BioAPI Technical Overview

BioAPI Consortium
Agenda

- What is the BioAPI Consortium
- Features of BioAPI
- Some Concepts
- Interface Modularity
- Architecture
Why Did We Form?

• Founding companies recognized the need for industry standards to ensure widespread rollout of new biometric technologies

• Interoperability is needed to integrate various biometric hardware and software products with computer systems

• An industry standard biometric API is the starting point
Consortium Goals

• Define project scope and architecture framework

• Standard multilevel API for multiple platforms and applications

• Interchangeable and interoperable products from supporting vendors
BioAPI Mission Statement

• Consortium was formed to develop a widely available and widely accepted API that will serve for various biometric technologies. The intent is to:
  – Work with industry biometric solution developers, software developers, and system integrators leverage existing standards to facilitate easy adoption and implementation.
  – Develop an OS independent standard.
  – Make the API biometric independent.
BioAPI Membership

• Promoters:
  – Compaq
  – IBM
  – Identicator
  – Microsoft
  – Miros
  – Novell

• Contributors (to date):
  – I/O Software
  – IriScan
  – NIST
  – NRI
  – NSA
  – Printrak International
  – Recognition Systems
  – Siemens AG
  – Unisys
Key design & development drivers

BioAPI must:

• Provide a multi-level API approach, addressing a wide variety of implementation environments
• Provide comprehensive support for multiple biometric technologies in a common framework
• Provide a robust security architecture for biometrics
• Provide a vendor independent process for development and ownership
• Momentum critical to driving rapid wide scale acceptance as the industry standard
BioAPI scope
Biometric features 1

• Provide multiple levels of interfaces from simple to complex and flexible. Suitable for prototyping, systems integration and product creation.
• An extensible framework capable of equitably supporting a comprehensive set of biometrics, including face, fingerprint, voice, hand recognition, iris and others
• Provide support for multiple sources of raw biometric data (i.e. not just hardware devices)
• Define common, cross-biometric, terminology
BioAPI scope

Biometric features 2

- Allow raw biometric data to undergo zero to any number of transformations to end up with a biometric model or template
- Provide tagging of biometric objects, but leave specification of content to vendors
- Provide a platform for the eventual standardisation of biometric objects
- Provide a comprehensive scoring mechanism for verification and identification
- Provide secure transport of biometric data
BioAPI scope
Environmental features

• Support for multiple languages: C, C++, JAVA
• Support for multiple platforms enabling cross-platform solutions.
• High level, platform specific support for components can be developed (ActiveX/COM, JAVA Beans)
• Support multi-process and multi-machine implementations
• “Hooks” for secure networking and encryption
BioAPI scope

Other features

• Support for internal and external data storage models
• Provide introspection of biometric system meta data. A programmer may find out about the biometric environment programmatically
• Enable internationalisation
The BioAPI Concept

A BioAPI view of how biometric systems work
BioAPI concepts

- The BioAPI Application has access to a Biometric Space
- Within that Space can be multiple Biometric Systems providing services via multiple levels of interfaces
Biometric System

The Source is a logical Biometric input device, it could be a real device, or a file. A system can have multiple sources of different types...
The transformer takes a biometric object from a suitable source and turns it into a different kind of biometric object: Feature Extraction, Compression, etc.

Multiple transformers may be chained to achieve complex results...
Transformers allow new and enhanced biometric algorithms to be plugged into the system as they evolve within the industry.
Biometric objects can be stored in one or more registries. This insulates the application from needing to know where and how the biometric data is stored.
Biometric data can be securely passed between BioAPI biometric systems using Transporters.
Biometric matchers can be used to verify, or identify using standard calls.
BioAPI Interface Modularity

The starting point for customer investment protection: biometric vendor interoperability and application re-use
BioAPI Interfaces

- Application is compiled against BioAPI headers and is linked to the run-time
- Calls are made via Interfaces which have multiple Function Members
- An interface is the “unit” of vendor modularity within BioAPI
**BioAPI Interfaces**

- Interfaces are dynamically bound at run-time
- No re-compilation or re-linking is required
- Multiple vendors may supply the same interface within a single system
- Management of this “plug & play” framework is handled by the run-time
- Interfaces are registered in the BioAPI Configuration Database

**APPLICATION**
calls
BioData_Export

BioAPI run-time provides *hollow* function
BioData_Export
BioAPI Interfaces

- Vendors can choose which interfaces to support
- Different interfaces are supported at different layers
- Vendor implementation libraries can be provided at different levels e.g.
  - Layer 1, BioDevice
  - Layer 3, BioSource, BioRegistry
BioAPI
Architecture Overview

How can all this be achieved?
A BioAPI conformant application makes calls into one or more of the API levels (1 to 4).

BioAPI will be a multi-level API from low-level, high degree of programmer control through to high-level, simpler to use functionality.

Each API level contains a set of interfaces. A BioAPI interface is a set of related functions and is the basic level modularity available to biometric vendors: a vendor can choose to implement an interface or not.

The BioAPI runtime provides “hollow” (or stub) implementations of each interface that delegate their functionality onto the vendor supplied implementation of a given interface.

The higher the application’s choice of programming level the easier the task of migrating the application from one biometric type to another or from one biometric vendor to another.
This shows the BioAPI runtime delegating the functionality of a given interface through to a vendor-supplied implementation of that interface.

The runtime handles the lookup, loading and calling of the vendor supplied modules.

Note that there can be multiple implementations of a given interface by the same or different vendors: the runtime maintains sufficient context to ensure the Application’s request is routed to the correct implementation module.
In addition to the API levels BioAPI will include a **Programming Support Library**.

This is used by both the application and the vendor supplied modules. The programming support library comprises a set of utility interfaces such as:

1) **BioConfig** - an interface to access information in the BioAPI configuration database
2) **BioUID**  - an interface for the generation and comparison of unique identifiers
3) **BioMemory** - an (optional) interface for the allocation and deallocation of memory
4) **BioError**  - an interface for reporting and querying errors
5) **BioTrace**  - an interface for writing diagnostic information to the BioAPI trace log
One further library is mentioned here: the **Framework Support Library**.

This library is not available publically to an application or a vendor: it is purely to support the implementation of the BioAPI runtime libraries.

It would contain utility interfaces to do such things as dynamically load a vendor module and to efficiently manage the creation and manipulation of BioAPI handles.

*BioFS_Handle*

*BioFS_Implementation*
The second library to support the vendor supplied modules is the **Platform Abstraction Library**. Unlike the Vendor Support Library the Platform Abstraction Library is a public library: it can be used by an application, a vendor and the framework (runtime) itself. As the names suggest it job is to hide all platform specific calls behind a standard set of interfaces. If this library is used correctly the job of porting any part of BioAPI (ie the framework (runtime), an application or a vendor supplied module) to an alternative platform is greatly reduced.

Some of the interfaces found in the Platform Abstraction Library include:-

1) BioPA_File - an interface for creating, reading and writing text and binary files.
2) BioPA_Memory - an interface for the allocation and deallocation of memory.
3) BioPA_Mutex - an interface for creating and using mutexes for thread synchronisation.
4) BioPA_Thread - an interface for creating a new thread of execution.
5) BioPA_ThreadData - an interface for managing thread specific data.
6) BioPA_Time - an interface for querying and formatting the time (and date).
7) BioPA_Trace - an interface for writing diagnostic information to the BioAPI trace log.
8) BioPA_UID - an interface for the generation and comparison of unique identifiers.
To aid the vendor’s development of interface implementations 2 support libraries are provided by BioAPI. The first of these is the **Vendor Support Library**. This is a set of interfaces that are only available to vendors and not to an application programmer. Such interfaces include:-

1) BioVS_Biometric - an interface for registering and removing biometrics from the BioAPI configuration database.

2) BioVS_Source - an interface for registering and removing sources from the BioAPI configuration database.
Who will want to use BioAPI?

• Implementation of BioAPI will provide value to:
  – Security products
  – Vendors of diverse biometric technologies
  – System Integrators and value added resellers
  – Application developers
  – End users of biometric technologies
For example, secure biometric authentication...

application client/server communications

secure BioAPI transporter “tunnel”
Summary of key features

- Multi-level application interfaces
- Standard modular access to biometric functions, algorithms, and devices via dynamically replaceable interfaces
- Common access mechanisms for biometric data management and storage
- Standard ways of differentiating biometric data and device types via a “plug and play” run-time
- Support for biometric identification in distributed, heterogeneous computing environments