Simulation Smackdown Environment Federate

Edwin Z. Crues
Simulation and Graphics Branch (ER7)
NASA Johnson Space Center
2101 NASA Parkway
Houston, Texas 77058

April 2013

Abstract

This document provides and overview of the SISO/SCS Simulation Smackdown Environment federate. The Environment federate provides three principal services to a Smackdown federation execution: control for time management of the federation execution, definition of the physical time standard and epoch of the federation execution, and the position and orientation of key planetary reference frames. This document also describes how to obtain, build and run the Environment federate.

1 Introduction

This document provides and overview of the SISO/SCS Simulation Smackdown Environment federate. The Environment federate provides three principal services to a Smackdown federation execution:

1. Control for time management of the federation execution,
2. Definition of the physical time standard and epoch of the federation execution, and
3. Position and orientation of key planetary reference frames.

This document also describes how to obtain, build and run the Environment federate.

2 What does the Environment federate do?

As stated in the introduction, the Environment federate has three principal responsibilities in the Smackdown federation execution: time regulation, time definition and planetary reference frame definition.

2.1 Time Regulation

The Environment federate is a time managed federate that is both time regulating and time constrained. It is time regulating in that it coordinates with the other time regulating federates in the Smackdown federation execution to determine if all the regulating federates agree that time can advance. The Environment federate is time constrained in that it will not advance in time until all the time regulating federates in the federation execution agree that time can advance. Time regulation and time constraint are achieved through the use of the High Level Architecture (HLA) time management APIs and semantics.

2.2 Physical Time

The Environment federate provides a physical time standard for the federation execution. This physical time standard is not necessarily the same as either the simulation time or the federation execution time. The physical time standard is based on a starting epoch. For the 2013 SISO/SCS Simulation Smackdown, the time standard is Terrestrial Time (TT). The starting epoch can be varied but by default is 10 April 2013 20:00:00 GMT or TJD=16392.83333333334.

2.3 Reference Frames

The Environment federate also provides state information for the planetary environment of interest to the federation execution. For now, that consists of the definition of the location and, in some cases, the orientation of the planetary bodies of interest. This information is expressed in the form of named reference frames.

The reference frame information is represented in a conceptual tree structure (see Figure 1). This means that every reference frame in the federation execution should have a parent reference frame (the frame in which it is expressed). The only exception is the root frame for the tree. In the case of the Environment federate, that will always be the solar system barycenter frame (`SolarSystemBarycentricInertial`).

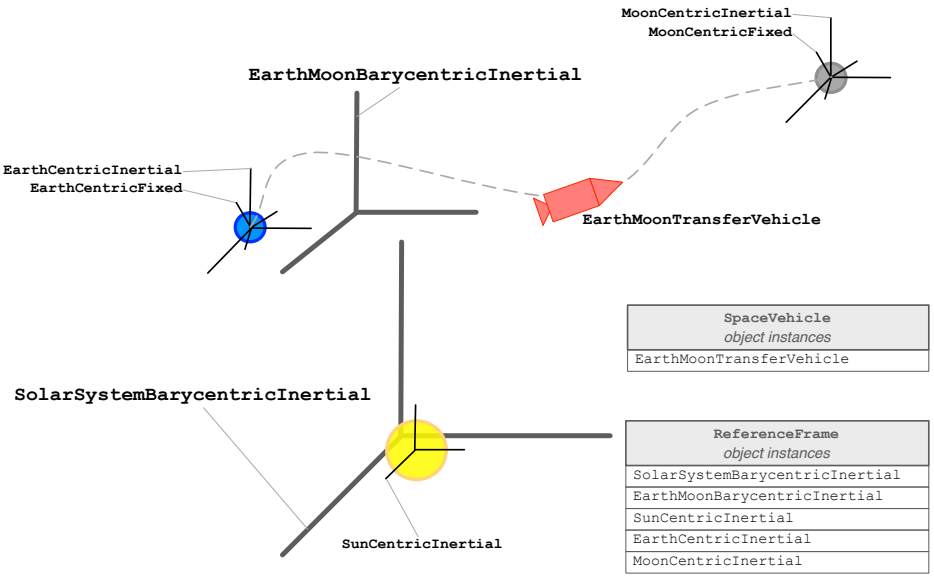

Figure 1: Environment federate reference frames

The Environment federate provides reference frames (state information) for the following planetary system locations:

`SunCentericInertial` The position of the solar systems central star, the Sun, with respect to the `SolarSystemBarycentricInertial` reference frame.[1]

`EarthMoonBarycentricInertial` A pseudo-inertial reference frame that is positioned at the Earth-Moon barycenter and expressed with respect to the `SolarSystemBarycentricInertial` reference frame.

`EarthMoonBarycentricRotating` A non-inertial rotating reference frame that is collocated with the `EarthMoonBarycentricInertial` frame but rotates with the Earth-Moon system and is expressed with respect to the `EarthMoonBarycentricInertial` reference frame.

`EarthCentricInertial` A pseudo-inertial reference frame that is positioned at the center of the Earth and is expressed with respect to the `EarthMoonBarycentricInertial` reference frame.

`EarthCentricFixed` A non-inertial reference frame that is positioned at the center of the Earth, rotates with the fixed surface of the Earth and is expressed with respect to the `EarthCentricInertial` reference frame.

[1]By definition, the `SolarSystemBarycentricInertial` reference frame is a null frame. It always has 0 position and 0 velocity as well as identity attitude and 0 angular velocity. So, this frame is never published since its state is implicitly known. It defines the base origin and attitude for all other frames.

`MoonCentricInertial` A pseudo-inertial reference frame that is positioned at the center of the Moon and is expressed with respect to the `EarthMoonBarycentricInertial` reference frame.

`MoonCentricFixed` A non-inertial reference frame that is positioned at the center of the Moon, rotates with the fixed surface of the Moon and is expressed with respect to the `MoonCentricInertial` reference frame.

`EarthMoonL2Rotating` A non-inertial reference frame that is positioned at the Earth-Moon Lagrange point L2, rotates with the Earth-Moon system and is expressed with respect to the `EarthMoonBarycentricInertial` reference frame.

`MarsCentricInertial` A pseudo-inertial reference frame that is positioned at the center of the Mars and is expressed with respect to the `SolarSystemBarycentricInertial` reference frame.

`MarsCentricFixed` A non-inertial reference frame that is positioned at the center of Mars, rotates with the fixed surface of Mars and is expressed with respect to the `MarsCentricInertial` reference frame.

Each reference frame consists of a collection of data that define the current state of the reference frame with respect to a parent reference frame (see Table 1). The `ReferenceFrameTranlation` and `ReferenceFrameRotation` are defined in Tables 2 and 3 in the Appendix.

The definition of all the objects, their attributes and the constituent elements of the attributes are described in the Smackdown 2013 Federation Object Model (FOM) modules. The Environment federate references two specific FOM modules: `SISO_Smackdown_2013_core.xml` and `SISO_Smackdown_2013_environment.xml`. These modules are contained in the Environment federation distribution under the `FOMs` directory (see Section 3).

3 Getting the Environment federate

You can obtain the Environment federate code in its entirety from the Simulation Smackdown Assemble Subversion repository. I am assuming that you are familiar with Subversion and how to use the appropriate commands. Here are the steps for getting access to the repository:

1. Register for an account at the Assembla website (`https://www.assembla.com/user/signup`)
2. Contact the Smackdown Assembla repository custodian (`edwin.z.crues@nasa.gov`) and provide your Assembla account name.
3. Once the custodian adds you to the Smackdown team, you can check out the Environment federate from the repository (`https://subversion.assembla.com/svn/SISO_Smackdown/trun`

Once you have checked out the code, you should have a top level directory `2013`. Once you enter the top level directory, you should see the following top level directory structure:

Field	Type	Description
name	`HLAunicodeString`	A unique name for this reference frame instance. Reference frame names are essential in forming 'links' between parent/child reference frames.
parent_name	`HLAunicodeString`	The name of this frame's parent reference frame. If this frame has no parent (i.e., is a 'root' reference frame), then this string must be empty, otherwise the non-empty string must correspond to the name attribute of some other ReferenceFrame object instance in the simulation.
translational_state	`ReferenceFrameTranslation`	This reference frame's translational state with respect to its parent frame. If this frame has no parent, this attribute is meaningless.
rotational_state	`ReferenceFrameRotation`	This reference frame's rotational state with respect to its parent frame. If this frame has no parent, this attribute is meaningless.
time	`Time`	This value serves as a 'time stamp' that specifies the simulated time (TT) to which the attributes values correspond. It may be used by federates that do not use HLA time management but still need to know when the attributes were valid. (E.g., a plotting federate that isn't time regulating or time constrained would need the time stamp in order to plot time series.)

Table 1: Constituent parts of a reference frame definition

```
2013/
   FOMs/
   bin/
   data/
   env/
   federates/
   models/
```

The Environment federate code is in the `federates` directory. Once you enter the `federates` directory, you should see the `Environment` directory. This is the top level directory for building the Environment federate. Once in the `Environment` directory you should see the following directory structure:

```
Environment/
   Manifest.txt
   build.xml
   docs/
   jat/
   lib/
   makefile
   run_env*
   run_test*
   src/
```

At this point you should be able to proceed to building the federate.

4 Building the Environment federate

The Environment federate is written in the Java programming language. I am going to assume that you are already familiar with Java coding and compiling. You can build the Environment federate with either the `makefile` or the Ant `build.xml` script. Choose the means you are most comfortable with.

To build the code, you will need the following:

- An installed Java Development Kit (JDK).
- An installed HLA 1516-2010 Run Time Infrastructure (RTI) Java JAR file.
- A link file named `rti1516e.jar` in the `lib` directory that points to your vendor's RTI JAR file.
- Know how to configure either the `make` or `ant` build systems to find these.
- Run either `make` or `ant` to build the Environment federate JAR file.

A successful build should result in an `Environment.jar` file in the `lib` directory.

5 Running the Environment federate

In this section we cover running the Environment federate. I am going to make the assumption that you are already familiar with HLA and the specifics of the Run Time Infrastructure (RTI) that you are using.

5.1 Setting up the environment

If you are running the Pitch RTI, I don't know of any environment variables that need to be set. If you are running the MÄK RTI, you will need to set the `LD_LIBRARY_PATH` environment variable to include the path to the RTI runt time libraries as will as the Java interface libraries. This is what mine looks like:

```
LD_LIBRARY_PATH = /opt/rti/makRti4.2/lib/java:/opt/rti/makRti4.2/lib
```

5.2 Running the Environment federate

Once the run time environment is set up, you should be able to run the Environment federate from the command line. It is possible to run the federate directly from the makefile with the command '`make run`'. However, I recommend using the `run_env` script. You should be able to see the the available options with the command `run_env -h`. If you do that, you should see the following:

```
*** Simulation Smackdown Environment Federate ***

usage: Environment [{-h,--help}]
                   [{-v,--verbose}]
                   [{-d,--date} <MM/dd/yyyy HH:mm:ss zzz>]
                   [{-j,--JD} UTC_Julian_date]
                   [{-m,--MJD} UTC_modified_Julian_date]
                   [{-t,--TJD} UTC_truncated_Julian_date]
                   [{-r,--run_time} seconds]
                   [{-f,--hla}]
                   [{-n,--name} federate_name]
                   [{--crc_host} CRC_host_name]
                   [{--crc_port} CRC_port_number]
```

Among the many options is the ability to set the initial date and time of the run using either Julian Date (JD), Modified Julian Date (MJD), or Truncated Julian Date (TJD). All of these are assumed to be in Coordinated Universal Time (UTC).

You can run the Environment federate with the command `./run_env`. If you do this and the RTI is running on the local host, you should see something like the following output:

```
*** Simulation Smackdown Environment Federate ***

Ephemeris file located in:
   /opt/smackdown/2013/federates/Environment/jat/data/core/ephemeris/DE405data/

***********************************************
RTI Name: pRTI 1516
RTI Version: v4.4.2
HLA Version: null
Federate "Simulation Smackdown Environment": Cannot advance to current time!
   This may be the first time regulating federate!

/opt/smackdown/2013/federates/Environment/jat/data//core/spacetime
***********************************************
CRC host: localhost
CRC port: 8989
***********************************************
Simulation Epoch: 2013,4,10 20:0:1.341104507446289E-5
Julian date: 2456393.3333333335
Truncated Julian date: 16392.83333333349
***********************************************

Executive Loop Counter: 39
```

The last line is the executive loop counter which should correspond to the number of seconds that the Environment federate has been running, in this case, 39 seconds.

You can see the Environment federate running in the RTI CRC interface in Figure 2.

Figure 2: Environment federate in RTI Interface

5.3 Testing the Environment federate

Once you have the Environment federate running, you should be able to test it with the Environment Test federate. It is possible to run the Environment Test federate directly from the makefile with the command '`make test`'. However, I recommend using the `run_test` script. You should be able to see the the available options with the command `run_env -h`.

I you run the Environment Test federate after the Environment federate is already started, you should see output like the following:

```
*** Java Environment Test Federate ***

************************************************
RTI Name: pRTI 1516
RTI Version: v4.4.2
HLA Version: null
************************************************
CRC host: localhost
CRC port: 8989
************************************************
EnvironmentTest "Environment Test": Starting time (epoch): 1.4163410391840134E9
************************************************
Times are:
   Executive Loop Counter: 0
   Simulation Execution Time: 0.0
   EnvironmentTest Physical Time: 1.4163410391840134E9
   Federation Execution Time (s): 175.0

Times are:
```

```
   Executive Loop Counter: 1
   Simulation Execution Time: 1.0
   EnvironmentTest Physical Time: 1.4163410401840134E9
   Federation Execution Time (s): 176.0

Times are:
   Executive Loop Counter: 2
   Simulation Execution Time: 2.0
   EnvironmentTest Physical Time: 1.4163410411840134E9
   Federation Execution Time (s): 177.0
```

You can see the Environment Text federate running in the RTI CRC interface in Figure 3.

Figure 3: Testing the Environment federate

6 Appendix

Field	Type	Description
position	`PostionVector`	Position of the subject frame origin with respect to the referent origin with components resolved onto the subject coordinate axes.
velocity	`VelocityVector`	Velocity of the subject frame origin with respect to its referent origin with components resolved onto the subject coordinate axes.

Table 2: Reference frame translational state definition

Field	Type	Description
attitude_quaternion	`AttitudeQuaternion`	Attitude quaternion that specifies the orientation of the subject frame with respect to the referent.
angular_velocity	`AngularVelocityVector`	Angular velocity of the subject frame with respect to the referent with components resolved onto the subject coordinate axes.

Table 3: Reference frame rotational state definition