CERTIUserDocumentation

3.4.3-cvs

Generated by Doxygen 1.8.6

Fri Feb 28 2014 16:33:59

Contents

1 Main Page

User Documentation

This is the CERTI user documentation. The user documentation is divided into several part:

- Introduction
- Executing HLA simulation
- IEEE 1516.2 Data Encoding Functions
  - Basic Types
  - Enumeration Types
  - Fixed Array
  - Fixed Record
  - Variable Array
  - Variant Record

2 Introduction

CERTI is an Open Source HLA compliant `RunTime Infrastructure (RTI)`, which aims at being a multi-standard, multi-language RTI including:

- DoD HLA 1.3 (a.k.a. RTI-NGv6): C++, Java, Python, Fortran90, Matlab
- IEEE1516-2000 (a.k.a. HLA 1516): C++
- IEEE1516-2010 (a.k.a. HLA Evolved): work-in-progress

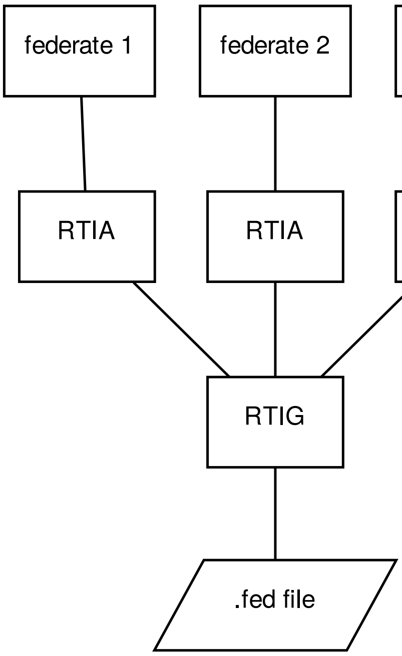
You'll find hereafter the documentation for building and installing CERTI. CERTI is primarily developed and maintained by the Toulouse research center of ONERA [`http://www.onera.fr`], the French Aerospace Labs. The primary goal of CERTI is to be used in research activities but CERTI has a growing number of users and contributors among the CERTI Open Source community.

People interested in CERTI may join the CERTI Open Source community at `https://savannah.nongnu.org/projects/certi` and/or by using the mailing list `http://lists.nongnu.org/mailman/listinfo/certi-devel` for discussion regarding CERTI usage.

3 Executing HLA simulation

3.1 CERTI executables

CERTI comes with two main executables: RTIA and RTIG.



3.1.1 certi_user_execute

If ones want to properly execute an HLA simulation using CERTI one must: (FIXME more detail to come).

1. configure PATH
2. store .fed (or .xml) FOM file in the search path of the rtig

   See Also

   CERTI FOM file search algorithm

3. run rtig,

   See Also

   RTIG

4. configure HOST/PORT/PROXY,
5. run federations, rtia is started automatically.

3.1.2 CERTI environment variables

CERTI uses a set of environment variables which may influence its execution behavior. Those variables may be set on federate execution location or on RTIG (a.k.a. The CERTI CRC) location.

Variable	Used by	Description
CERTI_HOME	RTIG	the CERTI installation base directory. This is used by the RTIG in order to look for FOM files (see RTIG).
CERTI_FOM_PATH	RTIG	the CERTI Federation Object Model search path. The variable may contain a ":" separated list of paths. This is used by the RTIG in order to look for FOM files (see RTIG).
CERTI_HOST	RTIA	machine on which RTIG is running. As soon as it starts the RTIA will try to connect to the RTIG running on CERTI_HOST (or "localhost" if the variable is not defined or void) (see RTIA).
CERTI_TCP_PORT	RTIG, RTIA	TCP port used for RTIA/RTIG communications (default: 60400)
CERTI_UDP_PORT	RTIG, RTIA	UDP port used for RTIA/RTIG communications (default: 60500)
CERTI_HTTP_PROXY	RTIA	HTTP proxy address in the format `http://host:port`. See HTTP tunneling.
http_proxy	RTIA	System-wide HTTP proxy address used if CERTI_HTTP_PROXY is not defined.
CERTI_NO_STATISTICS	RTIA	if set, do not display service calls statistics

3.1.3 RTIG: CERTI RunTime Infrastructure Gateway

The CERTI RunTime Infrastructure Gateway (RTIG) is a process which coordinate the HLA simulation with CERTI, there should be at least one rtig process for each federation. However a single RTIG may be used for several federations. The command line usage of the RTIG is following:

rtig [-v 2] [-l @IP|hostname]

- **-v** (optional) verbosity level
  - 0 -> no output
  - 1 -> small amount
  - 2 -> show fed parse
- **-l** (optional) listening address

Once the RTIG is launched an HLA Federate may interact with the RTI. In fact a federate does not talk to the RTIG directly but it uses its RTIA. RTIG is listening to RTIA connection on TCP port:

1. 60400 or,
2. the value of environment variable CERTI_TCP_PORT if it is defined

The RTIG exchange messages with the RTIA in order to satify HLA request coming from the Federate. In particular RTIG is responsible for giving to the Federate (through its RTIA) the FOM file needed to create or join the federation. When a federate calls the CreateFederationExcution API RTIG tries to open FOM file from different predefined places, using various environment variables:

1. Bare filename considered as a path provided through `FEDid_name`

2. Use CERTI federation object model serach PATH `getenv(CERTI_FOM_PATH) + FEDid_name`.

   `CERTI_FOM_PATH` environment variable may contains a list of path separated with ':'.

3. Using the `CERTI_HOME` environment variable `getenv(CERTI_HOME)+"/share/federations/"+ FEDid_name`

4. installation place plus `FEDid_name`

   `PACKAGE_INSTALL_PREFIX + "/share/federation/" + FEDid_name`

5. on Unix `"/usr/local/share/federation/" + FEDid_name` for backward compatibility reason.

3.1.4 RTIA: CERTI RunTime Infrastructure Ambassador

The CERTI RunTime Infrastructure Ambassador (RTIA) is a process which is automatically launched by the federate as soon as its RTIambassador is created. The command line usage of the RTIA is following:

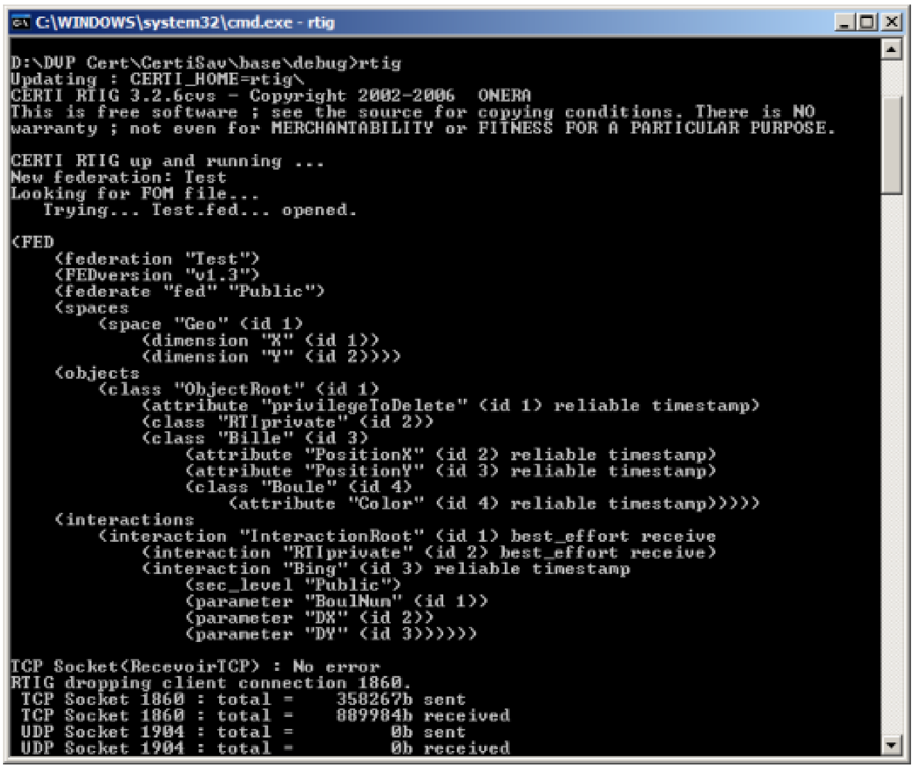
rtia [-v] [-p <port>]

- **-v** (optional) verbose, display more information
- **-p** (optional) tcp port to be used to communicate with FederateAmbassador
  RTIA will try to connect to RTIG process on the machine specified in CERTI_HOME (see CERTI environment variables) environment variable. If it is void or not set then he will try to connect to localhost. RTIA connect to TCP port specified by CERTI_TCP_PORT and UDP port specified by CERTI_UDP_PORT.

3.2 Sample federate: Billiard

Open a windows command prompt and run the RTIG.

```
rtig
```

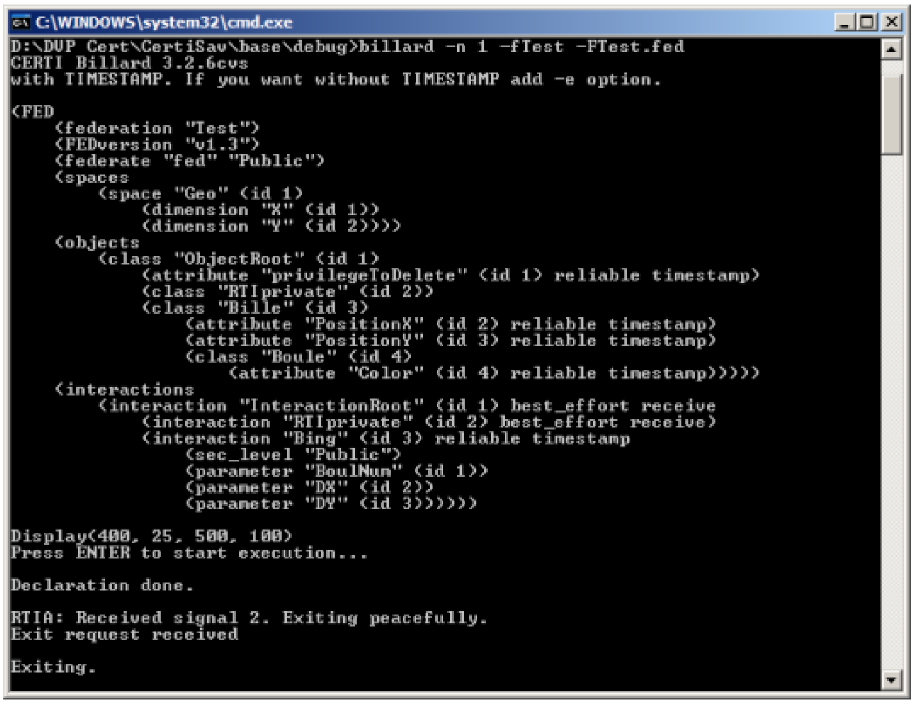


```
D:\DUP Cert\CertiSav\base\debug>rtig
Updating : CERTI_HOME=rtig\
CERTI RTIG 3.2.6cvs - Copyright 2002-2006  ONERA
This is free software ; see the source for copying conditions. There is NO
warranty ; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

CERTI RTIG up and running ...
New federation: Test
Looking for FOM file...
   Trying... Test.fed... opened.

(FED
    (federation "Test")
    (FEDversion "v1.3")
    (federate "fed" "Public")
    (spaces
        (space "Geo" (id 1)
            (dimension "X" (id 1))
            (dimension "Y" (id 2))))
    (objects
        (class "ObjectRoot" (id 1)
            (attribute "privilegeToDelete" (id 1) reliable timestamp)
            (class "RTIprivate" (id 2))
            (class "Bille" (id 3)
                (attribute "PositionX" (id 2) reliable timestamp)
                (attribute "PositionY" (id 3) reliable timestamp)
                (class "Boule" (id 4)
                    (attribute "Color" (id 4) reliable timestamp)))))
    (interactions
        (interaction "InteractionRoot" (id 1) best_effort receive
            (interaction "RTIprivate" (id 2) best_effort receive)
            (interaction "Bing" (id 3) reliable timestamp
                (sec_level "Public")
                (parameter "BoulNum" (id 1))
                (parameter "DX" (id 2))
                (parameter "DY" (id 3))))))

TCP Socket(RecevoirTCP) : No error
RTIG dropping client connection 1860.
 TCP Socket 1860 : total =    358267b sent
 TCP Socket 1860 : total =    889984b received
 UDP Socket 1904 : total =         0b sent
 UDP Socket 1904 : total =         0b received
```

Figure 1: RTIG screenshot

Open another windows command prompt and run the billard program.

```
billard -n 1 fTest FTest.fed
```



```
D:\DUP Cert\CertiSav\base\debug>billard -n 1 -fTest -FTest.fed
CERTI Billard 3.2.6cvs
with TIMESTAMP. If you want without TIMESTAMP add -e option.

(FED
    (federation "Test")
    (FEDversion "v1.3")
    (federate "fed" "Public")
    (spaces
        (space "Geo" (id 1)
            (dimension "X" (id 1))
            (dimension "Y" (id 2))))
    (objects
        (class "ObjectRoot" (id 1)
            (attribute "privilegeToDelete" (id 1) reliable timestamp)
            (class "RTIprivate" (id 2))
            (class "Bille" (id 3)
                (attribute "PositionX" (id 2) reliable timestamp)
                (attribute "PositionY" (id 3) reliable timestamp)
                (class "Boule" (id 4)
                    (attribute "Color" (id 4) reliable timestamp)))))
    (interactions
        (interaction "InteractionRoot" (id 1) best_effort receive
            (interaction "RTIprivate" (id 2) best_effort receive)
            (interaction "Bing" (id 3) reliable timestamp
                (sec_level "Public")
                (parameter "BoulNum" (id 1))
                (parameter "DX" (id 2))
                (parameter "DY" (id 3))))))

Display(400, 25, 500, 100)
Press ENTER to start execution...

Declaration done.

RTIA: Received signal 2. Exiting peacefully.
Exit request received

Exiting.
```

Figure 2: Billard consoleshot

4 Connecting to RTIG via a HTTP tunnel

To pass the RTIA–RTIG connection through firewalls, you may use the HTTP tunnel.

Federates behind a firewall may be unnable to connect to the RTIG. To connect via a HTTP tunnel

1. Set the CERTI_HOST and CERTI_TCP_PORT environment variables to RTIG address and port.

2. Set the CERTI_HTTP_PROXY environment variable to HTTP proxy address in the form http://host:port.

3. Run the federate.

If CERTI_HTTP_PROXY is not defined, the system-wide http_proxy is used. To disable HTTP tunneling, you must unset both environment variables, or set CERTI_HTTP_PROXY to an empty string.

If the HTTP proxy is directly accessible for the federate (RTIA), you can set the CERTI_HTTP_PROXY environment variable to address of the HTTP proxy, e.g. http://proxy.example.com. The default port is 3128.

Note: In the HTTP proxy configuration you may need to enable the HTTP CONNECT method for the port number defined in CERTI_TCP_PORT. For example, in the /etc/squid/squid.conf you may need to configure

```
acl CERTI_ports port 60400   # the value of CERTI_TCP_PORT
acl CONNECT method CONNECT
http_access allow CONNECT CERTI_ports
```

If you cannot access the HTTP proxy directly, you may use SSH port forwarding. The SSH client will listen to a local port and will ask the remote SSH server to open an outgoing connection to the HTTP proxy. It will then forward all traffic between the local port and the HTTP proxy inside the SSH connection.

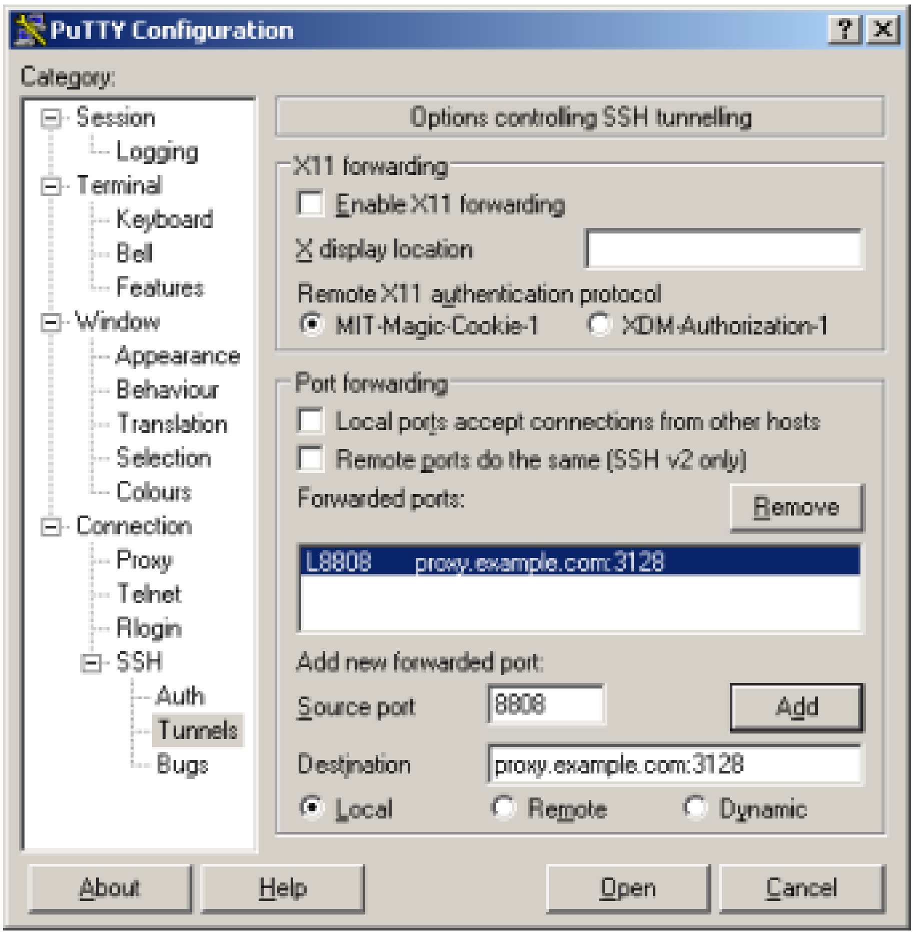
To use SSH port forwarding

1. Set the CERTI_HTTP_PROXY environment variable to an arbitrary local port number, e.g. http://localhost:8808.

2. Establish an SSH connection as follows.

On Windows you may use the PuTTY client http://www.chiark.greenend.org.uk/~sgtatham/putty

Create a SSH session and select the SSH protocol. Open the Connection – SSH – Tunnels configuration. Select "Local", enter chosen arbitrary "Source port" number (e.g. 8808) and the HTTP proxy address as "Destination". Make sure you then click "Add".



Options controlling SSH tunnelling

X11 forwarding

□ Enable X11 forwarding

X display location

Remote X11 authentication protocol

● MIT-Magic-Cookie-1 ○ XDM-Authorization-1

Port forwarding

□ Local ports accept connections from other hosts

□ Remote ports do the same (SSH v2 only)

Forwarded ports:

L8808 proxy.example.com:3128

Add new forwarded port:

Source port 8808

Destination proxy.example.com:3128

● Local ○ Remote ○ Dynamic

Figure 3: Putty Portforwarding

Most Linux systems have a SSH client installed. Use the `ssh` command.

```
ssh -L8808:proxy.example.com:3128 user@hostname
```

5 IEEE 1516.2 Data Encoding Functions

These templates implement efficient access functions that provide direct access to IEEE 1516.2 compliant data buffers.

The functions are RTI independent and thus compatible with any RTI. The data are manipulated "in situ", no temporary variables are created.

Use

```
hlaomtdif2cpp -i <file>
```

to generate FOM/SOM specific datatypes.

The following templates are provided:

- Basic Types
- Enumeration Types
- Fixed Array
- Fixed Record

- Variable Array
- Variant Record

The extensive use of template metaprogramming allows many operations to be pre-calculated during compile-time. See http://aszt.inf.elte.hu/~gsd/halado_cpp/ch06s09.html

6 Basic Types

The template

```
HLAbasicType<DATATYPE, STORAGE, ENCODING>
```

defines a user-convenient `DATATYPE`, stored in `STORAGE` using given `ENCODING`.

The data are stored in a buffer of sizeof(STORAGE).

The buffer is casted to a DATATYPE that provide data access operators. The data can be accessed in an usual way. The DATATYPE may have any sizeof(), but must have static-cast to STORAGE.
For example:

```
typedef HLAbasicType<long, uint32_t, LittleEndian> HLAinteger32BE;
HLAdata<HLAinteger32BE> value;

value = 42;
```

7 Enumeration Types

The template

```
HLAenumeratedType<ENUMERATION, REPRESENTATION>
```

defines an user-convenient ENUMERATION stored using given REPRESENTATION.

The data can be accessed in an usual way.

Some models may use one enumerated value in multiple enumerations. To avoid name collisions it's recommended to put the ENUMERATION in an individual namespace.
For example:

```
+------------+----------------+------------+--------+-----------+
| Name       | Representation | Enumerator | Values | Semantics |
+------------+----------------+------------+--------+-----------+
|            |                | HLAfalse   | 0      |           |
| HLAboolean | HLAinteger32BE +------------+--------+-----------+
|            |                | HLAfalse   | 1      |           |
+------------+----------------+------------+--------+-----------+

namespace __HLAboolean {
enum __enum {
  HLAfalse = 0,
  HLAtrue = 1
};
}
typedef HLAenumeratedType<__HLAboolean::__enum, HLAinteger32BE> HLAboolean;
HLAdata<HLAboolean> value;

value = HLAtrue;
```

8 Fixed Array

The template

```
HLAfixedArray<DATATYPE, NUMBER>
```

defines a fixed array of NUMBER elements of type DATATYPE.

The data can be accessed in an usual way.

For example:

```
+-------------+----------------+-------------+-----------------+-----------+
| Name        | Element type   | Cardinality | Encoding        | Semantics |
+-------------+----------------+-------------+-----------------+-----------+
| Coordinates | HLAinteger32BE | 3           | HLAfixedArray   |           |
+-------------+----------------+-------------+-----------------+-----------+

typedef HLAfixedArray<HLAinteger32BE,3> Coordinates;
HLAdata<Coordinates> value;

(*value)[0] = 100;
(*value)[1] = 200;
```

9 Fixed Record

The template

```
HLAfixedRecord<
  HLAfixedField<INDEX1, DATATYPE1,
  HLAfixedField<INDEX2, DATATYPE2,
  ...
  > ... > TYPENAME;
```

defines an ordered sequence of DATATYPE entries.

The data can be accessed using the field<INDEX>() function. The INDEX is a logical identifier only. The data are stored in the declaration order.

For example:

```
+-------------+-------------------------------------+----------------+-----------+
|             | Field                               |                |           |
| Record name +---------+--------------+-----------+ Encoding       | Semantics |
|             | Name    | Type         | Semantics |                |           |
+-------------+---------+--------------+-----------+----------------+-----------+
|             | FIELD_X | HLAfloat32LE |           |                |           |
|             +---------+--------------+-----------+                |           |
| Coordinates | FIELD_Y | HLAfloat32LE |           | HLAfixedRecord |           |
|             +---------+--------------+-----------+                |           |
|             | FIELD_Z | HLAfloat32LE |           |                |           |
+-------------+---------+--------------+-----------+----------------+-----------+

enum {
  FIELD_X = 0,
  FIELD_Y,
  FIELD_Z
};
typedef HLAfixedRecord<
  HLAfixedField<FIELD_X, HLAfloat32LE,
  HLAfixedField<FIELD_Y, HLAfloat32LE,
  HLAfixedField<FIELD_Z, HLAfloat32LE
  > > > > Coordinates;
HLAdata<Coordinates> value;

value->field<FIELD_X>() = 3.14;
value->field<FIELD_Y>() = 6.28;
value->field<FIELD_Z>() = 9.42;
```

10 Variable Array

The template

```
HLAvariableArray<DATATYPE>
```

defines an array of a variable number of DATATYPE elements.

The size() member must be set before accessing the data. No data are moved when the size() is changed.

For example:

```
+-------------+----------------+-------------+-----------------+-----------+
| Name        | Element type   | Cardinality | Encoding        | Semantics |
+-------------+----------------+-------------+-----------------+-----------+
| List        | HLAinteger32BE | Dynamic     | HLAvaribleArray |           |
+-------------+----------------+-------------+-----------------+-----------+

typedef HLAvariableArray<HLAinteger32BE> List;
HLAdata<List> value;

(*value).set_size(2);
(*value)[0] = 100;
(*value)[1] = 200;
```

11 Variant Record

The template

```
HLAvariantRecord<
  INDEX, DATATYPE,
  HLAvariantField<ENUMERATORS1, INDEX1, DATATYPE1,
  HLAvariantField<ENUMERATORS2, INDEX2, DATATYPE2,
  ...
  > ... > TYPENAME;
```

defines an ordered sequence of DATATYPE entries.

The data can be accessed using the field<INDEX>() function. The INDEX is a logical identifier only. The first field is a discriminant. It is followed by an alternative whose ENUMERATORS match the discriminant value.

For example:

```
+-------------+-------------------------------+------------------------------------+------------------+------
|             | Discriminant                  | Alternative                        |                  |
| Record name +------+-----------+------------+---------+--------------+-----------+ Encoding         | Semar
|             | Name | Type      | Enumerator | Name    | Type         | Semantics |                  |
+-------------+------+-----------+------------+---------+--------------+-----------+------------------+------
|             |      |           | AXIS_X     | FIELD_X | HLAfloat32LE |           |                  |
| Coordinates | TYPE | TypesEnum +------------+---------+--------------+-----------+ HLAvariantRecord |
|             |      |           | AXIS_Y     | FIELD_Y | HLAfloat32LE |           |                  |
+-------------+------+-----------+------------+---------+--------------+-----------+------------------+------

namespace __Fields {
enum __enum {
  TYPE = 0,
  FIELD_X = 101,
  FIELD_Y = 102
};
}
typedef HLAenumeratedType<__Fields::__enum, HLAinteger32BE> Fields;
typedef HLAvariantRecord<
  __Fields::TYPE, TypesEnum,
  HLAvariantField<HLAsetValue<AXIS_X>, __Fields::FIELD_X, HLAfloat32LE,
  HLAvariantField<HLAsetValue<AXIS_Y>, __Fields::FIELD_Y, HLAfloat32LE
  > > > Coordinates;
HLAdata<Coordinates> value;
```

```
value->set_discriminant(AXIS_X);
value->field<__Fields::FIELD_X>() = 3.14;
```

12 Module Documentation

12.1 RTIG

However a single RTIG may be used for several federations. The command line usage of the RTIG is following:

rtig [-v 2] [-l @IP|hostname]

- **-v** (optional) verbosity level
  - 0 -> no output
  - 1 -> small amount
  - 2 -> show fed parse
- **-l** (optional) listening address

Once the RTIG is launched an HLA Federate may interact with the RTI. In fact a federate does not talk to the RTIG directly but it uses its RTIA. RTIG is listening to RTIA connection on TCP port:

1. 60400 or,
2. the value of environment variable CERTI_TCP_PORT if it is defined

The RTIG exchange messages with the RTIA in order to satify HLA request coming from the Federate. In particular RTIG is responsible for giving to the Federate (through its RTIA) the FOM file needed to create or join the federation. When a federate calls the CreateFederationExcution API RTIG tries to open FOM file from different predefined places, using various environment variables:

1. Bare filename considered as a path provided through `FEDid_name`
2. Use CERTI federation object model serach PATH `getenv(CERTI_FOM_PATH) + FEDid_name`.
   `CERTI_FOM_PATH` environment variable may contains a list of path separated with ':'.
3. Using the `CERTI_HOME` environment variable `getenv(CERTI_HOME)+"/share/federations/"+ FEDid_name`
4. installation place plus `FEDid_name`
   `PACKAGE_INSTALL_PREFIX + "/share/federation/" + FEDid_name`
5. on Unix `"/usr/local/share/federation/" + FEDid_name` for backward compatibility reason.

12.2 CERTI FOM file search algorithm

1. Bare filename considered as a path provided through `FEDid_name`
2. Use CERTI federation object model serach PATH `getenv(CERTI_FOM_PATH) + FEDid_name`.

   `CERTI_FOM_PATH` environment variable may contains a list of path separated with ':'.
3. Using the `CERTI_HOME` environment variable `getenv(CERTI_HOME)+"/share/federations/"+ FEDid_name`
4. installation place plus `FEDid_name`

   `PACKAGE_INSTALL_PREFIX + "/share/federation/" + FEDid_name`
5. on Unix `"/usr/local/share/federation/" + FEDid_name` for backward compatibility reason.

12.3 RTIA

The command line usage of the RTIA is following:

rtia [-v] [-p <port>]

- **-v** (optional) verbose, display more information
- **-p** (optional) tcp port to be used to communicate with FederateAmbassador

RTIA will try to connect to RTIG process on the machine specified in CERTI_HOME (see CERTI environment variables) environment variable. If it is void or not set then he will try to connect to localhost. RTIA connect to TCP port specified by CERTI_TCP_PORT and UDP port specified by CERTI_UDP_PORT.

13 Class Documentation

13.1 __DiscriminantOrFieldAt< DE, DM, R, e > Struct Template Reference

Public Types

- typedef __variantRecord_if< e==DE,
  DM, typename __FieldAt< R, e >
  ::Type >::X **Type**

13.2 __FieldAt< R, i > Struct Template Reference

13.3 __FieldAt< HLAfixedEnd, d > Struct Template Reference

Public Types

- typedef HLAfixedEnd **Type**

13.4 __FieldAt< HLAfixedField< E, M, N, V >, d > Struct Template Reference

Public Types

- typedef __fixedRecord_if< d==E,
  M, typename __FieldAt< N, d >
  ::Type >::X **Type**

13.5 __FieldAt< HLAvariantEnd, e > Struct Template Reference

Public Types

- typedef HLAvariantEnd **Type**

13.6 __FieldAt< HLAvariantField< D, E, M, N, V >, e > Struct Template Reference

Public Types

- typedef __variantRecord_if< e==E,
  M, typename __FieldAt< N, e >
  ::Type >::X **Type**

13.7 __fixedRecord_if< C, Then, Else > Struct Template Reference

Public Types

- typedef Then **X**

13.8 __fixedRecord_if< false, Then, Else > Struct Template Reference

Public Types

- typedef Else **X**

13.9 __swap< T, i > Struct Template Reference

13.9.1 Detailed Description

template<class T, int i = sizeof(T)>struct libhla::__swap< T, i >

Template specializations are defined for each possible .

13.10 __swap< T, 1 > Struct Template Reference

Public Member Functions

- const T **operator()** (const T &x) const

13.11 __swap< T, 2 > Struct Template Reference

Public Member Functions

- const T **operator()** (const T &x) const

13.12 __swap< T, 4 > Struct Template Reference

Public Member Functions

- const T **operator()** (const T &x) const

13.13 __swap< T, 8 > Struct Template Reference

Public Member Functions

- const T **operator()** (const T &x) const

13.14 __variantRecord_if< C, Then, Else > Struct Template Reference

Public Types

- typedef Then **X**

13.15 __variantRecord_if< false, Then, Else > Struct Template Reference

Public Types

- typedef Else **X**

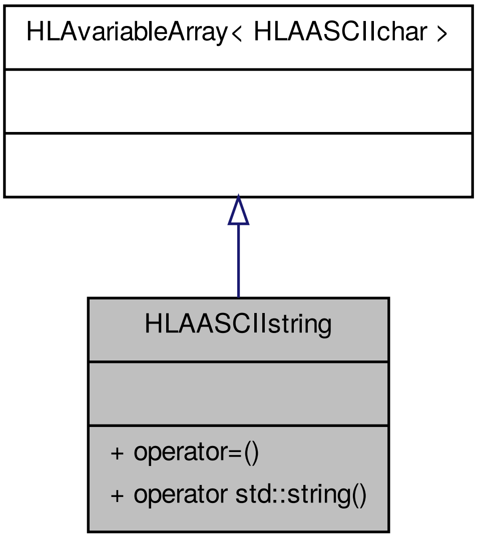
13.16 BigEndian< T > Struct Template Reference

Public Member Functions

- const T **operator()** (const T &x) const

13.17 HLAASCIIstring Struct Reference

Inheritance diagram for HLAASCIIstring:



Public Member Functions

- HLAASCIIstring & **operator=** (const std::string &it)
- **operator std::string** () const

13.18 HLAbasicType< T, S, E > Struct Template Reference

Public Member Functions

- HLAbasicType & **operator=** (const T &data)
- **operator T** () const
- void **copy** (void ∗source)

Static Public Member Functions

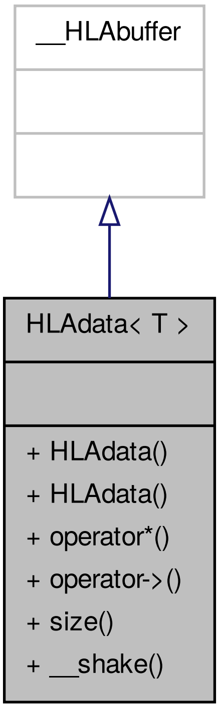
- static const size_t **emptysizeof** ()
- static const size_t **__sizeof** ()

Static Public Attributes

- static const size_t **m_octetBoundary** = sizeof(S)
- static const bool **m_isVariable** = false

13.19 HLAdata< T > Class Template Reference

Inheritance diagram for HLAdata< T >:



Public Member Functions

- HLAdata (size_t capacity=T::emptysizeof())
- HLAdata (void ∗begin, size_t capacity)
- T & **operator∗** () const
- T ∗ **operator->** () const
- virtual const size_t **size** () const
- virtual void __shake (const void ∗__that, int value, long resize)

13.20 HLAenumeratedType< E, R > Struct Template Reference

Public Member Functions

- HLAenumeratedType & **operator=** (const E &data)
- HLAenumeratedType & **operator=** (const int &data)
- **operator E** () const
- **operator int** () const
- void **copy** (void ∗source)

Static Public Member Functions

- static const size_t **emptysizeof** ()
- static const size_t **__sizeof** ()

Static Public Attributes

- static const size_t **m_octetBoundary** = R::m_octetBoundary
- static const bool **m_isVariable** = false

13.21 HLAfixedArray< M, N, hasVariable > Struct Template Reference

13.22 HLAfixedArray< M, N, false > Struct Template Reference

Public Member Functions

- M & **operator[ ]** (long i) const
- void **copy** (void ∗source)

Static Public Member Functions

- static const size_t **size** ()
- static const size_t **offset** (long i)
- static const size_t **emptysizeof** ()
- static const size_t **__sizeof** ()

Static Public Attributes

- static const size_t **m_octetBoundary** = M::m_octetBoundary
- static const bool **m_isVariable** = false

13.23 HLAfixedArray< M, N, true > Struct Template Reference

Public Member Functions

- const size_t **offset** (long i) const
- M & **operator[ ]** (long i) const
- const size_t **__sizeof** () const
- void **copy** (void ∗source)

Static Public Member Functions

- static const size_t **size** ()
- static const size_t **emptysizeof** ()

Static Public Attributes

- static const size_t **m_octetBoundary** = M::m_octetBoundary
- static const bool **m_isVariable** = true

13.24 HLAfixedEnd Struct Reference

Public Member Functions

- void **copy** (void ∗source, size_t offsD=0, size_t offsS=0)

Static Public Member Functions

- static const size_t **field_offsetof** (int d, size_t offs=0)
- static const size_t **emptysizeof** (size_t offs=0)
- static const size_t **__sizeof** (size_t offs=0)

Static Public Attributes

- static const size_t **memberBoundary** = 0
- static const size_t **m_octetBoundary** = 0
- static const bool **m_isVariable** = false

13.25 HLAfixedField< E, M, N, hasVariable > Struct Template Reference

13.26 HLAfixedField< E, M, N, false > Struct Template Reference

Public Member Functions

- void **copy** (void ∗source, size_t offsD=0, size_t offsS=0)

Static Public Member Functions

- static const size_t **field_offsetof** (int d, size_t offs=0)
- static const size_t **emptysizeof** (size_t offs=0)
- static const size_t **__sizeof** (size_t offs=0)

Static Public Attributes

- static const size_t **memberBoundary** = M::m_octetBoundary
- static const size_t **m_octetBoundary** = MAX(M::m_octetBoundary, N::m_octetBoundary)
- static const bool **m_isVariable** = false

13.27 HLAfixedField< E, M, N, true > Struct Template Reference

Public Member Functions

- const size_t **field_offsetof** (int d, size_t offs=0) const
- const size_t **__sizeof** (size_t offs=0) const
- void **copy** (void ∗source, size_t offsD=0, size_t offsS=0)

Static Public Member Functions

- static const size_t **emptysizeof** (size_t offs=0)

Static Public Attributes

- static const size_t **memberBoundary** = M::m_octetBoundary
- static const size_t **m_octetBoundary** = MAX(M::m_octetBoundary, N::m_octetBoundary)
- static const bool **m_isVariable** = true

13.28 HLAfixedRecord< R, hasVariable > Struct Template Reference

13.29 HLAfixedRecord< R, false > Struct Template Reference

Public Member Functions

- template<int i>
  __FieldAt< R, i >::Type & **field** () const
- void **copy** (void ∗source)

Static Public Member Functions

- static const size_t **emptysizeof** ()
- static const size_t **__sizeof** ()

Static Public Attributes

- static const size_t **m_octetBoundary** = R::m_octetBoundary
- static const bool **m_isVariable** = false

13.30 HLAfixedRecord< R, true > Struct Template Reference

Public Member Functions

- template<int i>
  __FieldAt< R, i >::Type & **field** () const
- const size_t **__sizeof** ()
- void **copy** (void ∗source)

Static Public Member Functions

- static const size_t **emptysizeof** ()

Static Public Attributes

- static const size_t **m_octetBoundary** = R::m_octetBoundary
- static const bool **m_isVariable** = true

13.31 HLAsetEnd Struct Reference

Static Public Member Functions

- static int **includes** (int i)

13.32 HLAsetOther< N > Struct Template Reference

Static Public Member Functions

- static int **includes** (int i)

13.33 HLAsetRange< e1, e2, N > Struct Template Reference

Static Public Member Functions

- static int **includes** (int i)

13.34 HLAsetValue< e, N > Struct Template Reference

Static Public Member Functions

- static int **includes** (int i)

13.35 HLAvariableArray< M, hasVariable > Struct Template Reference

13.36 HLAvariableArray< M, false > Struct Template Reference

Public Member Functions

- HLAinteger32BE & size () const
- void set_size (long i)
- M & **operator[]** (long i) const
- const size_t **__sizeof** () const
- void **copy** (void ∗source)

Static Public Member Functions

- static const size_t **offset** (long i)
- static const size_t **emptysizeof** ()

Static Public Attributes

- static const size_t **m_octetBoundary**
- static const bool **m_isVariable** = true

13.36.1 Member Data Documentation

13.36.1.1 const size_t m_octetBoundary [static]

Initial value:

```
=
        MAX(HLAinteger32BE::m_octetBoundary, M::m_octetBoundary)
```

13.37 HLAvariableArray< M, true > Struct Template Reference

Public Member Functions

- HLAinteger32BE & size () const
- void set_size (long i)
- const size_t **offset** (long i) const
- M & **operator[]** (long i) const
- const size_t **__sizeof** () const
- void **copy** (void ∗source)

Static Public Member Functions

- static const size_t **emptysizeof** ()

Static Public Attributes

- static const size_t **m_octetBoundary**
- static const bool **m_isVariable** = true

13.37.1 Member Data Documentation

13.37.1.1 const size_t m_octetBoundary [static]

Initial value:

```
=
        MAX(HLAinteger32BE::m_octetBoundary, M::m_octetBoundary)
```

13.38 HLAvariantEnd Struct Reference

Public Member Functions

- void **copy** (int e, void ∗source)

Static Public Member Functions

- static bool **has_field** (int d)
- static int **get_field** (int d)
- static const size_t **field_emptysizeof** (int e)
- static const size_t **field_sizeof** (int e)

Static Public Attributes

- static const size_t **m_octetBoundary** = 0
- static const bool **m_isVariable** = false

13.39 HLAvariantField< D, E, M, N, hasVariable > Struct Template Reference

13.40 HLAvariantField< D, E, M, N, false > Struct Template Reference

Public Member Functions

- void **copy** (int e, void ∗source)

Static Public Member Functions

- static bool **has_field** (int d)
- static int **get_field** (int d)
- static const size_t **field_emptysizeof** (int e)
- static const size_t **field_sizeof** (int e)

Static Public Attributes

- static const size_t **m_octetBoundary** = MAX(M::m_octetBoundary, N::m_octetBoundary)
- static const bool **m_isVariable** = false

13.41 HLAvariantField< D, E, M, N, true > Struct Template Reference

Public Member Functions

- const size_t **field_sizeof** (int e) const
- void **copy** (int e, void ∗source)

Static Public Member Functions

- static bool **has_field** (int d)
- static int **get_field** (int d)
- static const size_t **field_emptysizeof** (int e)

Static Public Attributes

- static const size_t **m_octetBoundary** = MAX(M::m_octetBoundary, N::m_octetBoundary)
- static const bool **m_isVariable** = true

13.42 HLAvariantRecord< DE, DM, R, hasVariable > Struct Template Reference

13.43 HLAvariantRecord< DE, DM, R, false > Struct Template Reference

Public Member Functions

- DM & discriminant () const
- void set_discriminant (int d)
- template<int e>
  __DiscriminantOrFieldAt< DE,
  DM, R, e >::Type & **field** () const
- const size_t **__sizeof** () const
- void **copy** (void ∗source)

Static Public Member Functions

- static const size_t **emptysizeof** ()

Static Public Attributes

- static const size_t **m_octetBoundary** = MAX(DM::m_octetBoundary, R::m_octetBoundary)
- static const bool **m_isVariable** = true

13.44 HLAvariantRecord< DE, DM, R, true > Struct Template Reference

Public Member Functions

- DM & discriminant () const
- void set_discriminant (int d)
- template<int e>
  __DiscriminantOrFieldAt< DE,
  DM, R, e >::Type & **field** () const
- const size_t **__sizeof** () const
- void **copy** (void ∗source)

Static Public Member Functions

- static const size_t **emptysizeof** ()

Static Public Attributes

- static const size_t **m_octetBoundary** = MAX(DM::m_octetBoundary, R::m_octetBoundary)
- static const bool **m_isVariable** = true

13.45 LittleEndian< T > Struct Template Reference

Public Member Functions

- const T **operator()** (const T &x) const