HCTS XTF Software Function Specification

1. Introduction

1.1 Glossary

HCTS: Hardware Certification Test Suite
HCL: Hardware Compatibility List
IHV: Independent Hardware Vendors
UI: User Interface
XTF: Extensible Test Framework

1.2 Background

The Hardware Certification Test Suite (HCTS) is an application comprised of a set of tests which enable Sun Platform groups, IHVs, system manufacturers, system integrators and end users to certify their hardware for use with the Solaris Operating System on x86 Platforms. This program has been being available on Bigadmin (http://www.sun.com/bigadmin/hcl/hcts/) since 2000.

1.3 Overview

This document describes new enhancements implemented in the Extensible Test Framework (XTF) as compared to HCTS 3.0. The XTF serves as middle layer between the User Interfaces and the TestCases. The XTF is the core engine managing all of the operational systems of the HCTS. Diagram 2-1 illustrates the primary functions and interactions of the XTF.

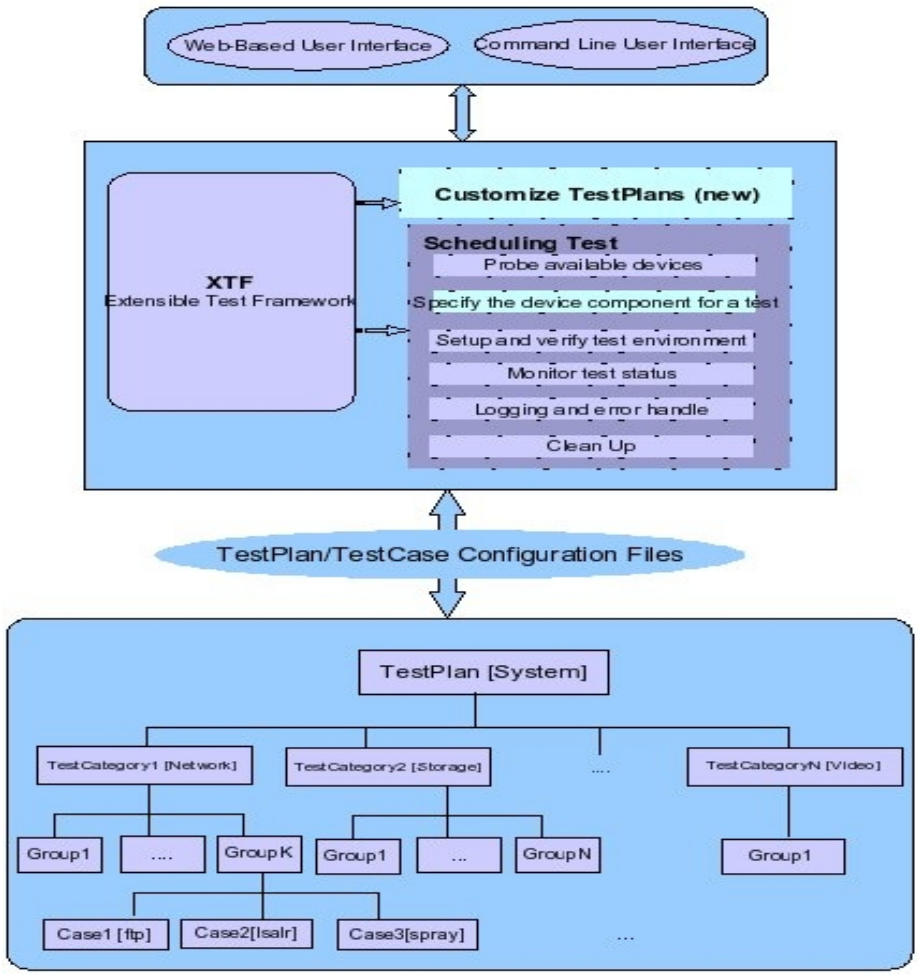
1.4 What's new

This project provides additional certification and debugging features to the XTF. The enhancement are as following:

- Addition of Custom TestPlan.
- Addition of Individual Controller Selection as part of Controller Certification.

2 Technical Description

2.1 Architecture Changes



***TestPlan**: Comprised of one or more TestCategories. TestCategories contained within a TestPlan are executed serially.*
***TestCategory:** Comprised of one or more Groups of TestCases. Groups contained within a TestCategory are executed serially.*
***TestCaseGroup:** Comprised of one or more TestCases which are executed in parallel.*
***TestCase**: The smallest executable test unit.*
***TestPlan/TestCase Descriptor Files**: Describe the organization of the TestCases and the TestPlan*

Diagram 2-1 HCTS Architecture

XTF stands for Extensible Test Framework. Most of the functions that XTF provides has been implemented in HCTS3.0. Two new functions which marked in green in Diagram 2-1 will be added to XTF. Custom TestPlan Mode will be an entirely new feature designed to enable users to implement a tailored testing procedure suited to specific diagnostic requirements. Device selection mode will provide more flexibility to a test . The XTF in the middle layer provides the APIs to UI layer and TestPlan/TestCase layer to support these function. See the interface changes for each layer in the following section.

2.2 XTF enhancements

Based on those changes described in 2.1 and 2.2, XTF implements the enhancement as following.

2.2.1 Add Customized test mode

HCTS 3.0 permits the execution of individual Unit Tests; however, only one Unit Test may be executed at a time and the operation of the test can not be modified in any way. In order to permit a much greater degree of flexibility, this project introduces a customizable TestPlan generation feature.
The primary features of this mode are as follows:

- Users can create a custom TestPlan. TestCases may be defined to execute in parallel or in serial. Any number of individual TestCases or groups of TestCases can be defined.
- The controller on which the TestCase(s) will execute can be selected.
- Arguments defined by the TestCase develop can be modified by the user, e.g. runtime, data set size, iterations, etc.

As an example: The TestCase "spray" generates UDP traffic. In HCTS 3.0, if the user executes this Unit Test, testing would be conducted on all detected network port(s). The only possible way to make the test act as a custom background UDP traffic generator would be to modify HCTS code. The Custom TestPlan Mode will permit the selection of which ports this test will run on and how long the runtime will be. Other more intricate arguments will also be tunable, such as the number of TestCase instances, the number of UDP packets to be transmitted, packet size, and interval between groups of packets. In this way, the user can fairly easily specify the type of UDP traffic desired from a friendly User Interface. By combining such alterations with other customized TestCases the user will be able to create powerful debugging or failure reproduction tools.

2.2.2 Add selection mode for controller certification

In HCTS3.0, user can not choose specific controller for the controller certification. The controller certification will run test on all available controllers. With selection mode of controller certification, user has the option to decide which controller need to be certified.

2.3 Interface changes

2.3.1 Interface changes between Test/TestCase and XTF

The interface between TestPlan/TestCase layer and XTF layer consists three kinds of Descriptor files(.xml) – TestPlan Descriptor Files, TestCase Descriptor Files and the newly

added **TestTemplate Descriptor File**s.

- **TestPlan Descriptor File (.xml) changes**

  A TestPlan Descriptor File(.xml) describes the organization of the TestCases and the steps to be carried out before and after executing a TestPlan. Each TestPlan Descriptor file corresponds to an individual TestPlan. There are two elements changes in the file:

  (1) Hook changes

  The Hook element describes location of a program which will be executed in a TestPlan to implement a specific function.
  There are six types of Hooks which can be defined in a TestPlan Descriptor File. They are probe, **getDevices,** setTestArgs, setup, check and clean. The functions of these Hook types are defined as follows:

  - probe : Probes devices and collect device information.
  - **getDevices: Retrieves the hardware device tree information. (New)**
  - setTestArgs: Adjusts test argument(s) based on information of devices.
  - setup: Set up test environment to facilitate execution of an associated TestCategory.
  - check: Verify the test environment has been successfully established.
  - clean: Restores the pre-test environment.

  The getDevices hook is a newly added hook type. With this newly added getDevices hook, XTF can implement the individual Controller Selection as part of Controller Certification. The purpose is to retrieve applicable hardware device tree for each specified TestPlan. This information may be used to display system summary information as well as to facilitate the selection of individual controllers for testing. The probe and getDevices hook will be executed before users specify the TestPlan. User can check the hardware information and the valid controller/device which can be tested before test begin. The probe, setTestArgs, setup and check Hooks are executed before the TestCases are scheduled. The clean Hook is executed after the all TestCases have completed. Please refer to diagram 2-2 for the hooks work flow.

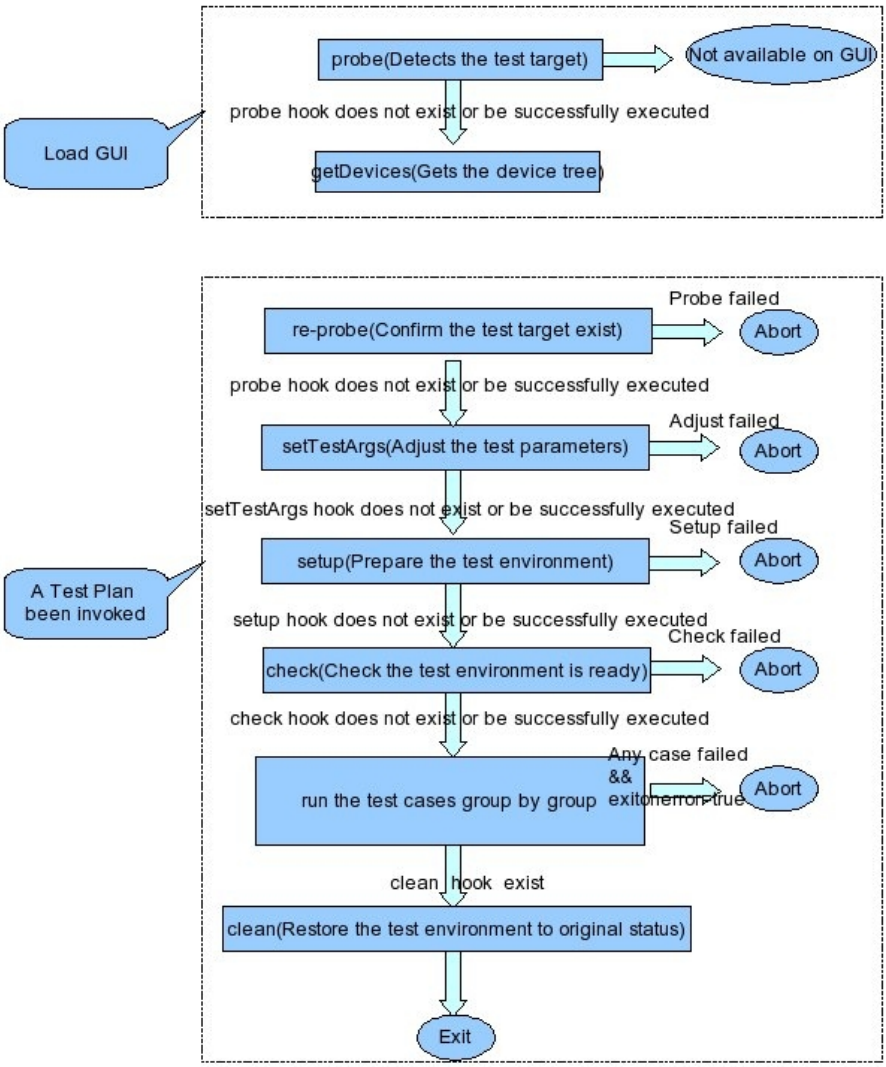


Diagram 2-2: TestPlan hooks' work flow

(2) Newly added TestCategory element

The TestCategory element is a newly added element in TestPlan Descriptor File. TestCases associated with a TestPlan may be divided into several categories, such as System TestPlan. Each category may be composed of several groups. The TestCases belonging to a group are executed in parallel.

A sample of a TestPlan Descriptor File(.xml) with newly added elements

```
<test name="system" spec="1.0" basedir="network" autoGroupId="false">
  <testcategory name="network">
    <testcasegroup id="1" autoCaseId="false">
      <testcase id="0" name="ftpstress" argument_auto_bind="false">
        <attribute>
          <name>instance</name>
          <value>1</value>
        </attribute>
      </testcase>
      ...
    </testcasegroup>
    ...
  </testcategory>
```

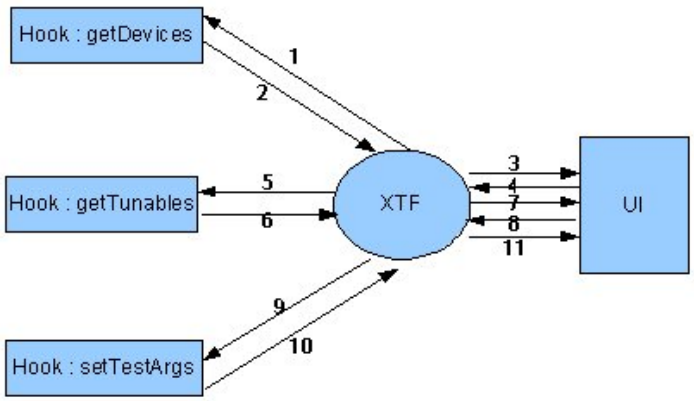
```
<testcategory name="storage">
   <testcasegroup id="1" autoCaseId="false">
     <testcase id="0" name="bonnie" argument_auto_bind="false">
        <attribute>
           <name>instance</name>
           <value>1</value>
        </attribute>
     </testcase>
     ...
  </testcasegroup>
   ...
</testcategory>
<hook>
     <name>probe</name>
     <executable>
        <command>sys_probe.ksh</command>
     </executable>
</hook>

<hook>
      <name>getDevices</name>
      <executable>
         <command>sys_fetch.ksh</command>
      </executable>
</hook>

...

<hook>
      <name>clean</name>
      <executable>
         <command>sys_clean.ksh</command>
      </executable>
 </hook>
</test>
```

- **TestCase Descriptor File (.xml) changes**
  Each TestCase Descriptor File corresponds to an individual TestCase. See the following hook element changes in the file compare to the previous version. There are four types of Hooks which can be defined in a TestCase Descriptor File. They are run, getTunables, getDevices, setTestArgs. Only the “run” hook is mandatory for a TestCase. The functions of these Hook types are defined as follows.

  - run(Required) : Defines where the actual TestCase program is located. As dictated by the generated schedule, the XTF will execute the program associated with this Hook.
  - **getTunables(Optional) : Retrieves list of user tunable options for the TestCase.(New)**
  - **getDevices(Optional) : Retrieves list of devices available to a TestPlan.(New)**
  - **setTestArgs(Optional) : Initializes the validation and setting/captures of**
    Tunable option data.
    Selected Device data.
    Testing environment based variables.(New)

  The 'getTunables', 'getDevices' and 'setTestArgs' are newly added hook types. The purpose of the 'getTunables' hook is to allow various TestCase arguments to be modified by a user, such as the runtime, the number iteration, etc. The 'getDevices' hook is very similar with the 'getTunables', but it only allow hardware based TestCase arguments, such as the controller or device, etc. If the 'getTunables' and

'getDevices' hooks are not defined, no augmentation options will be presented to the user. Please refer to diagram 2-3 for the hooks work flow.



Step1: XTF invokes the getDevices hook.
Step2: getDevices hook returns the devices list available of the TestCase to XTF.
Step3: XTF provides the devices list to UI.
Step4: User selects a device from the device list and UI returns the selection result to XTF. XTF sets this device as an argument to the TestCase.
Step5: XTF invokes the getTunables hook.
Step6: getTunables hook returns the tunable options that can be specified of the TestCase.
Step7: XTF provides the tunable options list to UI. UI prompt user to input.
Step8: User input. And UI passes the result of user input to XTF. XTF sets the arguments based on the input result.
Step9: The setTestArgs hook initializes the validation and setting/captures of
- Tunable option data.
- Selected Device data.
- Testing environment based variables.

Step10: XTF receives validation re-adjustments and makes appropriate data structure update.
Step11: XTF informs UI of accept/reject status of tunable options.

Diagram 2-3 TestCase hooks' work flow

A sample of a TestCase Descriptor File(.xml) with newly added elements.

```
<testcase name="audio" spec="1.0" basedir="audio">
  <attribute>
     <name>version</name>
     <value>1.0</value>
  </attribute>
  ...
  <argument>
      <name>RUNTIME</name>
      <type>env</type>
      <default-value>480</default-value>
  </argument>
  ...
  <hook>
     <name>run</name>
     <executable>
        <command>audio.stf</command>
     </executable>
  </hook><hook>
     <name>getTunables</name>
     <executable>
```

```
        <command>audio_interact</command>
      </executable>
    </hook><hook>
      <name>setTestArgs</name>
      <executable>
        <command>audio_adjust</command>
      </executable>
    </hook>
  </testcase>
```

2.3.2 Newly added Interface between Test/TestCase and XTF

- **New TestTemplate Descriptor File (.xml)**

  A TestTemplate Descriptor File is a subset of a TestPlan Descriptor File. All of the components remain the same except the TestCategory element all associated children elements are omitted. Creating a TestTemplate Descriptor File will enable developer created TestCategory and associated TestCases to appear in the Custom TestPlan segment of any HCTS UI. Please be note that only one TestCategory can be contained in a TestTemplate File.

With the newly added TestTemplate Descriptor File and the newly added TestCase hooks (getTunables, getDevices, setTestArgs), XTF provides the proper APIs to add TestCases to a TestTemplate and specify the test target for a TestCase and other tunable options. See the Diagram 2-4 for the customization process. TestTemplateManager loads a TestTemplate object by parsing the corresponding TestTemplate Descriptor File. The TestTemplate is the same with a TestPlan except only an empty TestCategory contained. User can add organize their own TestCategory. For example, user can add a group to the TestCategory, add a TestCase to the added group and specify the test target and other tunable options for the TestCase. After customization, a TestTemplate is upgraded to a normal TestPlan. XTF treat and execute it as a normal TestPlan.

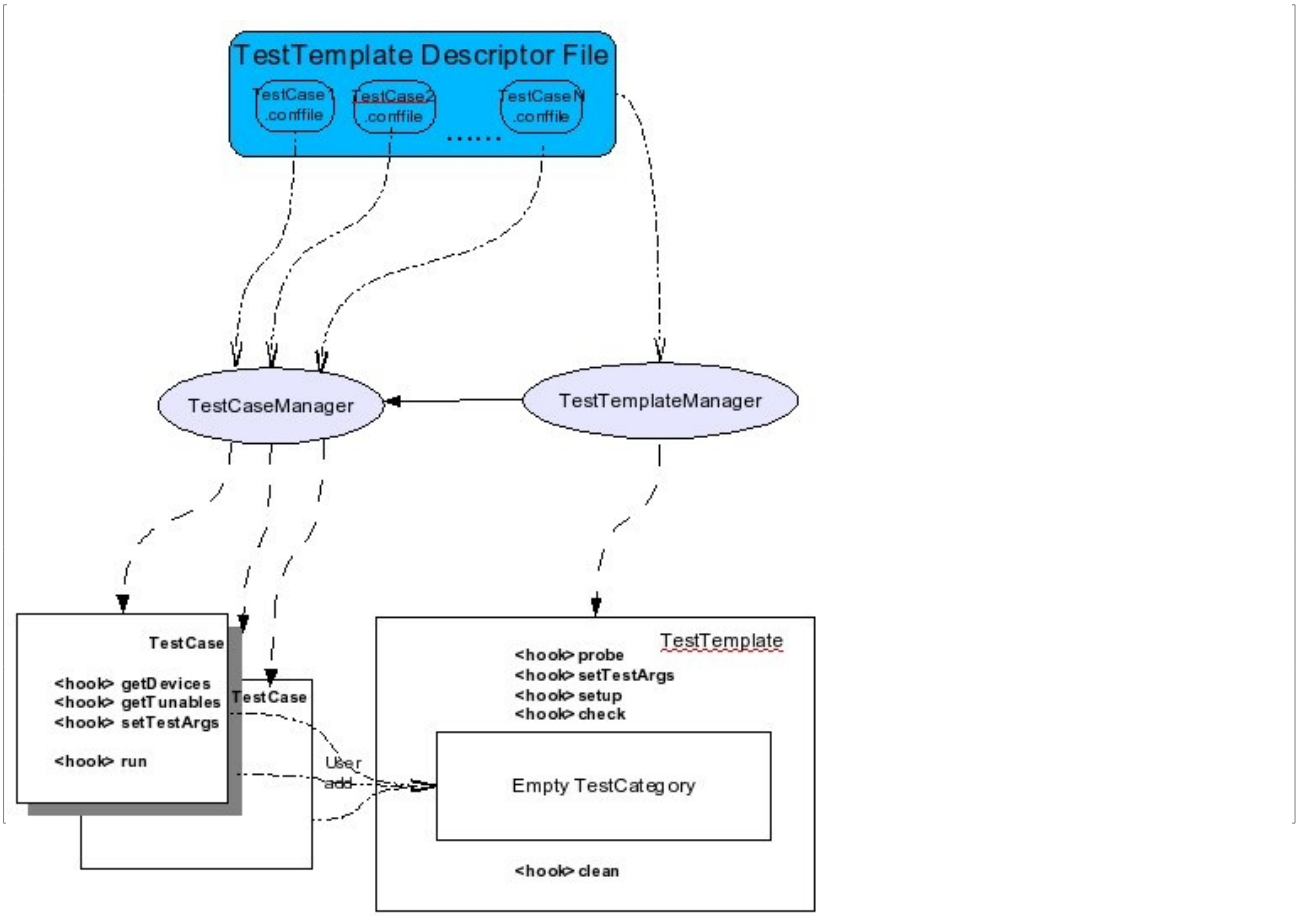


Diagram 2-4 Customization Process

Table 2-1 Glossary for Diagram 2-4

2.3.3 New Interface for UIs

In order to implement the test customization modular and device selection modular, XTF will also provide the proper APIs for UIs. More details refer to the XTF JavaDoc.

2.4 Interfaces

2.4.1 Exported Interfaces

Note: Only the changed interfaces are listed.

Interface Name	Proposed Stability Classification	Specified in What Document?	Former Stability Classification or Other Comments
TestPlan Descriptor File	Committed	Test Developer's Guide	Refer to section 2.3.1 for TestPlan Descriptor File. The file should be located at /opt/SUNWhcts/tests directory.
TestCase Descriptor File	Committed	Test Developer's Guide	Refer to section 2.3.1 for TestPlan Descriptor File. The file should be located

			at /opt/SUNWhcts/testcases directory.
TestTemplate Descriptor File	private	Design doc of XTF	Refer to section 2.3.2
/opt/SUNWhcts/templates	private	Design doc of XTF	The directory of TestTemplate Descriptor Files.
New XTF APIs for UIs	private	JavaDoc of XTF	

Table 2-2 Exported Interfaces

3 References

1. HCTS3.0 PSARC materials
2. HCTS Test Developer's Guide

Appendix A.1 HCTS XTF Scheduler

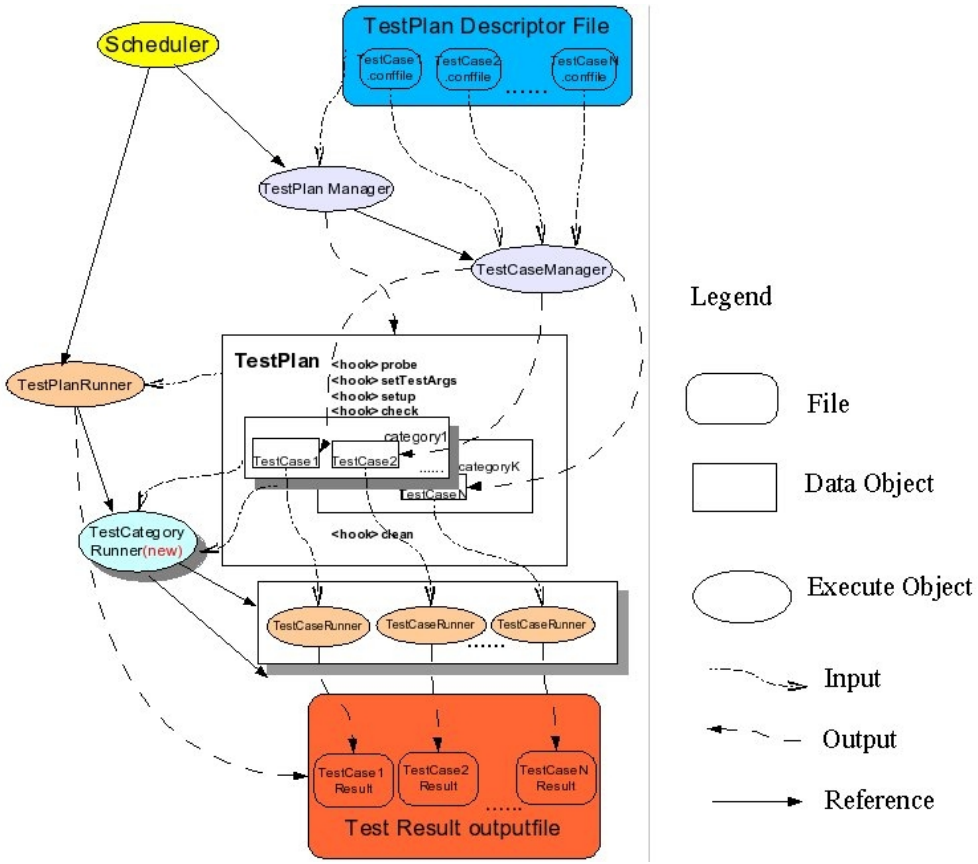


Diagram A-1: HCTS XTF Scheduler

Name	Description
TestCase Descriptor File	Contains definitions of attributes and arguments related to a TestCase.
TestPlan Descriptor File	Contains definitions of attributes, arguments, TestCategories and hooks related to a TestPlan. Valid TestPlan Hooks: probe, getDevices, setTestArgs, setup, check and clean.
Scheduler	Object : Controls the execution of TestCategories and TestCases.
Hook	Object : Responsible for executing a program in a TestPlan/TestCase running cycle.
TestCase	The smallest unit that can be executed by the XTF. Ideally, a TestCase focuses on the testing of a single function or a small group of functions related to one type of hardware. The **TestCase Object** in Diagram A-1 includes many Attributes, Arguments, and Hooks. This object is passed to TestCase Runner and TestCase Runner is responsible for executing it.
TestPlan	Composed of several TestCategories. Each TestCategory is composed of groups of associated TestCases to be executed in parallel. The **TestPlan Object** in Diagram A-1 includes many Attributes, Arguments, TestCategories and Hooks. The TestCases in the TestCase Group can share arguments defined by the TestPlan. The TestPlan object is passed to TestPlan Runner and TestPlan Runner is responsible for executing it.
TestPlan Runner	Object : Responsible for running a TestPlan. Each TestPlan Runner corresponds to one TestPlan. This object generates a TestCategory Runner for each TestCategory defined in the TestPlan and executes the TestCategory Runners in serial.
TestCase Manager	Object : Responsible for TestCase management. A TestCase Manager parses the TestCase Descriptor Files and creates a TestCase object.
TestPlan Manager	Object : Responsible for TestPlan management. A TestPlan Manager parses a TestPlan Descriptor File and creates a TestPlan object.
TestCase Runner	Object : Responsible for running a TestCase. Each TestCase Runner corresponds to one TestCase.
TestCategory Runner	Object : Responsible for running a TestCategory. Each TestCategory Runner corresponds to one TestCategory object. This object generates a TestCase Runner for each TestCase defined in the TestCase Group and executes the TestCase Runners in parallel.

Table A-1 Glossary for Diagram A-1: HCTS Scheduler