
  public void unconditionalAttributeOwnershipDivestiture (
    int                theObject,
    AttributeHandleSet theAttributes)
  throws
    ObjectNotKnown,
    AttributeNotDefined,
    AttributeNotOwned,
    FederateNotExecutionMember,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError,
    ConcurrentAccessAttempted;

  // 7.3
  public void negotiatedAttributeOwnershipDivestiture (
    int                theObject,
    AttributeHandleSet theAttributes,
    byte[]             userSuppliedTag)
  throws
    ObjectNotKnown,
    AttributeNotDefined,
    AttributeNotOwned,
    AttributeAlreadyBeingDivested,
    FederateNotExecutionMember,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError,
    ConcurrentAccessAttempted;

  // 7.7
  public void attributeOwnershipAcquisition (
    int                theObject,
    AttributeHandleSet desiredAttributes,
    byte[]             userSuppliedTag)
  throws
    ObjectNotKnown,
    ObjectClassNotPublished,
    AttributeNotDefined,
    AttributeNotPublished,
    FederateOwnsAttributes,
```

```
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 7.8
public void attributeOwnershipAcquisitionIfAvailable (
  int                theObject,
  AttributeHandleSet desiredAttributes)
throws
  ObjectNotKnown,
  ObjectClassNotPublished,
  AttributeNotDefined,
  AttributeNotPublished,
  FederateOwnsAttributes,
  AttributeAlreadyBeingAcquired,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 7.11
public AttributeHandleSet
attributeOwnershipReleaseResponse (
  int                theObject,
  AttributeHandleSet theAttributes)
throws
  ObjectNotKnown,
  AttributeNotDefined,
  AttributeNotOwned,
  FederateWasNotAskedToReleaseAttribute,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 7.12
public void cancelNegotiatedAttributeOwnershipDivestiture (
  int                theObject,
  AttributeHandleSet theAttributes)
throws
  ObjectNotKnown,
  AttributeNotDefined,
  AttributeNotOwned,
  AttributeDivestitureWasNotRequested,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 7.13
public void cancelAttributeOwnershipAcquisition (
  int                theObject,
```

```
    AttributeHandleSet theAttributes)
  throws
    ObjectNotKnown,
    AttributeNotDefined,
    AttributeAlreadyOwned,
    AttributeAcquisitionWasNotRequested,
    FederateNotExecutionMember,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError,
    ConcurrentAccessAttempted;

  // 7.15
  public void queryAttributeOwnership (
    int theObject,
    int theAttribute)
  throws
    ObjectNotKnown,
    AttributeNotDefined,
    FederateNotExecutionMember,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError,
    ConcurrentAccessAttempted;

  // 7.17
  public boolean
  isAttributeOwnedByFederate (
    int theObject,
    int theAttribute)
  throws
    ObjectNotKnown,
    AttributeNotDefined,
    FederateNotExecutionMember,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError,
    ConcurrentAccessAttempted;

//////////////////////////////
// Time Management Services //
//////////////////////////////

  // 8.2
  public void enableTimeRegulation (
    LogicalTime         theFederateTime,
    LogicalTimeInterval theLookahead)
  throws
    TimeRegulationAlreadyEnabled,
    EnableTimeRegulationPending,
    TimeAdvanceAlreadyInProgress,
    InvalidFederationTime,
    InvalidLookahead,
    FederateNotExecutionMember,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError,
```

```
  ConcurrentAccessAttempted;

// 8.4
public void disableTimeRegulation ()
throws
  TimeRegulationWasNotEnabled,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 8.5
public void enableTimeConstrained ()
throws
  TimeConstrainedAlreadyEnabled,
  EnableTimeConstrainedPending,
  TimeAdvanceAlreadyInProgress,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 8.7
public void disableTimeConstrained ()
throws
  TimeConstrainedWasNotEnabled,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 8.8
public void timeAdvanceRequest (
  LogicalTime    theTime)
throws
  InvalidFederationTime,
  FederationTimeAlreadyPassed,
  TimeAdvanceAlreadyInProgress,
  EnableTimeRegulationPending,
  EnableTimeConstrainedPending,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 8.9
public void timeAdvanceRequestAvailable (
  LogicalTime    theTime)
throws
  InvalidFederationTime,
  FederationTimeAlreadyPassed,
  TimeAdvanceAlreadyInProgress,
  EnableTimeRegulationPending,
```

```
  EnableTimeConstrainedPending,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 8.10
public void nextEventRequest (
  LogicalTime    theTime)
throws
  InvalidFederationTime,
  FederationTimeAlreadyPassed,
  TimeAdvanceAlreadyInProgress,
  EnableTimeRegulationPending,
  EnableTimeConstrainedPending,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 8.11
public void nextEventRequestAvailable (
  LogicalTime    theTime)
throws
  InvalidFederationTime,
  FederationTimeAlreadyPassed,
  TimeAdvanceAlreadyInProgress,
  EnableTimeRegulationPending,
  EnableTimeConstrainedPending,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 8.12
public void flushQueueRequest (
  LogicalTime    theTime)
throws
  InvalidFederationTime,
  FederationTimeAlreadyPassed,
  TimeAdvanceAlreadyInProgress,
  EnableTimeRegulationPending,
  EnableTimeConstrainedPending,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 8.14
public void enableAsynchronousDelivery()
throws
  AsynchronousDeliveryAlreadyEnabled,
  FederateNotExecutionMember,
```

```
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 8.15
public void disableAsynchronousDelivery()
throws
  AsynchronousDeliveryAlreadyDisabled,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 8.16
public LogicalTime queryLBTS ()
throws
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 8.17
public LogicalTime queryFederateTime ()
throws
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 8.18
public LogicalTime queryMinNextEventTime ()
throws
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 8.19
public void modifyLookahead (
  LogicalTimeInterval theLookahead)
throws
  InvalidLookahead,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 8.20
public LogicalTimeInterval queryLookahead ()
throws
  FederateNotExecutionMember,
```

```
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError,
    ConcurrentAccessAttempted;

  // 8.21
  public void retract (
    EventRetractionHandle theHandle)
  throws
    InvalidRetractionHandle,
    FederateNotExecutionMember,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError,
    ConcurrentAccessAttempted;

  // 8.23
  public void changeAttributeOrderType (
    int                 theObject,
    AttributeHandleSet theAttributes,
    int                 theType)
  throws
    ObjectNotKnown,
    AttributeNotDefined,
    AttributeNotOwned,
    InvalidOrderingHandle,
    FederateNotExecutionMember,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError,
    ConcurrentAccessAttempted;

  // 8.24
  public void changeInteractionOrderType (
    int theClass,
    int theType)
  throws
    InteractionClassNotDefined,
    InteractionClassNotPublished,
    InvalidOrderingHandle,
    FederateNotExecutionMember,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError,
    ConcurrentAccessAttempted;

////////////////////////////////
// Data Distribution Management //
////////////////////////////////

  // 9.2
  public Region
  createRegion (
    int spaceHandle,
    int numberOfExtents)
  throws
    SpaceNotDefined,
```

```
  InvalidExtents,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 9.3
public void notifyOfRegionModification (
  Region modifiedRegionInstance)
throws
  RegionNotKnown,
  InvalidExtents,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 9.4
public void deleteRegion (
  Region  theRegion)
throws
  RegionNotKnown,
  RegionInUse,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

//9.5
public int
registerObjectInstanceWithRegion (
  int      theClass,
  int[]    theAttributes,
  Region[] theRegions)
throws
  ObjectClassNotDefined,
  ObjectClassNotPublished,
  AttributeNotDefined,
  AttributeNotPublished,
  RegionNotKnown,
  InvalidRegionContext,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 9.5
public int
registerObjectInstanceWithRegion (
  int      theClass,
  String   theObject,
  int[]    theAttributes,
  Region[] theRegions)
```

```
throws
  ObjectClassNotDefined,
  ObjectClassNotPublished,
  AttributeNotDefined,
  AttributeNotPublished,
  RegionNotKnown,
  InvalidRegionContext,
  ObjectAlreadyRegistered,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 9.6
public void associateRegionForUpdates (
  Region             theRegion,
  int                theObject,
  AttributeHandleSet theAttributes)
throws
  ObjectNotKnown,
  AttributeNotDefined,
  InvalidRegionContext,
  RegionNotKnown,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 9.7
public void unassociateRegionForUpdates (
  Region theRegion,
  int    theObject)
throws
  ObjectNotKnown,
  InvalidRegionContext,
  RegionNotKnown,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 9.8
public void subscribeObjectClassAttributesWithRegion (
  int                theClass,
  Region             theRegion,
  AttributeHandleSet attributeList)
throws
  ObjectClassNotDefined,
  AttributeNotDefined,
  RegionNotKnown,
  InvalidRegionContext,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
```

```
  RTIinternalError,
  ConcurrentAccessAttempted;

// 9.8
public void subscribeObjectClassAttributesPassivelyWithRegion (
  int                theClass,
  Region             theRegion,
  AttributeHandleSet attributeList)
throws
  ObjectClassNotDefined,
  AttributeNotDefined,
  RegionNotKnown,
  InvalidRegionContext,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 9.9
public void unsubscribeObjectClassWithRegion (
  int    theClass,
  Region theRegion)
throws
  ObjectClassNotDefined,
  RegionNotKnown,
  FederateNotSubscribed,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 9.10
public void subscribeInteractionClassWithRegion (
  int     theClass,
  Region  theRegion)
throws
  InteractionClassNotDefined,
  RegionNotKnown,
  InvalidRegionContext,
  FederateLoggingServiceCalls,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 9.10
public void subscribeInteractionClassPassivelyWithRegion (
  int     theClass,
  Region  theRegion)
throws
  InteractionClassNotDefined,
  RegionNotKnown,
  InvalidRegionContext,
  FederateLoggingServiceCalls,
```

```
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 9.11
public void unsubscribeInteractionClassWithRegion (
  int    theClass,
  Region theRegion)
throws
  InteractionClassNotDefined,
  InteractionClassNotSubscribed,
  RegionNotKnown,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

//9.12
public void sendInteractionWithRegion (
  int                theInteraction,
  SuppliedParameters theParameters,
  byte[]             userSuppliedTag,
  Region             theRegion)
throws
  InteractionClassNotDefined,
  InteractionClassNotPublished,
  InteractionParameterNotDefined,
  RegionNotKnown,
  InvalidRegionContext,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
  ConcurrentAccessAttempted;

// 9.12
public EventRetractionHandle
sendInteractionWithRegion (
  int                theInteraction,
  SuppliedParameters theParameters,
  byte[]             userSuppliedTag,
  Region             theRegion,
  LogicalTime        theTime)
throws
  InteractionClassNotDefined,
  InteractionClassNotPublished,
  InteractionParameterNotDefined,
  InvalidFederationTime,
  RegionNotKnown,
  InvalidRegionContext,
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError,
```

```
    ConcurrentAccessAttempted;

  // 9.13
  public void requestClassAttributeValueUpdateWithRegion (
    int                theClass,
    AttributeHandleSet theAttributes,
    Region             theRegion)
  throws
    ObjectClassNotDefined,
    AttributeNotDefined,
    RegionNotKnown,
    FederateNotExecutionMember,
    SaveInProgress,
    RestoreInProgress,
    RTIinternalError,
    ConcurrentAccessAttempted;

//////////////////////////
// RTI Support Services //
//////////////////////////

  // 10.2
  public int
  getObjectClassHandle (
    String theName)
  throws
    NameNotFound,
    FederateNotExecutionMember,
    RTIinternalError;

  // 10.3
  public String
  getObjectClassName (
    int theHandle)
  throws
    ObjectClassNotDefined,
    FederateNotExecutionMember,
    RTIinternalError;

  // 10.4
  public int
  getAttributeHandle (
    String theName,
    int    whichClass)
  throws
    ObjectClassNotDefined,
    NameNotFound,
    FederateNotExecutionMember,
    RTIinternalError;

  // 10.5
  public String
  getAttributeName (
    int theHandle,
    int whichClass)
  throws
    ObjectClassNotDefined,
```

```
  AttributeNotDefined,
  FederateNotExecutionMember,
  RTIinternalError;

// 10.6
public int
getInteractionClassHandle (
  String theName)
throws
  NameNotFound,
  FederateNotExecutionMember,
  RTIinternalError;

// 10.7
public String
getInteractionClassName (
  int theHandle)
throws
  InteractionClassNotDefined,
  FederateNotExecutionMember,
  RTIinternalError;

// 10.8
public int
getParameterHandle (
  String theName,
  int whichClass)
throws
  InteractionClassNotDefined,
  NameNotFound,
  FederateNotExecutionMember,
  RTIinternalError;

// 10.9
public String
getParameterName (
  int        theHandle,
  int whichClass)
throws
  InteractionClassNotDefined,
  InteractionParameterNotDefined,
  FederateNotExecutionMember,
  RTIinternalError;

// 10.10
public int
getObjectInstanceHandle (
  String theName)
throws
  ObjectNotKnown,
  FederateNotExecutionMember,
  RTIinternalError;

// 10.11
public String
getObjectInstanceName (
  int theHandle)
```

```
throws
  ObjectNotKnown,
  FederateNotExecutionMember,
  RTIinternalError;

// 10.12
public int
getRoutingSpaceHandle (
  String theName)
throws
  NameNotFound,
  FederateNotExecutionMember,
  RTIinternalError;

// 10.13
public String
getRoutingSpaceName (
  int theHandle)
throws
  SpaceNotDefined,
  FederateNotExecutionMember,
  RTIinternalError;

// 10.14
public int
getDimensionHandle (
  String theName,
  int    whichSpace)
throws
  SpaceNotDefined,
  NameNotFound,
  FederateNotExecutionMember,
  RTIinternalError;

// 10.15
public String
getDimensionName (
  int theHandle,
  int whichClass)
throws
  SpaceNotDefined,
  DimensionNotDefined,
  FederateNotExecutionMember,
  RTIinternalError;

// 10.16
public int
getAttributeRoutingSpaceHandle (
  int theHandle,
  int whichClass)
throws
  ObjectClassNotDefined,
  AttributeNotDefined,
  FederateNotExecutionMember,
  RTIinternalError;

// 10.17
```

```
public int
getObjectClass (
  int theObject)
throws
  ObjectNotKnown,
  FederateNotExecutionMember,
  RTIinternalError;

// 10.18
public int
getInteractionRoutingSpaceHandle (
  int theHandle)
throws
  InteractionClassNotDefined,
  FederateNotExecutionMember,
  RTIinternalError;

// 10.19
public int
getTransportationHandle (
  String theName)
throws
  NameNotFound,
  FederateNotExecutionMember,
  RTIinternalError;

// 10.20
public String
getTransportationName (
  int theHandle)
throws
  InvalidTransportationHandle,
  FederateNotExecutionMember,
  RTIinternalError;

// 10.21
public int
getOrderingHandle (
  String theName)
throws
  NameNotFound,
  FederateNotExecutionMember,
  RTIinternalError;

// 10.22
public String
getOrderingName (
  int theHandle)
throws
  InvalidOrderingHandle,
  FederateNotExecutionMember,
  RTIinternalError;

// 10.23
public void enableClassRelevanceAdvisorySwitch()
throws
  FederateNotExecutionMember,
```

```
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError;

// 10.24
public void disableClassRelevanceAdvisorySwitch()
throws
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError;

// 10.25
public void enableAttributeRelevanceAdvisorySwitch()
throws
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError;

// 10.26
public void disableAttributeRelevanceAdvisorySwitch()
throws
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError;

// 10.27
public void enableAttributeScopeAdvisorySwitch()
throws
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError;

// 10.28
public void disableAttributeScopeAdvisorySwitch()
throws
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError;

// 10.29
public void enableInteractionRelevanceAdvisorySwitch()
throws
  FederateNotExecutionMember,
  SaveInProgress,
  RestoreInProgress,
  RTIinternalError;

// 10.30
public void disableInteractionRelevanceAdvisorySwitch()
throws
  FederateNotExecutionMember,
  SaveInProgress,
```

```
    RestoreInProgress,
    RTIinternalError;

  public Region getRegion(int regionToken)
     throws
     FederateNotExecutionMember,
     ConcurrentAccessAttempted,
     RegionNotKnown,
     RTIinternalError;

  public int getRegionToken(Region region)
     throws
     FederateNotExecutionMember,
     ConcurrentAccessAttempted,
     RegionNotKnown,
     RTIinternalError;

   public void tick()
   throws
      RTIinternalError,
      ConcurrentAccessAttempted;
}
```

B.87 RTIexception.java

```
package hla.rti;

/**
 * Superclass of all exceptions thrown by the RTI.
 * All RTI exceptions must be caught or specified.
*/
public class RTIexception extends Exception {
  protected int _serial;


  /**
   * @param reason    String to be carried with exception; equivalent
   *                  of 'message' parameter in .java.lang.Exception
   */
  public RTIexception(String reason) {
      super(reason);
      _serial = 0;
  }
  /**
  * @param serial    serial number also printed with the exception:
  *                   programmer can assign these to tell where the
  *                   exception was thrown
   */
  public RTIexception(String reason, int serial) {
      super(reason);
      _serial = serial;
  }

  public int getSerial()
  {
     return _serial;
  }
```

```
  public String toString() {
      String rep = super.toString();
      return rep += " serial:" + _serial;
  }
}
```

B.88 RTIinternalError.java

```
package hla.rti;

/**
 * Public exception class RTIinternalError. This is deliberately
 * not a final class.
*/
public class RTIinternalError extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public RTIinternalError(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public RTIinternalError(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.89 SaveInProgress.java

```
package hla.rti;

/**
 * Public exception class SaveInProgress
*/
public final class SaveInProgress extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public SaveInProgress(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public SaveInProgress(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.90 SaveNotInitiated.java

```
package hla.rti;

/**
 * Public exception class SaveNotInitiated
*/

public final class SaveNotInitiated extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public SaveNotInitiated(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public SaveNotInitiated(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.91 SpaceNotDefined.java

```
package hla.rti;

/**
 * Public exception class SpaceNotDefined
*/

public final class SpaceNotDefined extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public SpaceNotDefined(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public SpaceNotDefined(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.92 SpecifiedSaveLabelDoesNotExist.java

```
package hla.rti;

/**
 * Public exception class SpecifiedSaveLabelDoesNotExist
*/
```

```
public final class SpecifiedSaveLabelDoesNotExist extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public SpecifiedSaveLabelDoesNotExist(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public SpecifiedSaveLabelDoesNotExist(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.93 SuppliedAttributes.java

```
package hla.rti;

/**
 * This packages the attributes supplied to the RTI for
 * updateAttributeValues. This is conceptually an array
 * with an initial capacity and the ability to grow.
 * You enumerate by stepping index from 0 to size()-1.
 *
 */
public interface SuppliedAttributes {

/**
 * Add pair beyond last index.
 * @param handle int
 * @param value byte[]
 */
public void add (int handle, byte[] value );
/**
 * Removes all handles & values.
 */
public void empty ( );
/**
 * Return handle at index position.
 * @return int attribute handle
 * @param index int
 * @exception hla.rti.ArrayIndexOutOfBounds
 */
public int getHandle ( int index) throws ArrayIndexOutOfBounds;
/**
 * Return copy of value at index position.
 * @return byte[] copy (clone) of value
 * @param index int
 * @exception hla.rti.ArrayIndexOutOfBounds
 */
public byte[] getValue ( int index) throws ArrayIndexOutOfBounds;
/**
 * Return length of value at index position.
 * @return int value length
```

```
 * @param index int
 * @exception hla.rti.ArrayIndexOutOfBounds
 */
public int getValueLength ( int index) throws ArrayIndexOutOfBounds;
/**
 * Get the reference of the value at position index (not a clone)
 * @return byte[] the reference
 * @param index int
 * @exception hla.rti.ArrayIndexOutOfBounds
 */
public byte[] getValueReference ( int index) throws ArrayIndexOutOfBounds;
/**
 * Remove handle & value corresponding to handle. All other elements shifted
down.
 * Not safe during iteration.
 * @param handle int
 * @exception hla.rti.ArrayIndexOutOfBounds if handle not in set
 */
public void remove ( int handle) throws ArrayIndexOutOfBounds;
/**
 * Remove handle & value at index position. All other elements shifted down.
 * Not safe during iteration.
 * @param index int
 * @exception hla.rti.ArrayIndexOutOfBounds
 */
public void removeAt ( int index) throws ArrayIndexOutOfBounds;
/**
 * @return int Number of elements
 */
public int size ( );
}
```

B.94 SuppliedAttributesFactory.java

```
package hla.rti;

/**
 * Factory for SuppliedAttributes instances.
 */
public interface SuppliedAttributesFactory {

   /**
    * Creates a new SuppliedAttributes instance with specified initial
capacity.
    * @return hla.rti.SuppliedAttributes
    * @param capacity int
    */
   public SuppliedAttributes create ( int capacity);
}
```

B.95 SuppliedParameters.java

```
package hla.rti;

/**
 * This packages the parameters supplied to the RTI for
 * sendInteraction. This is conceptually an array
```

```
 * with an initial capacity and the ability to grow.
 * You enumerate by stepping index from 0 to size()-1.
 *
 */
public interface SuppliedParameters {

/**
 * Add pair beyond last index.
 * @param handle int
 * @param value byte[]
 */
public void add (int handle, byte[] value );
/**
 * Removes all handles & values.
 */
public void empty ( );
/**
 * Return handle at index position.
 * @return int parameter handle
 * @param index int
 * @exception hla.rti.ArrayIndexOutOfBounds
 */
public int getHandle ( int index) throws ArrayIndexOutOfBounds;
/**
 * Return copy of value at index position.
 * @return byte[] copy (clone) of value
 * @param index int
 * @exception hla.rti.ArrayIndexOutOfBounds
 */
public byte[] getValue ( int index) throws ArrayIndexOutOfBounds;
/**
 * Return length of value at index position.
 * @return int value length
 * @param index int
 * @exception hla.rti.ArrayIndexOutOfBounds
 */
public int getValueLength ( int index) throws ArrayIndexOutOfBounds;
/**
 * Get the reference of the value at position index (not a clone)
 * @return byte[] the reference
 * @param index int
 * @exception hla.rti.ArrayIndexOutOfBounds
 */
public byte[] getValueReference ( int index) throws ArrayIndexOutOfBounds;
/**
 * Remove handle & value corresponding to handle. All other elements shifted
down.
 * Not safe during iteration.
 * @param handle int
 * @exception hla.rti.ArrayIndexOutOfBounds if handle not in set
 */
public void remove ( int handle) throws ArrayIndexOutOfBounds;
/**
 * Remove handle & value at index position. All other elements shifted down.
 * Not safe during iteration.
 * @param index int
 * @exception hla.rti.ArrayIndexOutOfBounds
```

```
 */
public void removeAt ( int index) throws ArrayIndexOutOfBounds;
/**
 * @return int Number of elements
 */
public int size ( );
}
```

B.96 SuppliedParametersFactory.java

```
package hla.rti;

/**
 * Factory for SuppliedParameters instances.
 */
public interface SuppliedParametersFactory {

/**
 * Creates a new SuppliedParameters instance with specified initial capacity.
 * @return hla.rti.SuppliedParameters
 * @param capacity int
 */
public SuppliedParameters create ( int capacity);
}
```

B.97 SynchronizationLabelNotAnnounced.java

```
package hla.rti;

/**
 * Public exception class AttributeNotDefined
*/
public final class SynchronizationLabelNotAnnounced extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public SynchronizationLabelNotAnnounced(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public SynchronizationLabelNotAnnounced(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.98 TimeAdvanceAlreadyInProgress.java

```
package hla.rti;

/**
 * Public exception class TimeAdvanceAlreadyInProgress
*/
```

```
public final class TimeAdvanceAlreadyInProgress extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public TimeAdvanceAlreadyInProgress(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public TimeAdvanceAlreadyInProgress(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.99 TimeAdvanceWasNotInProgress.java

```
package hla.rti;

/**
 * Public exception class TimeAdvanceWasNotInProgress
*/

public final class TimeAdvanceWasNotInProgress extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public TimeAdvanceWasNotInProgress(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public TimeAdvanceWasNotInProgress(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.100 TimeConstrainedAlreadyEnabled.java

```
package hla.rti;

/**
 * Public exception class TimeConstrainedAlreadyEnabled
*/

public final class TimeConstrainedAlreadyEnabled extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public TimeConstrainedAlreadyEnabled(String reason) {
      super(reason, 0);
```

```
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public TimeConstrainedAlreadyEnabled(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.101 TimeConstrainedWasNotEnabled.java

```
package hla.rti;

/**
 * Public exception class TimeConstrainedWasNotEnabled
*/

public final class TimeConstrainedWasNotEnabled extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public TimeConstrainedWasNotEnabled(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public TimeConstrainedWasNotEnabled(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.102 TimeRegulationAlreadyEnabled.java

```
package hla.rti;

/**
 * Public exception class TimeRegulationAlreadyEnabled
*/

public final class TimeRegulationAlreadyEnabled extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public TimeRegulationAlreadyEnabled(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public TimeRegulationAlreadyEnabled(String reason, int serial) {
      super(reason, serial);
```

```
  }
}
```

B.103 TimeRegulationWasNotEnabled.java

```
package hla.rti;

/**
 * Public exception class TimeRegulationWasNotEnabled
*/

public final class TimeRegulationWasNotEnabled extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */

  public TimeRegulationWasNotEnabled(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public TimeRegulationWasNotEnabled(String reason, int serial) {
      super(reason, serial);
  }
}
```

B.104 UnableToPerformSave.java

```
package hla.rti;

/**
 * Public exception class UnableToPerformSave
*/

public final class UnableToPerformSave extends RTIexception {

  /**
   * @param reason    String to be carried with exception
   */
  public UnableToPerformSave(String reason) {
      super(reason, 0);
  }

  /**
   * @param serial    serial number also printed with the exception
   */
  public UnableToPerformSave(String reason, int serial) {
      super(reason, serial);
  }
}
```