NIST/ITL's Biometric Application Programming Interface (BioAPI) Conformance Test Suite (CTS) Implementation (the BioAPI Test Environment)

Beta Implementation V1.1
February 28, 2006

Overview

National Institute of Standards and Technology (NIST)
Information Technology Laboratory (ITL)
Computer Security Division (CSD)

Fernando Podio
NIST/ITL CSD

Mark Jerde (NIST Guest Researcher)
The Biometric Foundation

Table of Contents

Acknowledgements

NIST/ITL's Biometric Application Programming Interface (BioAPI) Conformance Test Suite (CTS) beta implementation (also known as the BioAPI Test Environment (BTE)) was developed to help users verify the conformance of Biometric Service Providers (BSPs) to American National Standard InterNational Committee for Information Technology Standard (ANSI INCITS) 358-2002, the BioAPI Specification 1.1. The initial version of the BioAPI specification was developed by the BioAPI Consortium. The BioAPI standard defines generic interfaces to a broad range of biometric technologies.

NIST/ITL BioAPI CTS beta implementation is based on the fourth draft of a standards conformance testing methodology for ANSI INCITS 358-2002. The conformance testing methodology standard is under development in INCITS Technical Committee M1 – Biometrics. The fourth draft (M1/06-0073) can be downloaded from the NIST/ITL BioAPI CTS web site. The INCITS M1 web site is: http://www.incits.org/tc_home/m1.htm.

The International Organization for Standardization (ISO) / International Electrotechnical Commission (IEC) Joint Technical Committee 1 (JTC 1) Subcommittee 37 (SC 37) completed the development of an international version of the BioAPI specification (recently approved) and is also developing the associated conformance testing methodology standard.

NIST/ITL and DoD's Biometric Management Office co-sponsored the development of the conformance testing methodology standard under development in INCITS M1 together with the National Biometric Security Project (NBSP), Saflink Corp. and the Biometric Foundation.

DoD BMO independently developed a similar BioAPI CTS. NIST and DoD BMO conducted intensive testing of the CTS implementations to cross-validate the test results to ensure that these testing tools would derive the same results while testing the same products. Thanks are due to DoD's Biometric Management Office (BMO) for its close collaboration with NIST/ITL during the cross-validation tests and for its significant contributions to the draft conformance testing methodology standard under development.

Special thanks are also due to the National Biometric Security Project (NBSP) for co-sponsoring the development of NIST/ITL BioAPI CTS implementation through its collaborative agreement with Saflink Corp. and also for supporting The Biometric Foundation who collaborated with NIST/ITL in performing the required tests.

Abstract

This document provides a background on the NIST Biometrics Standards program. It refers to American National Standard InterNational Committee for Information Technology Standards (ANSI INCITS) 358-2002, the BioAPI specification and the BioAPI conformance testing methodology standard under development in INCITS Technical Committee M1 – Biometrics. It discusses the need for conformity assessment efforts in support of the BioAPI standard and other biometric interoperability and data interchange standards.

The document provides an overview of NIST/ITL's BioAPI Conformance Test Suite, its development history, its overall architecture and a description of its components. A brief reference to the organizations involved in the development and testing of NIST/ITL BioAPI CTS is also included.

Preface

NIST/ITL's Biometric Application Programming Interface (BioAPI) Conformance Test Suite (CTS) Implementation (the BioAPI Test Environment)

Beta Implementation
V1.0 - January 4, 2006

Biometric technologies are crucial components of secure personal identification and verification systems. Although for many years biometric technologies have been used mainly in law enforcement, they can be now found in all levels of government functions, in national defense applications and in commercial fields ranging from financial transactions to visitor authentication in amusement parks. World events in the last few years have further increased global interest in highly secure personal authentication using biometrics. National security priorities have led to the use of biometrics in machine readable travel documents, employee identification badges, and other secure applications. Deploying new information technology systems for government and commercial applications require both national and international consensus standards for biometrics. NIST has been a major contributor to the development of measurements, standards, and tests for biometrics for many years. Areas of investigation include fingerprints, face recognition, iris recognition and speech recognition. NIST supports the development of voluntary industry standards and the development of conformance tests, reference implementations, and evaluation procedures to facilitate the implementation of standards in biometric products.

Responding to legislative, government and market requirements for open-system standards, NIST is supporting the acceleration of the development of formal national and international biometric standards and associated conformity assessment. In order to achieve this goal, NIST works in coordination with other government agencies, industry, academic institutions and other research organizations. NIST strongly supports national and international standards organizations as the best environments for the development of voluntary consensus standards for biometric technology and the deployment of standards-based solutions.

NIST's Biometrics Standards Program, provides leadership to the national and the international biometric standards development bodies, including committee officers and technical editors and also provides technical expertise for critical standards development projects.

Background

Conformance Testing Tools in Support of Biometric Standards

Standards-based, high quality conformance testing tools help both developers and users by validating conformance claims, leading to greatly increased levels of confidence in products. Testing can also help ensure interoperability between standards-based products and systems. The implementation of NIST/ITL's Conformance Test Suite for the Biometric Application Programming Interface (BioAPI) also known as the "Biometric Test Environment (BTE)" was promoted and developed in support of users requiring or planning to require conformance to the BioAPI specification. It supports product developers interested in developing products conforming to voluntary consensus biometric standards by using the same test tools available to users and also supports the possible establishment of conformity assessment programs to validate conformance to the BioAPI standard and other emerging standards.

The BioAPI Specification

American National Standard INCITS 358-2002, the BioAPI Specification, is a biometric API standard that defines a generic way of interfacing to a broad range of biometric technologies. This biometric API can work with any type of biometric application. The standard API allows for easy substitution of biometric technologies, the use of biometric technology across multiple applications and easy integration of multiple biometrics using the same interface.

The original version of the BioAPI specification was developed by the BioAPI Consortium and completed in March 2001. It was approved by the InterNational Committee for Information Technology Standards (INCITS) as an American National Standard in February 2002 (ANSI INCITS 358-2002). The development of the international version (v 2.0) has been completed by the International Organization for Standardization (ISO) / International Electrotechnical Commission (IEC) Joint Technical Committee 1 Subcommittee 37 – Biometrics and it is expected to become an ISO standard early in 2006. The associated conformance testing methodology standard is under development in ISO/IEC JTC 1 SC 37.

Previous BioAPI-related implementations developed in support of the national version of the BioAPI specification (developed and funded by their sponsors) include the Win32 (Framework Reference Implementation Ver 1.1, 2000) developed by Intel, SAFLINK, IriScan, and Mytec Technologies Inc., the Linux Reference Implementation – developed by NIST, the Unix/Solaris Reference Implementation – developed by IBG, the WinCE Reference Implementation – developed by Saflink and sponsored by NBSP and a JNI Wrapper – developed by GenSoft.

Several government agencies and programs require conformance to BioAPI such as Department of Homeland Security TWIC program and Registered Traveler. BioAPI (1.1) is also included in DoD IT Standards Registry (DISR) and biometric application profiles approved by INCITS or under development in INCITS M1.

Conformance Testing Methodology Standard for the BioAPI Specification

NIST/ITL co-sponsored with other INCITS M1 member organizations a standards development project within the InterNational Committee for Information Technology Standards (INCITS) Technical Commmittee M1 – Biometrics for the development of a Conformance Testing Methodology for ANSI INCITS 358-2002, BioAPI Specification. Project co-sponsors are the Department of Defense (DoD) Biometrics Management Office (BMO), the National Biometric Security Project (NBSP), Saflink Corp., and The Biometric Foundation (TBF). NIST/ITL BioAPI CTS beta implementation is based on the fourth draft of the standard under development (document M1/06-0073).

This draft standard can be downloaded from the NIST/ITL BioAPI CTS web site www.nist.gov/biometrics/NISTITLBioAPICTS or through INCITS M1's Document Register at www.incits.org/tc_home/m1htm/docs/m1docreg.htm. Since this version of the NIST/ITL BioAPI CTS implementation is based on a standard under development, it is possible that changes to the standard may lead to changes in the CTS software.

NIST/ITL and the Department of Defense (DoD) Biometrics Management Office (BMO) have been working in close collaboration in the development of biometric standards and supporting testing tools. For over a year NIST and DoD BMO have been independently developing implementations of BioAPI test tools using concepts and principles specified in the draft conformance testing methodology standard under development in INCITS M1. NIST/ITL and DoD BMO conducted intensive testing of the initial versions of the test tools to cross-validate the test results using a number of vendor Biometric Service providers (BSPs) that claim their products conform to the BioAPI standard.

Development History

NIST/ITL BioAPI CTS beta implementation was developed in Java. It utilized a third-party Java Native Interface (JNI) component for interaction with the BioAPI Reference Implementation, which was written in C. Using the Java language allowed rapid development of a user-friendly GUI and efficient processing of XML. The JNI component was required because Java applications cannot invoke native C code unless specific JNI entry points have been defined in the code. The third-party JNI interface to BioAPI provided

access to only a small subset of the functions provided in the BioAPI specification. Since the source code for the third-party JNI interface was not available, development began on a new and more comprehensive JNI interface to BioAPI. This new JNI interface supports the BTE application and also provides a platform for other Java applications to use BioAPI, thus encouraging increased industry development of BioAPI compliant applications.

As development progressed, it became apparent that in certain instances, the behavior of native code invoked from within the Java Virtual Machine environment is not identical to the behavior of the same code when invoked from a native process. It was decided to mitigate the impact of this issue by retaining the existing Java components and adding a Native Execution Server component to the BTE. This component executes BioAPI functions and BioSPI functions from a native context, eliminating the restrictions of the Java Virtual Machine environment. Additionally, several modifications to the BioAPI Framework were implemented to support testing of error conditions that the framework is designed to prevent.

Architecture

Overall Architecture

The BioAPI Test Environment (BTE) is a Conformance Test Suite implementation composed of:

- a Java application that provides a Graphical User Interface (GUI) for selecting Biometric Service Providers to test, running tests, and displaying results.
- an Assertion Processor / Test Engine that processes XML assertions.
- a Java Native Interface Layer for interaction with native C code.
- a Native Execution Server for invoking BioAPI and BioSPI functions.
- a Native Execution Monitor for controlling the Native Execution Server
- a customized implementation of the BioAPI Framework.

These software tools were designed to test Biometric Service Providers for conformance to the American National Standard INCITS 358-2002, the BioAPI 1.1 specification.

Component Descriptions

Component:	**Graphical User Interface**
Language:	Java
Purpose:	Provide a user-friendly environment to support testing BSPs for conformance to the BioAPI Specification.

Description:	At startup the GUI populates the list of assertions by examining the assertion files stored on disk. After the user selects one or more assertions to run, the GUI invokes the Assertion Processor / Test Engine for each assertion selected by the user and displays the results. As the Assertion Processor completes each assertion, the GUI reads the log and displays the test result to the user. Each invocation of the Assertion Processor executes in a separate Java Virtual Machine to prevent assertions from interfering with each other or the GUI.
Package:	com.saflink.BSPTest
Key Classes:	TestPanel, LogDialog, TimeoutDialog, CheckNode, CheckRenderer

Component:	**Assertion Processor / Test Engine**
Language:	Java
Purpose:	Execute assertions and generate log files containing the test results.
Interface:	Command line parameters. Disk files.
Description:	This component opens the XML file that defines the assertion, parses the assertion language, and invokes the related commands. The test results are stored in a file on disk for the GUI to retrieve.
Package:	com.saflink.BSPTest
Key Classes:	RunAssertion, TestEngine, BSPInterface, XMLLog

Component:	**Java BioAPI Interface**
Language:	Java
Purpose:	Provide a Java interface for BioAPI and BioSPI commands.
Interface:	Java methods
Description:	The Java BioAPI interface provides Java methods and data types for invoking BioAPI and BioSPI functions. The Win32 class encapsulates the Java code used to invoke native C functions. The classes in this interface also perform initial translations of parameters and data types.
Package:	com.saflink.bioapi
Key Classes:	Win32

Component:	**Java Native Interface Layer**
Language:	C
Purpose:	Translate Java method calls to C function calls.
Interface:	Native C functions based on Java method signatures
Description:	The JNI Layer implements special native C functions that are callable from Java. The JNI Layer translates the parameters from Java data types to the appropriate C data types, and uses shared memory to transmit the commands and parameters to the Native Execution Server. Once the Native Execution Server finishes performing the commands, the JNI Layer translates the results from C data types back to Java types and returns them to the Java BioAPI Interface.
Project:	h_layer
Key Source Files:	bioapi_jni.c, bioapi_memfile.c

Component:	**Customized BioAPI Framework**
Language:	C/C++
Purpose:	Provide BioAPI Framework functionality without preventing testing of BSP's.
Interface:	C functions
Description:	The Customized Framework is based on the BioAPI Reference Framework and the majority of the code is identical. The Customized Framework includes modifications to allow incorrect parameters to pass through the Framework to the BSP. These modifications enable the BTE to test error conditions that the original Reference Framework is designed to prevent.
Project:	h_layer
Key Source Files:	bioapi_api.c, addmgr.c

Component:	**Native Execution Server**
Language:	C
Purpose:	Execute BioAPI / BioSPI Functions.
Interface:	Shared Memory
Description:	The Native Execution Server executes BioAPI / BioSPI functions. The Native Execution Server is a separate process that executes BioAPI and BioSPI functions. The

	Native Execution Server allows BioAPI and BioSPI functions to execute without the memory restrictions that exist when a native function is invoked from within a Java Virtual Machine. Once BioAPI / BioSPI functions complete, the results are returned to the JNI Layer.
Project:	BTEServer
Key Source Files:	BTEServer.cpp

Component:	**Native Execution Monitor**
Language:	C
Purpose:	Start and monitor the Native Execution Server and restart it as needed.
Description:	The Native Execution Monitor starts the Native Execution Server and restarts it when an assertion completes. This ensures that each assertion runs in a clean execution context.
Project:	BTEMonitor
Key Source Files:	BTEMonitor.cpp

Graphical User Interface and Assertion Processor / Test Engine

The Java package com.saflink.BSPTest contains the GUI code for the application. It also contains code for interpreting assertion files and generating test logs.

Java package: com.saflink.BSPTest

Class	**Description**
StartTest	Main entry point for the application.
TestPanel	Application GUI. Provides controls for the user to enter selections and run assertions. Invokes assertions via the CommandRunner class. Reads log files and displays results to the user.
CommandRunner	Runs a single assertion in a separate Java Virtual Machine.
RunAssertion	Main entry point and command-line parser for execution of a single assertion.
TestEngine	Assertion executor. Invokes BioAPI and BioSPI functionality

	via the BSPInterface class.
SwingWorker	Utility class for starting a new application thread.
PackageLog	Data structure for recording the results of an assertion.
ActivityLog	Data structure for recording the results of an activity.
XMLLog	Log writer for temporary files.
InputLog	Data structure for recording the values of input variables.
ReturnLog	Data structure for recording the return values of functions.
BSPInterface	Encapsulates the functionality of the com.saflink.bioapi package for use by classes in the com.saflink.BSPTest package. This module reduces the impact of any changes to the com.saflink.bioapi package on the com.saflink.BSPTest package.
CheckNode	Represents a node within a tree control.
CheckRenderer	Renders a node within a tree control.
FunctionLog	Data structure for recording details about a function call.
LogDialog	Displays the XML log.
TimeoutDialog	Displays the settings for capture timeout, verify timeout, etc. and allows the user to update the values.
test	Supports testing and debugging.

Java BioAPI Interface

The Java package com.saflink.bioapi provides a Java representation of BioAPI / BioSPI functions and the required data types.

Java package: com.saflink.bioapi

Class	**Description**
BioAPIData	Java equivalent of BioAPI_DATA.
BioAPIRuntimeException	Represents runtime exceptions generated in native code.
Bir	Java equivalent of BioAPI_BIR.
BirBiometricDataFormat	Java equivalent of BIR_BIOMETRIC_DATA_FORMAT.
BirDataTypes	Defines constants for raw, intermediate, processed, signed, and encrypted data types.

BirHandle	Java equivalent of BioAPI_BIR_HANDLE.
BirHeader	Java equivalent of BioAPI_BIR_HEADER.
BirPurposes	Defines constants for verify, identify, enroll, enroll for verification, enroll for identification, and audit purposes.
BspSchema	Java equivalent of BioAPI_BSP_SCHEMA.
ByteArrayHolder	Data structure for storing a byte array.
Errors	Defines constants for BioAPI error conditions.
IModuleEventHandler	Defines an interface for objects to implement in order to receive notification of BioAPI module events.
InputBir	Java equivalent of BioAPI_INPUT_BIR.
InputBirForms	Defines constants for database, full BIR, and BIR handle input.
ModuleEvent	Java equivalent of BioAPI_MODULE_EVENT.
ModuleHandle	Java equivalent of BioAPI_HANDLE.
Result	Java equivalent of BioAPI_RETURN.
Version	Java equivalent of BioAPI_VERSION.
Win32	Defines Java equivalents for BioAPI and BioSPI function calls. Invokes JNI entry points defined in a customized version of bioapi100.dll.

Customized BioAPI Framework

The BioAPI Framework included with BTE is based on the Reference Implementation. However it includes a number of customizations. The following table summarizes the changes that were made to the BioAPI Framework Reference Implementation. A JNI Layer has been added to support invocation from Java. The framework code has also been modified to allow testing of error conditions that the framework is designed to prevent.

Visual Studio Project: h_layer, Output: bioapi100.dll

Source File	**Description**
com_saflink_bioapi_Win32.h (new)	Declares JNI entry points for invocation from Java code. This header is generated from the Win32 class using the javah.exe utility. It defines the entry points used by the Win32 class.
bioapi_jni.h (new)	Declares utility functions implemented in

	bioapi_jni.c that are not JNI entry points.
bioapi_jni.c (new)	Implements JNI entry points and associated utility functions. This module converts the Java datatypes from JNI function calls into their native equivalents. After it converts the parameters are they are written to shared memory for execution by an independent native process (BTEServer.exe).
bioapi_memfile.h (new)	Declares constants and functions associated with transferring BioAPI / BioSPI parameters to and from shared memory.
bioapi_memfile.c (new)	Implements functions for transferring BioAPI / BioSPI parameters to and from shared memory.
biospi_test.h (new)	Declares variables and functions for managing UUID's and ModuleHandles.
biospi_test.c (new)	Implements functions for managing UUIDs and ModuleHandles. These functions allow the Framework to use correct values internally, while presenting invalid values to BSPs as needed.
addmgr.h (modified)	Declares functions for managing BSPs. The declaration for bioapi_CleanInternalAttachRecord has been updated to return BioAPI_RETURN instead of void.
addmgr.c (modified)	Implements functions for managing BSPs, including loading, unloading, attaching, and detaching BSPs. The custom version of this module allows the BTE to pass invalid ModuleHandles and UUIDs to BSPs. It also allows the BTE to perform multiple invocations of BioSPI_ModuleUnload for the same BSP. The function bioapi_CleanInternalAttachRecord has been updated to return the result code generated by the call to the BSP's BioSPI_ModuleDetach function.
bioapi_api.c (modified)	Implements functions for invoking biometric functions in BSPs. The custom version of this module allows the BTE to pass invalid ModuleHandles to BSPs.

Native Execution Server

The Native Execution Server (BTEServer.exe) invokes BioAPI and BioSPI functions when requests arrive via a shared memory area. Once the module receives a request, it invokes the requested function using the parameters submitted, and returns the results to shared memory. This component allows the BTE to execute BioAPI and BioSPI calls from a pure native process without the memory constraints of a Java Virtual Machine.

Visual Studio Project: BTEServer, Output: BTEServer.exe

Source File	Description
BTEServer.cpp	Implements functions to read BioAPI / BioSPI parameters from shared memory, invoke the corresponding functions, and return the results to shared memory.

Native Execution Monitor

The Native Execution Monitor starts the Native Execution Server and restarts it each time an assertion completes. This ensures that assertions that terminate abnormally or invalidate memory structures do not alter the results of subsequent assertions.

Visual Studio Project: BTEMonitor, Output: BTEMonitor.exe

Source File	Description
BTEMonitor.cpp	Starts the Native Execution Server and restarts it when an assertion completes.

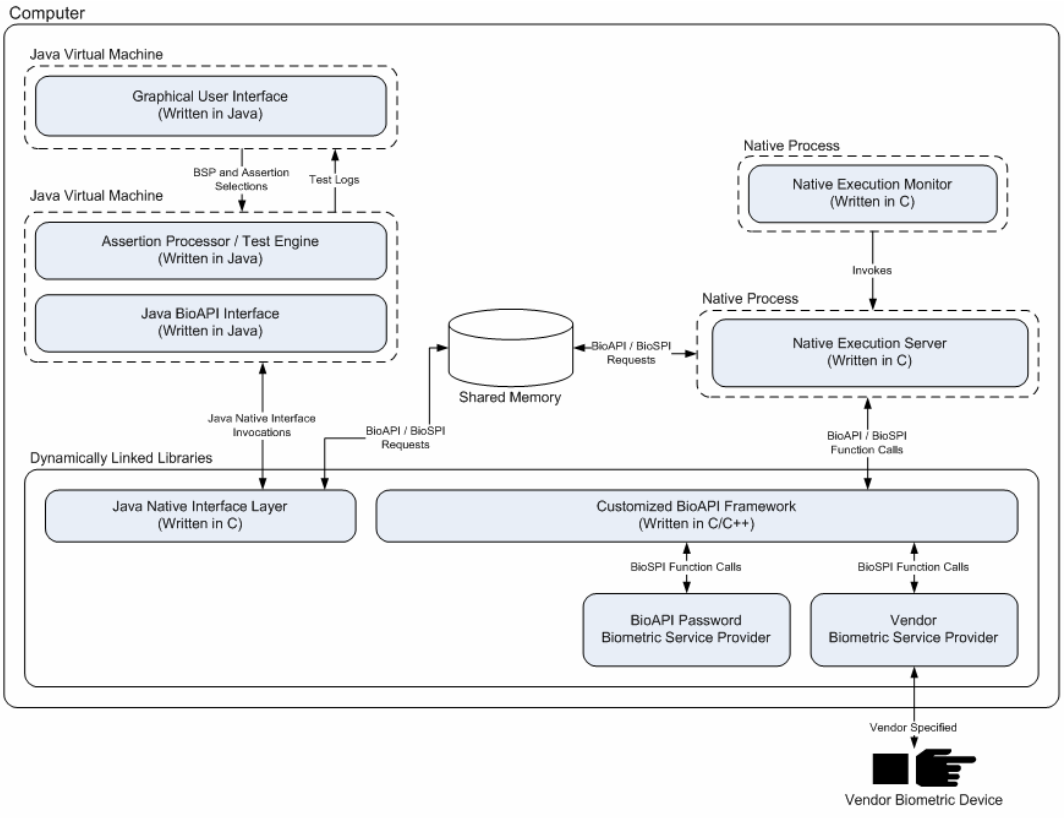
Architecture

The diagram shown below illustrates the overall structure of the BTE.

Essentially, execution occurs in the following manner:

- The user executes a batch file to start the BTE.
- The batch file invokes BTEMonitor.exe to start the Native Execution Monitor.
- BTEMonitor.exe invokes BTEServer.exe to start the Native Execution Server.
- The batch file starts the application GUI in a new Java Virtual Machine.
- The user selects a BSP and one or more assertions to test.
- The user presses the "Run Test" button to start the test.
- The application GUI starts the Assertion Processor/Test Engine in a new Java Virtual Machine.
- The Assertion Processor/Test Engine parses the assertion file. For each BioAPI/BioSPI function invocation defined in the assertion, the Assertion Processor/Test Engine invokes the corresponding method in the Java BioAPI Interface.
- The Java BioAPI Interface performs an initial translation of data types, and invokes the required function in the Java Native Interface Layer.
- The Java Native Interface Layer translates Java data types to C data types and transfers the function call request to shared memory.
- The Native Execution Server retrieves the function call request from shared memory and invokes the required function in the Customized BioAPI Framework.
- The Customized BioAPI Framework performs any processing required of the framework and invokes the required BioSPI function in the BSP.
- The BSP executes the BioSPI function and displays a user interface and/or communicates with a biometric device if necessary. The BSP returns the results to the Customized BioAPI Framework.
- The Customized BioAPI Framework returns the function results to the Native Execution Server.
- The Native Execution Server transfers the function results to shared memory.
- The Java Native Interface Layer retrieves the function results from shared memory, and translates C data types to Java data types.
- The Java Native Interface Layer returns the results to the Java BioAPI Interface.
- The Java BioAPI Interface performs final translations of data types and returns the results to the Assertion Processor/Test Engine.
- The Assertion Processor/Test Engine interprets the results, assesses compliance, and logs the results to an XML file.

- After the Assertion Processor/Test Engine finishes invoking the functions defined in the assertion, the Native Execution Server terminates.
- The Native Execution Monitor starts a new instance of the Native Execution Server.
- The Assertion Processor/Test Engine terminates.
- The application GUI reads the test log and displays the result.
- If the user selected additional assertions, the application GUI starts the Assertion Processor/Test Engine to process the next assertion.
- Once the selected assertions have completed, the user can review the results, save them to a file, and/or perform additional tests.
- The user closes the BTE.
- The Native Execution Server terminates.
- The Native Execution Monitor terminates.



Acronyms

BSP - Biometric Service Provider

BTE - BioAPI Test Environment

GUI - Graphical User Interface

Glossary of Terms

BioAPI Framework - A standard application programming interface and service provider for biometric technologies.

BioAPI Framework, Enhanced Version - The BTE uses an enhanced version of the BioAPI framework. The framework has been extended in two ways.

1. Miscellaneous bugs have been fixed.
2. The framework has been extended to return values required by the BTE.

biometric data

The extracted information taken from a user's biometric sample that is used to build an encrypted reference template. The reference template can then be used to create an enrollment or to match against a user's existing enrollment for authentication purposes.

Biometric Service Provider (BSP) module

A BSP is a software algorithm that extracts a user's biometric sample from a biometric hardware device and translates it into an encrypted numeric representation of that sample. That extracted information is known as biometric data.

Annex A - Development Organizations Involved in the BioAPI CTS Implementation Development

National Institute of Standards and Technology

Founded in 1901, NIST is a non-regulatory federal agency within the U.S. Commerce Department's Technology Administration. NIST's mission is to develop and promote measurement, standards, and technology to enhance productivity, facilitate trade, and improve the quality of life. NIST carries out its mission in four cooperative programs:

- the **NIST Laboratories**, conducting research that advances the nation's technology infrastructure and is needed by U.S. industry to continually improve products and services;

- the **Baldrige National Quality Program**, which promotes performance excellence among U.S. manufacturers, service companies, educational institutions, and health care providers; conducts outreach programs and manages the annual Malcolm Baldrige National Quality Award which recognizes performance excellence and quality achievement;

- the **Manufacturing Extension Partnership**, a nationwide network of local centers offering technical and business assistance to smaller manufacturers; and

- the **Advanced Technology Program**, which accelerates the development of innovative technologies for broad national benefit by co funding R&D partnerships with the private sector.

In FY 2005, NIST had an operating budget of about $858 million and operates in two locations: Gaithersburg, Md., (headquarters - 234-hectare/578-acre campus) and Boulder Colo., (84-hectare/208-acre campus). NIST employs about 3,000 scientists, engineers, technicians, and support and administrative personnel. About 1,800 guest researchers complement the staff. In addition, NIST partners with 1,400 manufacturing specialists and staff at affiliated centers around the country.

NIST/Information Technology Laboratory (ITL)

ITL is one of the measurement and standards laboratories of NIST. ITL works with industry, research, and government organizations to make information technology more usable, more secure, more scalable, and more interoperable than it is today. ITL develops the tests and test methods that both the developers and the users of the technology need to objectively measure, compare and improve their systems. For many years, ITL has been mandated by legislation to provide computer security standards and guidelines to federal

agencies for the protection of sensitive unclassified information in their IT systems and networks.

News and general information about NIST programs and services are available on the World Wide Web at http://www.nist.gov, or by calling General Inquiries at (301) 975-NIST (975-6478), TTY (301) 975-8295 or e-mail: inquiries@nist.gov.

National Biometric Security Project (NBSP)

The National Biometric Security Project is a not-for-profit test, research and analysis organization focused entirely on the application of biometrics to improve the security of the U.S. civil infrastructure. NBSP services are available to government agencies at the federal, state, and local level and private sector agencies responsible for maintaining key components of the national infrastructure.
http://www.nationalbiometric.org/

The Biometric Foundation

The mission of The Biometric Foundation is to advance the use of biometric technologies to accomplish personal verification and identification, to protect privacy, to secure infrastructures that are critical to the nation's economic success, and to prevent identity theft. Under the guidance of industry leaders and experts, the Foundation will conduct and sponsor technology-neutral research, evaluation, and educational programs. These programs are intended to increase understanding of how biometrics work, examine social, economic, and legal issues that affect widespread use of biometrics, and provide authoritative data to support public and private commitment to biometric solutions. The Foundation is committed to standards and practices that ensure it will be a premier independent resource for scientists, policymakers, and citizens who are seeking accurate, reliable information about biometric technologies and their uses.
http://www.biometricfoundation.org/

SAFLINK Corp.

SAFLINK Corporation, a leading provider of integrated enterprise security systems, provides cost-effective software solutions that verify individual identity, protect intellectual property, secure information assets, and eliminate passwords. These solutions are designed to safeguard and simplify access to computer networks, applications, and physical facilities. For more information, please see the Company's website at www.saflink.com or call 800-762-9595.