TestBase

```
/**
* @brief
* enumeration for test result
* @remarks
* The following status can be returned and their meanings are
* PASS : executed and the result was pass.
* FAIL : executed and the reuslt was fail.
* FAILED_TO_RUN : got error(s) during execution.
* NOT_TESTED : not executed yet.
* OTHER : non-standard test-specific status.
*/
enum TestStatusEnum
{
PASS = 1,
FAIL,
FAILED_TO_RUN,
NOT_TESTED,
OTHER
};


/**
* @brief
* Basic initialization routine entry point
* @return
* Status of the operation
*/
virtual int init( void ) = 0;

/**
* @brief
* PreActions entry point
* @return
* Status of the operation
*/
virtual int PreActions( void ) = 0;

/**
* @brief
* PostActions entry point
* @return
* Status of the operation
*/
virtual int PostActions( void ) = 0;

/**
* @brief
* TestExec entry point
* @return
* Status of the operation
*/
virtual int TestExec( void ) = 0;

/**
* @brief
* Set a name for this Test object
* @param name [in]:
* Name for this test object
*/
virtual void setName( const OFCString& name ) = 0;


/**
* @brief
* Get a name of this Test object
```

```
* @return
* Name of this test object
*/
virtual const OFCString& getName( void ) const = 0;


/**
* @brief
* Specifies the test id.
*
* @param testId
* The test Id.
*/
virtual void setId(int testId) = 0;
/**
* @brief
* Retrieves the test id.
*
* @return
* The test Id.
*/
virtual int getId() const = 0;

@brief
* Get a name of test class which implements ITest
* @return
* Name of test class
*/
virtual const OFCString& getTestClassName( void ) const = 0;


* @brief
* Gets Pass/Fail status.
* @return
* Pass/Fail stauts as the result of the test execution.
* @remarks
* If the returned status is OTHER, getOtherSubStatus() should be
* called to get detailed specific fined-granule status.
*/
virtual TestStatusEnum getPassFailStatus() const = 0;

/**
* @brief
* Specifies the pattern list tree.
*
* @param patternTree
* The pattern list tree.
*/
virtual void addPatternList(PatternTree * const patternTree) = 0;

/**
* @brief
* Get the pattern lists that the ITest derived object is associated with.
* @param patLists [out]:
* array of pattern lists
*/
virtual void getPatternLists( PatternTreeArray_t& patLists ) = 0;

/**
* @brief
* Get the number of pattern lists specified to the ITest derived
object
* @return
* number of pattern lists
*/
virtual size_t getNumOfPatternLists() = 0;
```

```
/**
* @brief
* Specifies the test condition.
*
* @param testCondition
* The test condition to be set.
*/
virtual void addCondition(TestCondition * const testCondition) = 0;

/**
ITest_base.h Page 5
* @brief
* Get TestConditions that are associated with this ITest instance
* @param testConds [out]:
* array of TestConditions
*/
virtual void getConditions( TestConditionArray_t& testConds ) = 0;
/**
* @brief
* Get the number of TestConditions specified to the ITest derive
d object
* @return
* number of TestConditions
*/
virtual size_t getNumOfConditions() = 0;


State variables (we may not have them explicitly listed in order to be
"abstract"

String Name (or TestID);
Enum (or Int) ExecResult (comes from the TestEnumStatus enum)
Bin FailBin
Integer MultiSiteSerial (not sure about what this is)
```