FWS Interface Specifications

1. DataBanker

1.1 Introduction

The Data Banker provides synchronized storage for sensor readings.

1.2 Interface Overview

1.2.1 Services Provided

Service	Provided By	Tested By
1. Initialize the set of stored sensor readings.	initialize	TC1, TC2, TC3, TC4, TC5
2. Store a new sensor reading, maintaining only the necessary history, and retrieve the current sensor reading history, keeping reads and writes synchronized.	read, write	TC1, TC2, TC3, TC4, TC5

1.2.2 Access Methods

Access Method	Parameter name	Parameter type	Description	Exceptions	Map to services
initialize	sensorType	String	Type of sensor.		1
write	sensorType:I r:I	String SensorReading	Type of sensor. Sensor reading value		2
read:O	sensorType:I :O	String Vector<SensorReading>	Type of sensor. Vector of elements of type SensorReading		2

1.2.3 Access Method Effects

Access Method	Description
initialize	Initializes a vector of elements of type *sensorType* of length *HistoryLength* for each sensor of *sensorType* with initial values of null
write	Adds the SensorReading r to the back of the queue and removes the oldest sensor reading value from the front of the queue.

read	Returns the vector of sensor readings of type *sensorType*. With the most recent values of the sensor readings. The vector is of length (HistoryLength * number of sensors) of that type.

Synchronization: This module supports concurrent access to the *read* and *write* methods. Where any read or write can occur concurrently, the read and write statements act as atomic operators (i.e., the user will see either the sequence *read.write* or the sequence *write.read*).

1.3 Local Types

Type	**Value Space**
HistoryLength	The number of sequential, past sensor values kept

1.4 Terms

1.5 Uses

Type	**Value Space**
SensorReading	A triple (r, v, w) where r is of type SensorReading.resolution, v is of type SensorReading.value, and w of type SensorReading.weight

1.6 Exception Dictionary

None

1.7 Test Cases

1.7.1 T1

Step	**Description**	**Input Type/Value**	**Expected Results**	**Service**	**Preamble**
1	Initialize	sensorType	Type of sensor.		1
2	read	sensorType	Returns vector of null values		2

1.7.2 T2

….

1.8 Design issues

1. Should we let the user read an empty vector of sensor readings after initialization, or just throw an exception?
A1. Yes. An empty vector should be treated just as any other.
A2. No. There are no valid values in an empty vector that can be averaged, so we should let he user know that the vector is empty by throwing the exception.
Resolution: Yes. We will check values during testing during testing to save space and CPU cycles.