Spatial Reference Model (SRM)
Timing Assessment Procedure

RELEASE / REVISION DATE:
v0.5b / 29 February 2008

ABSTRACT:
This procedure describes the steps to be taken in assessing the SRM timing performance in a consistent way by establishing an assessment method based on fixed set of input data and pre-established runtime routines and iteration steps. This document is intended for users to independently assess the SRM timing performance in their particular computing environment and capture the results of the assessment in the results section of this document.

Authors

Name	Role	Organization	E-mail / Phone
David Shen	SEDRIS Eng.	SEDRIS Project	david.t.shen@saic.com

Contributors

Name	Role	Organization	E-mail / Phone
Farid Mamaghani	SEDRIS Eng.	SEDRIS Project	farid@halcyon.com
Ralph Toms	SRM SME; SEDRIS Eng.	SRI; SEDRIS Project	ralph.toms@sri.com
Warren Macchi	SEDRIS Eng.	AccentGeographic; SEDRIS Project	wmacchi@accentgeographic.com
Les Habel	QA	FCS Training IPT	leslie.j.habel@saic.com
Rob Cox	Environment SME	FCS Training IPT	robert.m.cox@saic.com

Revision History

Version	Date	Description of Update
v0.1	26 Feb 2008	Initial draft for review.
v0.5	29 Feb 2008	Added clarification on SRM validation and data items to be captured for each SRM assessment experiment.

1 SRM Timing Assessment

1.1 Test Item:

The software under test is the SRM C/C++ SDK version 4.3. Only the C++ API implementation is currently included in the assessment procedure described in this document. The C API and Java API implementations of the SRM may be included in the future.

1.2 Description:

The assessment procedure described in this document is for the collection of timing metrics related to the execution of the SRM `changeCoordinateSRF` (coordinate conversion) operation applied to a pre-selected set of the celestiodetic (geodetic) coordinates representing various locations across the globe. The pre-selected set of 10 coordinates is stored in the input file, `timing_test_coord.csv`, whose content is as follows:

```
Test Points,      lon,        lat,    height
      #,          deg,        deg,      m

Coordinate 1, 126.9813889, 37.5141667, 10.0
Coordinate 2,  44.3922222, 33.3155556, 20.0
Coordinate 3, 151.2072222,-33.8672222, 30.0
Coordinate 4,  -0.1261111, 51.5002778, 40.0
Coordinate 5,  37.6175000, 55.7558333, 50.0
Coordinate 6, -46.6388889,-23.5488889, 60.0
Coordinate 7,-106.0208333, 32.9936111, 70.0
Coordinate 8, -85.9725000, 32.3591667, 80.0
Coordinate 9,-149.8997222, 61.2163889, 90.0
Coordinate 10,-78.5241667, -0.2294444, 0.0
```

SRM supports over 23 types of SRF templates and many other SRF sets and standardized SRF instances. This assessment only involves a subset of those SRFs, namely the Celetiocentric (aka, geocentric), the Celestiodetic (aka, geodetic), and the Augmented Universal Transverse Mercator (AUTM) SRFs. The Object Reference Model (ORM), which specifies the shape of the Earth, associated with the three SRFs under test is the World Geodetic System 1984 (WGS 84) ellipsoid.

The timing assessment application (herein called the Test Driver) invokes the SRM `changeCoordinateSRF` operation on the selected set of coordinates and performs a pair-wise conversion between two of the three SRFs under test. Each conversion is repeated 10 million times and the mean conversion time is computed for each test coordinate.

1.3 Software Information:

The software under test is SRM C++ SDK version 4.3. The Test Driver invokes the necessary SRM initialization and execution of operations for the SRM C++ coordinate conversion timing metrics collection.

1.4 Test Driver/Environment Information:

The Test Driver invokes the SRM C++ `changeCoordinateSRF` method to carry out the coordinate conversions of the coordinates stored in the input file `timing_test_coord.csv`. The output of the timing assessment is also stored in a .csv file. The Test Driver software was written in C++ and supports both WIN32 and Linux platforms, and can be built using the native SRM C/C++ SDK build environment with minimal setup. See the SRM C/C++ SDK documentation for instructions on how to compile and build an SRM application with the SRM API.

1.5 Test Runtime Environment:

Any WIN32 or Linux computing platform supported by the SRM C/C++ SDK can be used to run the SRM performance test. The platform characteristics should be recorded and kept with the test results. As a minimum, the following platform characteristics should be captured:

CPU.	(Example: 2.4 GHz Xeon / 533 Processor)
RAM.	(Example: 1.5 Gb DDR at 266 MHz)
Operating System.	(Example: Linux RedHat 8.0)
Compiler.	(Example: GCC v3.2.2)

1.6 Initial Conditions:

The computing platform where the test is to be conducted must not be running any other process that would significantly alter the timing results. In other words, no other process should be competing with the software under test for the computing resources. It is recommended that all other applications running on the platform be terminated, prior to running the SRM tests. This includes any background applications that may be part of the operating system functions that may preempt the CPU and force the Test Driver application to wait while the CPU responds to a different program.

1.7 Timing Assessment

1.7.1 Description

For the timing assessment, the Test Driver invokes the SRM `changeCoordinateSRF` operation on the selected set of coordinates and performs a pair-wise conversion between two of the three SRFs under test. Each conversion on a particular coordinate is repeated 10 million times and the mean conversion time is computed. In addition, minimum, maximum and mean conversion times for each combination of the pair-wise conversions of all the test coordinates are also computed at the end. The complete timing assessment is performed twice, once with the SRM coordinate validation function turned on and once without the SRM validation. When the SRM validation mode is on, the `changeCoordinateSRF` operation checks the input (source) coordinate before the actual conversion takes place and then checks the resulting output (target) coordinate to ensure they are within the bounds of the coordinate definitions. When the validation mode if off, such validations are skipped. Upon completion of the timing assessment execution, an output .csv file is created (the output file name is specified by the user). An example output .csv file is as follows:

```
Timing For One Coordinate Conversion With Validation (in micro-
seconds) – SRM C++ 4.3

Test coord,CD to UTM,UTM to CD,CC to UTM,UTM to CC,CD to CC,CC to
CD

Coordinate 1, 0.375, 0.547, 0.672, 0.781, 0.375, 0.437,
Coordinate 2, 0.39, 0.578, 0.672, 0.796, 0.391, 0.5,
Coordinate 3, 0.437, 0.562, 0.671, 0.766, 0.359, 0.422,
Coordinate 4, 0.422, 0.594, 0.703, 0.797, 0.359, 0.437,
Coordinate 5, 0.406, 0.531, 0.796, 0.75, 0.36, 0.422,
Coordinate 6, 0.406, 0.547, 0.687, 0.766, 0.36, 0.422,
Coordinate 7, 0.375, 0.547, 0.656, 0.781, 0.343, 0.422,
Coordinate 8, 0.391, 0.531, 0.657, 0.781, 0.359, 0.422,
Coordinate 9, 0.406, 0.579, 0.688, 0.813, 0.375, 0.422,
Coordinate 10, 0.407, 0.547, 0.718, 0.766, 0.344, 0.453,

MIN, 0.375, 0.531, 0.656, 0.75, 0.343, 0.422,
MAX, 0.437, 0.594, 0.796, 0.813, 0.391, 0.5,
MEAN, 0.4015, 0.5563, 0.692, 0.7797, 0.3625, 0.4359,


Timing for One Coordinate Conversion Without Validation (in
micro-seconds) – SRM C++ 4.3

Test coord,CD to UTM,UTM to CD,CC to UTM,UTM to CC,CD to CC,CC to
CD

Coordinate 1, 0.266, 0.437, 0.531, 0.625, 0.297, 0.375,
```

```
Coordinate 2, 0.265, 0.406, 0.531, 0.609, 0.297, 0.375,
Coordinate 3, 0.281, 0.422, 0.546, 0.578, 0.282, 0.375,
Coordinate 4, 0.297, 0.438, 0.594, 0.625, 0.296, 0.39,
Coordinate 5, 0.266, 0.422, 0.547, 0.61, 0.297, 0.375,
Coordinate 6, 0.265, 0.422, 0.547, 0.609, 0.297, 0.375,
Coordinate 7, 0.266, 0.421, 0.531, 0.625, 0.297, 0.359,
Coordinate 8, 0.266, 0.422, 0.532, 0.594, 0.296, 0.375,
Coordinate 9, 0.297, 0.453, 0.562, 0.625, 0.297, 0.375,
Coordinate 10, 0.281, 0.421, 0.594, 0.625, 0.281, 0.422,

MIN, 0.265, 0.406, 0.531, 0.578, 0.281, 0.359,
MAX, 0.297, 0.453, 0.594, 0.625, 0.297, 0.422,
MEAN, 0.275, 0.4264, 0.5515, 0.6125, 0.2937, 0.3796,
```

1.7.2 Test Procedure

This test procedure assumes that `srm_timing` (Test Driver) executable is built from the `srm_timing.cpp` source code and statically linked to the SRM C++ 4.3 library. The SRM C/C++ SDK 4.3 documentation can be consulted on how to build an executable using SRM. Note: the default build has the compiler optimization turned on.

No.	Step Description	Expected Result
1	Run the `srm_timing` executable with the following command line arguments: `srm_timing timing_test_coord.csv <out_results_file>.csv` (Note: Replace the *`<out_results_file>`* with the desired output file name)	Two messages will appear on the display indicating the beginning and the end of the test as follows: "Started SRM Timing…" and "Completed SRM Timing!" The result of the test will be saved in the `<out_results_file>.csv` file.

Note: The content of the output file (.csv) can be viewed in Excel.

1.7.3 Actual Timing Results

This section is intended for recording the relevant information associated with a particular timing assessment experiment including the actual results from that assessment.

Person who performed the assessment:

- Name:
- Affiliation:
- Phone #:
- E-mail:

Timing Assessment Execution Information:

- Date (mm/dd/yyyy):
- Time Started (hh:mm):
- Time Completed (hh:mm):

Timing Assessment computation environment:

- CPU:
- RAM:
- Operating System:
- Compiler:

Timing Assessment Results:

<Insert the timing results from the output file `<out_results_file>.csv` here>

Notes and/or Conclusions: