



BALTIMORETM
www.baltimore.com

Product Overview

Baltimore **KeyTools**TM

elsecurity for developers

Preface

Copyright © 2000 Baltimore Technologies plc.

All rights reserved.

The copyright of this work is vested in Baltimore Technologies plc and the documentation is issued in confidence for the purpose only for which it is supplied. It must not be reproduced in whole or in part or used for tendering or manufacturing purposes except under an agreement or with the consent in writing of Baltimore Technologies plc and then only on the condition that this notice is included in any such reproduction. No information as to the content or subject matter of this document or any part thereof arising directly or indirectly there from shall be given orally or in writing or communicated in any manner whatsoever to any third party, being an individual firm or company or any employee thereof, without the prior consent in writing of Baltimore Technologies plc.

Global e-security, the Baltimore Logo, Baltimore KeyTools and Baltimore product names including Baltimore Telepathy, Baltimore SureWare, KeyTools Lite, KeyTools Pro, KeyTools SSL, KeyTools S/MIME, KeyTools XML, KeyTools Telepathy m-Sign are trademarks of Baltimore Technologies plc. All other trademarks used throughout this publication are the property of their respective owners. Users should ensure that they comply with all national legislation regarding the export, import and use of cryptography.

Table of Contents

Executive Summary	1
KeyTools Lite and KeyTools Pro at a Glance	3
KeyTools Snap-in Components at a Glance.....	5
KeyTools Standalone Modules at a Glance.....	7
E-security infrastructures and e-business	8
Integrating security into application environments.....	10
KeyTools Overview.....	12
KeyTools Architecture.....	14
KeyTools API Structure	17
KeyTools Lite and KeyTools Pro	18
Security Policy Enforcement - Enterprise-wide common security and manageability.....	18
Certificate Authority Support – The open standards approach	19
Software-based Cryptography - seamless replacement of cryptographic libraries.....	20
Private Key Security.....	21
Plug-in Crypto.....	21
Smartcard and Token support - hardware based security.....	22
Directory Support – the LDAP interface.....	22
Transport Mechanism Support - communicating with remote entities.....	23
KeyTools Components.....	24
KeyTools SSL.....	24
KeyTools S/MIME.....	25
KeyTools XML.....	26
KeyTools Telepathy m-Sign	27
KeyTools Standalone Modules.....	29
KeyTools Crypto	29
Sample Code #1	30
About Baltimore Technologies.....	34
Appendix A: Public Key Infrastructure Systems	35
Appendix B: PKCS Standards	37
Glossary	39

Executive Summary

Full e-security for the Application Developer

In today's internet environment your application's e-security has become an essential factor to your application's success. Baltimore KeyTools, the leading suite of security tools, enables the developer to focus on core application development and not worry about complex security coding. This means rapid application development, speed to market, and competitive advantage. It also means choosing a set of components that will solve all your security needs both now and into the future.

Baltimore's KeyTools is the industry's leading developer security suite, and offers the developer an unparalleled value proposition: best-of-breed technology, practical licensing, trusted vendor, ease of use, full strength cryptography, industry-wide interoperability and scalability.

Baltimore Technologies understands the imperatives and constraints of application development. As the recognized global leaders in developer security tools, Baltimore Technologies provides a complete range of snap-in components accessed via a single plug-and-code architecture: Baltimore KeyTools.

Baltimore's KeyTools provides the necessary tools to enable applications to operate within a digital certificate based security infrastructure and to implement Digital Certificate systems with strong cryptography and a range of algorithms including the RSA algorithm.

Baltimore's KeyTools modular framework offers rapid application development and the flexibility required to integrate industry standards (such as SSL, XML, S/MIME and more) into any application framework and all via a single API. Based around an intuitive architectural, KeyTools' modular approach enables the developer to add additional functionality to an application by simply adding a snap-in component to the core library. These snap-in components include:

- SSL – for fully secure (authentication, encryption, data integrity) on-line communication using any sockets-based connection
- S/MIME – for secure messaging, such as email (signing and encrypting)
- XML – for securing (signing and encrypting) the industry standard format for document exchange
- Telepathy m-Sign – a WAP compliant digital signature toolkit for the wireless world

Soon to be integrated into the KeyTools architecture:

- Path Validator – for verification of digital certificates, including full certificate path construction

Baltimore KeyTools forms an essential component of Baltimore's e-security offering. As a leading global supplier of solutions for e-business, e-commerce and enterprise systems, Baltimore Technologies understands the security requirements of the application developer. When developers need to address a new market demand, they need a partner who provides up to the minute technology to allow for rapid product development. Baltimore Technologies is the leader for first-to-market developer security tools:

- first to market Java cryptography (April 97 – later licensed to RSA Security Inc.)
- first to market XML security (Aug 99)
- first to market wireless security - WTLS (Sept 99).

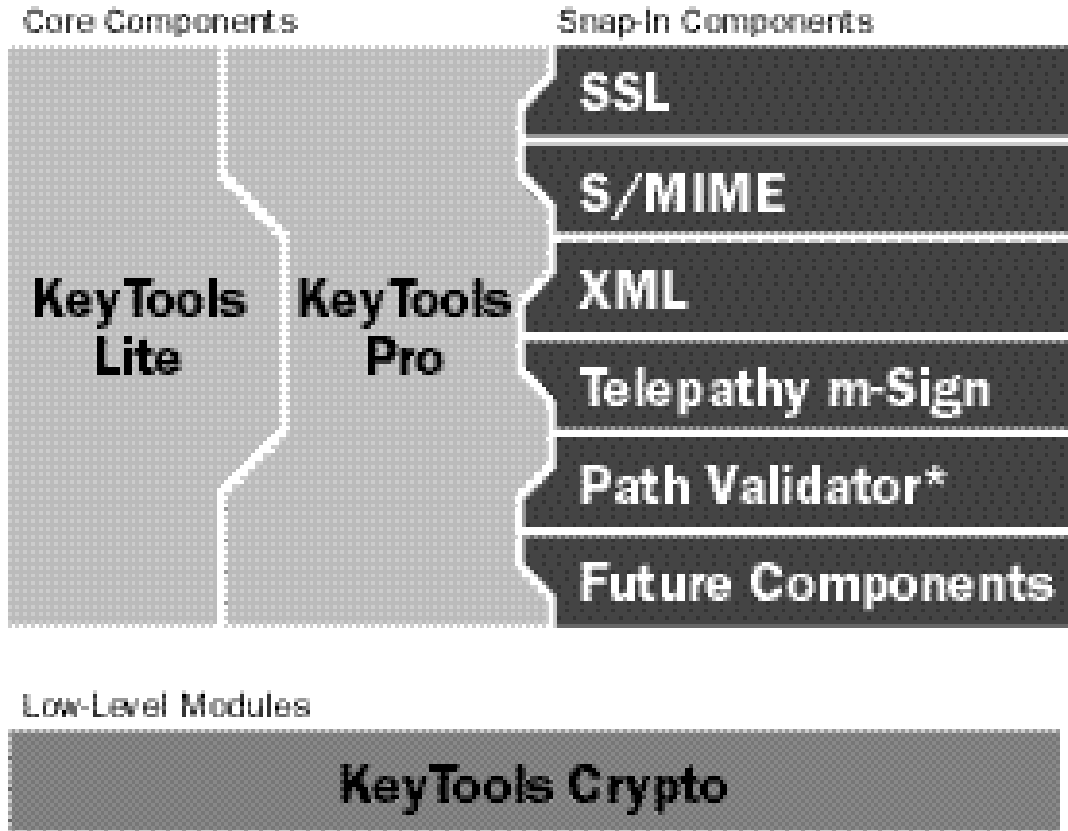
Years of experience has lead to specialized expertise in developing security solutions for sectors such as:

- On-line healthcare and tax systems
- E-Government infrastructures
- Finance and banking applications
- E-business and e-commerce components
- Application Service Providers
- Secure email clients and servers
- Access control systems
- Digital Signature software systems
- Virtual Private Networks (VPNs)
- Interaction with Certificate Authority systems

Baltimore Technologies is committed to providing leading edge products and has a proven track record of first-to-market innovation to meet new developer requirements. With sales and support offices throughout the Americas, Europe, Asia and Australia, development centers in America, Europe and Australia, and alliances worldwide with Baltimore TrustedWorld Partners, Baltimore offers unparalleled global reach.

KeyTools Lite and KeyTools Pro at a Glance

The core components of the KeyTools developer suite



The KeyTools Developers Toolbox core library is available in two possible configurations, depending on your requirements:

- KeyTools Lite
- KeyTools Pro

All snap-in components are available with either KeyTools Lite or KeyTools Pro. These include the S/MIME, SSL, XML, Telepathy m-Sign and Path Validator*.

* **Note:** The Path Validator will be available in early 2001

KeyTools Lite and KeyTools Pro are the core libraries used by application developers and are designed to implement Digital Certificate systems, with strong cryptography, for interaction with other components within a PKI.

KeyTools Lite and KeyTools Pro implement the following features:

KeyTools Lite

- Policy based – locally defined security policies
- High-level, intuitive API
- Full strength cryptographic functionality
- Support for a wide range of Certificate Authorities
- File transporter (PKCS #10 / PKCS #7 based) for interaction with Certificate Authorities
- PKCS #12 support
- Support for X.500 Directory systems via LDAP (Lightweight Directory Access Protocol)
- Full digital certificate handling functionality
- Limited certificate extension support
- CRL (Certificate Revocation List) checking

KeyTools Pro

All features in KeyTools Lite, plus:

- Policy based – centrally/remotely generated and locally defined security policies
- Support for a wide range of smartcards and other hardware cryptographic tokens
- PSE (Baltimore's Personal Secure Environment) support
- Email transporter (PKCS #10 / PKCS #7 based) for interaction with Certificate Authorities
- HTTP transporter (PKCS #10 / PKCS #7 based) for interaction with Certificate Authorities
- CRS transporter for interaction with Certificate Authorities
- SCEP transporter for interaction with Certificate Authorities
- Full certificate extension support
- CRL Distribution Point
- OCSP support
- PKIX support

Language and Platform support

KeyTools Lite and KeyTools Pro are available as both C++ and Java libraries

The C++ versions of KeyTools are available on the following platforms:

- » Windows 98
- » Windows NT v4
- » Solaris 2.7
- » HP-UX 10.20
- » Additional UNIX platform support to be added

The Java versions of KeyTools Lite and KeyTools Pro are built for the following environments:

- » JDK 1.1.x
- » JDK 1.2.x
- » JDK 1.3

KeyTools Snap-in Components at a Glance

Maximum flexibility through a comprehensive set of snap-ins

KeyTools SSL

KeyTools SSL fully implements both the SSL v3.0 and TLS v1.0 protocols, ensuring secure session-based transactions.

- Client and server RSA and DSA authentication
- Anonymous Diffie-Hellman key agreement
- Dynamic security re-negotiation
- Fully-configurable session caching
- Supports TCP/IP and other user-defined transport layers (e.g., X.25 and serial communications)
- Full socket support library for clients and servers
- Support for blocking and non-blocking I/O in C++
- Full support for smartcards and hardware security modules via KeyTools Pro
- Full support of PKI and certification authorities via KeyTools Lite/KeyTools Pro

KeyTools S/MIME

KeyTools S/MIME fully implements the S/MIME v2 protocol for signing and encrypting data, as typically used in secure email clients and servers.

- Fully secure encryption and digital signing
- Session key generation and encryption
- Message signature verification
- Encryption and decryption of session key for multiple recipients
- Full support for smartcards and hardware security modules via KeyTools Pro
- Full support of PKI and certification authorities via KeyTools Lite/KeyTools Pro

KeyTools XML

KeyTools XML is the latest component of the KeyTools developer suite, implementing the latest IETF/W3C standard on digital signing of XML documents. KeyTools XML also includes routines for encrypting XML documents.

- Digital signing and verification of XML documents guided by the IETF's and W3C's standard
- Full strength encryption and decryption of XML documents
- Easy integration with new and legacy applications.
- Full support for smartcards and hardware security modules via KeyTools Pro
- Full support of PKI and certification authorities via KeyTools Lite/KeyTools Pro

KeyTools Telepathy m-Sign

KeyTools Telepathy m-Sign provides developer products for use within the wireless world. Currently KeyTools Telepathy m-Sign consists of a digital signature toolkit that is compliant with the latest standards for mobile security.

- Support WMLScript SignText(), Verify Text() and Encrypt Text() as specified in WAP
- Verifies digital signatures
- Digitally sign and encrypt data
- Central Management and Configuration through policies
- Full support for smartcards and hardware security modules via KeyTools Pro
- Full support of PKI and certification authorities via KeyTools Lite/KeyTools Pro

Other wireless specific applications and infrastructures are also produced by Baltimore and are available within its Telepathy product range.

KeyTools Standalone Modules at a Glance

Low-level cryptographic components

KeyTools Crypto

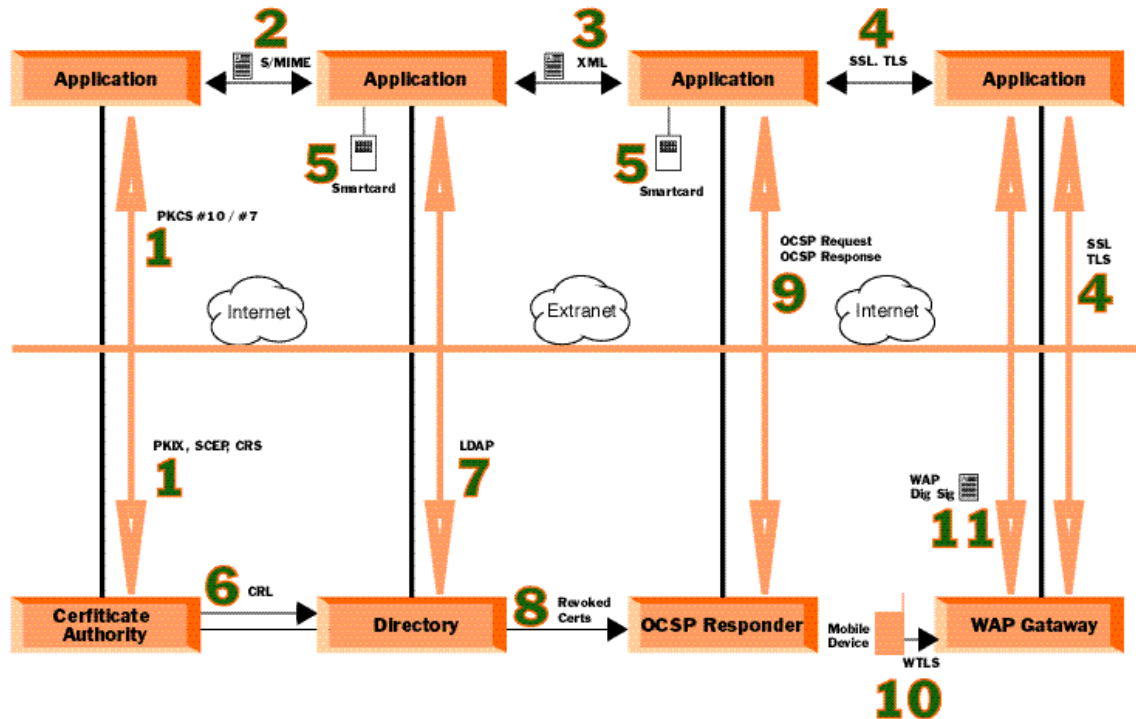
KeyTools Crypto is a low-level standalone developer component providing implementations of common cryptographic algorithms and techniques used in security systems. KeyTools Crypto is available as either highly optimized C libraries or as a 100% pure Java toolkit.

- Full cryptographic functionality, including:
 - RSA, DSA, Diffie-Hellman, DES, Triple DES, RC2, RC4, SHA-1, MD2, MD5, RIPEMD-160, Blum-Blum-Shub & FIPS 186 pseudo random number generation
- ASN.1 handling, Base 64, BER, DER coders
- Compliant with PKCS standards
- CERG compliant pseudo random number generator (C libraries only)
- Support for Intel's hardware based RNG (C libraries only)
- Highly optimized code written in 100% pure Java (Java libraries only)
- PKCS #11 interface (Java libraries only)

KeyTools Crypto is designed to be a pure cryptographic module that does not support other PKI elements, such as digital certificate handling, directory access, SSL, S/MIME and PKCS #7 implementations. KeyTools Crypto is a standalone module and will not act as a snap-in component to KeyTools Lite or Pro.

E-security infrastructures and e-business

The components and procedures for protecting information assets



E-security infrastructures typically consist of a number of complementary components, which together ensure that applications can effectively and securely interact with each other within e-business environments. These components enable applications to conduct e-business transactions that fulfil the basic security requirements of confidentiality, integrity, authentication and non-repudiation. The following outlines some of the typical steps taken when setting up and utilizing e-security protocols and systems:

1. A number of e-security protocols (S/MIME, SSL ...) are based on digital certificates technology. End users, machines and devices can all be certified and issued with digital certificates. **Applications** use standard protocols to make certificate requests to a **Certificate Authority (CA)** and retrieve the generated certificates. These protocols include the PKCS #10 / #7 certificate request / response formats, the PKIX standard and the SCEP and CRS protocols. This is normally the initial step taken by end users and applications.
2. Once a digital certificate is obtained, applications can then interact with each other in a secure manner. Typically applications either communicate via **message based systems** (for example email) or session based systems (web browsing or other http based systems). For messaging based systems the industry-wide security standard is the **S/MIME** protocol. This ensures documents and other electronic data can be signed and encrypted before being sent to the intended recipient, ensuring that origin of the document can be verified and that only the intended recipients have access to the documents contents.

3. **XML** is the extensible mark-up language that has become the industry standard for information exchange. The need for security in XML is clear and the IETF and W3C have defined a standard mechanism for generating XML signatures, which can be used by applications exchanging XML documents.
4. As an alternative to message based systems, applications also communicate via on-line sessions. For **session based security** systems, **SSL** (and the latest version, known as **TLS**) are used to provide confidentiality, integrity and authentication during on-line transactions. These protocols allow secure communication across open networks.
5. When security is of paramount importance, file based storage of sensitive information (such as cryptographic keys) many not provide the level of protection required. In this situation **smartcards** can be used by applications, and their end users. Smartcards store cryptographic keys and other personal information and perform cryptographic operations on the card itself. This ensures that keys never leave the card and are never present, even temporarily, on the computer system that is running the application.
6. Often it is necessary for applications and their users to obtain certificates of other users or to check if the certificates have been revoked. These certificates and **Certificate Revocation Lists (CRLs)** are normally stored in public **directories**. The CA will publish certificates and CRLs to these directories.
7. Applications can then retrieve the certificates and CRLs from the directory by using the **LDAP** protocol.
8. Alternatively, CAs may publish revoked certificate information to an **Online Certificate Status Protocol (OCSP)** responder.
9. Applications can then query the OCSP responder to check the status of a certificate (revoked, non-revoked or status unknown).
10. For wireless devices modifications have been made to standard security protocols to take into account the constraints of these devices, as compared to the systems typically used in standard networks. Wireless devices can connect to the internet and other applications by passing through a **WAP (Wireless Application Protocol) Gateway**, which provides a portal into the wired network. When communicating securely with the WAP Gateway, wireless devices use the **WTLS** protocol, which is the wireless version of the standard SSL / TLS protocol. The Gateway can then use standard SSL / TLS when connecting to other applications.
11. There is also a standard for digitally signing content of wireless devices. These signatures are known as the **WAP Digital Signatures**. Data and the corresponding WAP digital signature can then be passed from a mobile device, along standard networks to an application for processing. This procedure ensures the origin and integrity of the data sent.

Integrating security into application environments

Developer requirements explored

PKI-enabled applications typically interact with other entities within a Public Key Infrastructure (PKI) and other e-security environments in a number of standard ways. These include:

- Generating cryptographic identities for application users
- Sending and receiving digitally signed and encrypted data to and from other applications
- Requesting certificates from Certification Authorities
- Retrieving user and CA certificates from certificate repositories (e.g. an LDAP-enabled directory)
- Checking the revocation status of certificates (e.g. using OCSP or CRLs)
- Leveraging on the security offered by cryptographic tokens (smartcards and hardware security modules)

In order for interoperability with third party systems, all the above operations must be based on open, robust, industry-wide standards. These include standard implementations of common cryptographic algorithms including the RSA algorithm, security communication protocols such as SSL and S/MIME, internet standards such as LDAP and PKI components including X.509 certificates, PKCS, PKIX and OCSP protocols and ANSI standards.

Any integration of security components into applications should be performed in as efficient and simple a manner as possible. What does this mean in practice? What do developers, product and project managers and their CTOs look for in an e-security toolkit?

Any e-security component should have the following attributes available to the developer:

- High-level interface – rapid application development and speed to market.
This is an essential component of any developer product. In today's fast moving environment speed to market is vital to success. In order to achieve this objective, any third party product being used within a development team must be accessed through a high-level, intuitive, easy-to-use interface. This will reduce development time and ensure that developers do not need to learn complex cryptographic and security techniques and protocols.
- Fully integrated set of e-security components - the 'toolbox' approach.
Every application has a different set of security requirements. E-security developer products should reflect this fact. Developers need products which allow them to choose the components they require for their application. Further, each component should be fully integrated with all others to ensure that components interoperate seamlessly. In addition, an integrated 'toolbox' approach means that each component conforms to an overall architecture and all interfaces presented by each component have a common look and feel. This approach ensures that future e-security requirements can easily be met, with new components quickly and readily integrated into existing and future applications. An added benefit is that there is little additional learning when the developer wants to add further functionality.

- **Manageability – security policy control.**
The paradigm of a Security Policy driven API is a radically new concept. Security policy can be used to enable the provision of a simple top-level interface for developers. With security policy, all necessary security parameters can be set quickly and easily by either the application developer, the application user or a central administrator. This increases speed and ease of development, reduces risk of error, makes testing easier and lowers the cost of maintenance. With central security policy control all application security parameters can be managed from one central point, giving unparalleled configuration and control over a distributed install base. Importantly, this security policy is not simply a development feature, but something which is resident in the end-user application. In this way, Security Policy becomes a value added feature of the application. This affords the application developer major competitive advantage over competitors.
- **Cross platform support.**
Developers require high performance components to run both natively on industry standard platforms and to be fully platform independent i.e. to be built using 100% pure Java. Only with this comprehensive cross platform support will a fully deployable solution be possible.
- **Rapid Problem Resolution – Global Customer Support.**
A significant consideration in ensuring speed to market is that complex problems must be resolved quickly. Developers require expert support from a trusted vendor, with the scale, experience and global reach to ensure support issues can be resolved rapidly.

KeyTools Overview

The complete e-security infrastructure for the application developer

Baltimore Technologies provides a range of security products and services for e-commerce and enterprise systems. Each product is designed to be the best in its class and together they offer a complete solution for integrating security into both legacy and new applications and environments.

Baltimore's products are designed and built based on a number of principles including:

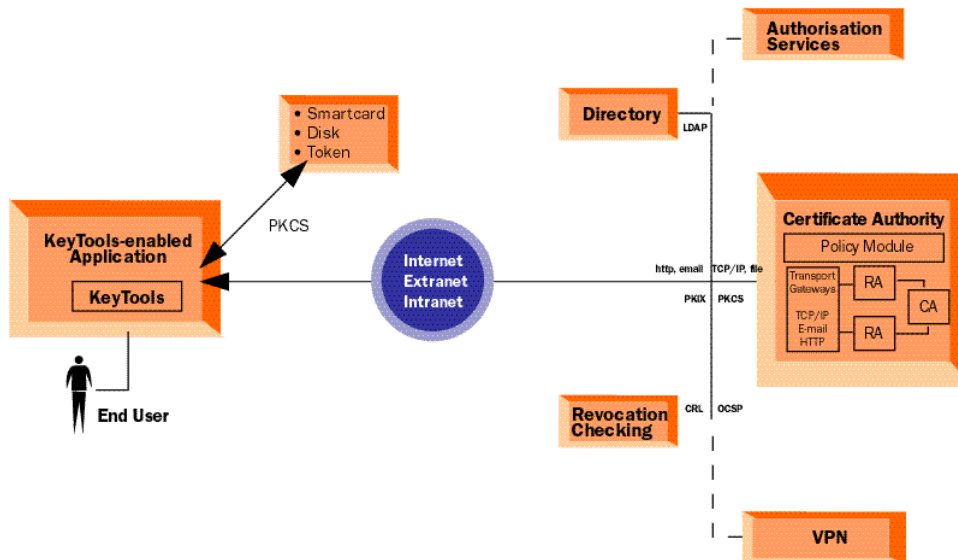
- Flexibility
- Scalability
- Open standards
- Policy based
- Resilience
- Full Strength Security

Within the overall product set, Baltimore's KeyTools provides all the components required to PKI-enable any application. By adopting a modular approach to its e-security developer products, Baltimore ensures that application developers have the most consistent, easily integrated set of security libraries for integration into new and legacy applications.

Unlike other developer products, KeyTools requires minimum cryptographic or digital certificate knowledge on behalf of the developer. KeyTools offers a high-level API which can be readily integrated into applications without difficulty. Additionally, its unique policy-enforcement system allows enterprises to dictate their security policy at the application and desktop level.

The KeyTools developer suite consists of a core library along with a number of optional components that extend the functionality of the core library. Rather than simply a set of separate toolkits, the toolbox approach ensures that each additional component is tightly integrated with the core library. Each component extends the core library's API, ensuring that developers program using the same architecture, irrespective of the particular security components that are being integrated into an application. This enables rapid application development and total flexibility. Additional security features can be deployed within an application simply by plugging in the appropriate component. The core library and all components are fully policy based, ensuring a consistent security infrastructure across all applications that integrate the KeyTools product range. The policy driven nature of KeyTools enables central policy control and update of applications, leading to the most efficient, flexible and customizable application security management system available in the market today.

Since security-enabled applications can be operated on a private (intranet, extranet) and public (internet) basis, KeyTools is designed to work with both scenarios without difficult



Global Customer Support

Baltimore Technologies is committed to providing leading edge products and ongoing support to its worldwide customer base. With support offices in the Americas, Europe, Asia and Australia, development centers in America, Europe and Australia, and alliances worldwide with Baltimore TrustedWorld Partners, Baltimore offers a global support infrastructure. This ensures that an expert team is always standing by to help resolve any difficulties, no matter what the time of day or the geographical location.

KeyTools Architecture

Policy-based, modular design accessed through a high-level, intuitive API

The KeyTools design is based on a number of principles:

- Simplicity of use
- Interoperability with a wide range of systems
- Object Oriented Design
- Support of standards
- Support for security policies

Additionally, KeyTools allows for:

- Seamless replacement of the cryptographic library
- Enhanced compatibility with the Baltimore UniCERT Certificate Authority through its support of a PKI Policy Enforcement system

KeyTools is designed to be technology and algorithm neutral, and the interface is unchanged irrespective of the particular PKI technology used. This means that new cryptographic algorithms, certificate revocation mechanisms etc. can be integrated into KeyTools without requiring changes to the interface. Thus application developers can concentrate on solving real world problems (such as protecting data by signing and encrypting it), rather than worrying about implementation details of specific cryptographic technologies.

KeyTools architecture is described in detail in the following categories:

- High-level interface
- Modular Approach
- Policy Control
- Cross Platform Support

High-level interface – intuitive API

KeyTools Lite and KeyTools Pro form the core of the KeyTools developer suite. The cryptographic and certificate engines that are common to all components are contained within the KeyTools Lite and KeyTools Pro modules. The modules present a fully object oriented, high-level, intuitive Application Programmers Interface (API). This allows for rapid application development and ensures that developers are given access to cryptographic and e-security techniques without having to gain expertise in these areas. All standard operations can be performed with a minimum of calls to the KeyTools API and all operations come with an in-built set of default security parameters, ensuring that security is maintained.

The following are some of the major sections of functionality that can be accessed through the KeyTools high-level API:

- Full cryptographic functionality
- Digital certificate handling
- Secure storage of personal information
 - » such as cryptographic keys and certificates
- Directories
 - » via LDAP
- Cryptographic token support
 - » via PKCS #11
- Transport mechanisms for interacting with Certificate Authorities
 - » PKCS #10 / PKCS 7 via email and http, SCEP, Certificate Request Syntax (CRS), PKIX

Modular Approach

KeyTools is designed to offer a modular structure for application developers. This gives two major advantages:

- Developers need only integrate the components that are required for each particular application
- Since all components are part of a general KeyTools API, all component APIs follow the same design and architecture. Thus developers already have a familiarity with the API using any of the other components. This ensures that any future security requirements can be quickly and easily integrated into existing and new applications. Furthermore, all components will be fully interoperable with each other.

The KeyTools developer suite offers two configurations of its core library: KeyTools Lite and KeyTools Pro.

KeyTools Lite is a fully functional library that includes all components that are needed to interact with other elements of a Public Key Infrastructure. KeyTools Pro gives additional e-security mechanisms, including: smartcard support, online transport mechanisms to CAs (email, http), SCEP, OCSP and PKIX.

Additional libraries can be added to KeyTools Lite/Pro to provide further e-security functionality. These include modules for: SSL, S/MIME, XML security, wireless e-security and Certificate Path Construction and Validation*. Each of these components extends the core API and offers unparalleled integration of all major security requirements into one comprehensive architecture.

* **Note:** Path Validator component will be available in early 2001

Further details of all components can be found in the section entitled 'KeyTools Lite and KeyTools Pro at a Glance'.

Security Policy Control

Product managers understand that it is customer demand which is driving the adoption of security in applications. Keeping this in mind, the value of offering your customers richer security features can easily be seen. Security policy control is one of the unique features within KeyTools that gives both users and developers the flexibility and control they require to ensure full and easy-to-use manageability in their applications.

Whereas other vendors have tried to add security policy to their architectures long after the product has been designed, Baltimore is unique as the only developer e-security toolkit vendor to have built policy in from day one.

The benefit from an operational point of view is that security policy can be dictated centrally thus providing 'managed organizational security' for sets of end-users, servers or devices. This means multiple applications and entities can be controlled by one central IT security administrator. This is one of the most important benefits of the policy driven API. Security policy control enables management of e-security parameters such as the cryptographic settings for different users and applications, certificate registration settings, key storage settings and the ability to enforce a rules based security infrastructure. See the 'Security Policy Enforcement' section for further details.

Developers have to deal with variations in the registration and certification process, type of keys and tokens to be issued, and how these credentials should then be used by end-entities. Security policy control gives the developer the ability to address these issues effectively.

Baltimore has applied the concept of Security Policy within its toolkit architecture to deliver a number of benefits. In the development environment the policy is used to enable the provision of a simplified top-level API, since the setting of all necessary security parameters is firstly set in the policy. From a development perspective this delivers tangible benefits such as: increased speed and ease of development, reduced risk of error, ease of testing and low cost of maintenance.

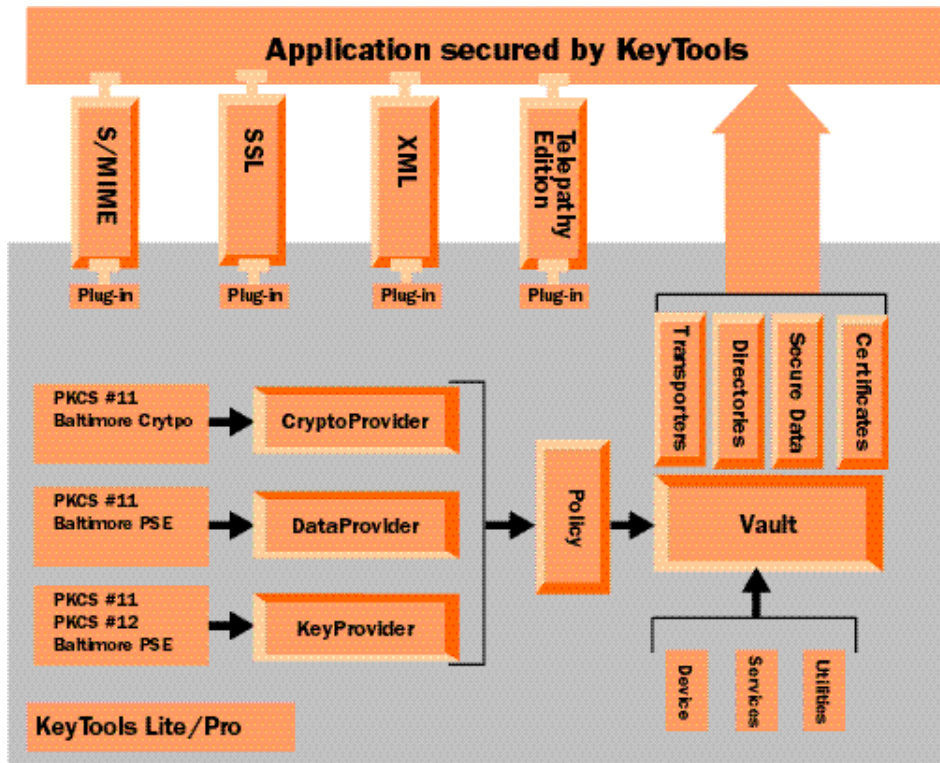
Baltimore is in the process of extending the range of security policy elements that can be centrally managed and providing a new Policy Authority that will be responsible for distributing new policies to end-entities. The KeyTools components will enforce these new security policy rules and hence deliver policy enforcement to PKI enabled server and desktop applications.

Cross Platform Support

KeyTools is available both as C++ and 100% pure Java libraries. This ensures highly portable, high performance code that can be integrated into nearly all environments, operating on multiple platforms.

The C++ versions of KeyTools Lite / KeyTools are available on Windows and major Unix platforms, among them: Windows 98, Windows, Solaris, and HP-UX. With code written in Java, nearly all platforms in use today are available for running the applications. In order to ensure maximum platform support, KeyTools is available for use in three Java environments, namely: JDK 1.1x, JDK 1.2.x (Java 2) and JDK 1.3.

KeyTools API Structure



KeyTools Lite and KeyTools Pro

The core components of the complete PKI solution

KeyTools Lite and KeyTools Pro are the core components of the KeyTools product family. These are the key elements which enable the developer to build an application to operate within a Public Key Infrastructure. The cryptographic and certificate engines that are common to snap-in components are contained within the Lite and KeyTools Pro modules. Additional features such as policy control, certificate revocation checking etc. are performed by the core components.

KeyTools Lite is a basic library that includes all the essential elements needed to interact with other elements of a PKI such as certificate handling, directory support, certificate request and retrieval. Alternatively, KeyTools Pro provides a broader range of e-security mechanisms to enrich the functionality of the end application. Examples include smartcard support, additional transporters, OCSP, central policy control, CRL distribution points. Full details can be found in the 'KeyTools Lite and KeyTools Pro at a Glance' section.

Security Policy Enforcement - Enterprise-wide common security and manageability

KeyTools offers a unique security policy system, which can be used to enforce an enterprise-wide security policy. This is a unique and extremely powerful mechanism which offers highly flexible control.

Baltimore's KeyTools architecture has encompassed security policy management since its inception in order to deliver a number of benefits. The paradigm of a policy driven API is unique to Baltimore Technologies. The security policy is used to enable the provision of a simplified top-level API, since the setting of all necessary security parameters is firstly set in the security policy. The benefits of this approach include increased speed and ease of development, reduced risk of error, ease of testing and low cost of maintenance. From an operational point of view this means that security policy can be dictated centrally thus providing 'managed organizational security' for sets of end-users, servers or devices.

Central policy control allows for central policy creation and enforcement. The central policy can then be used by all KeyTools enabled applications. For example, security policy allows you to specify a key size to be used by thousands of clients. A simple change of policy by an operator can change the key size for all client applications with a single instruction. This is extremely beneficial to enterprise security systems.

Application programmers can avail of this Policy Enforcement system as required. With KeyTools security policy enforcement, the security behavior of the application being enabled can be constrained to any degree by the developer by delegating security settings to the central policy administrator.

Real world examples of security policy in action are given below:

- To direct users to use the PKI server in finance for future certificate issuance.
- To provide a template for registration information required when requesting a certificate from a CA
 - » distinguished name, validity, signing algorithm, extensions mandated, CA signing algorithm...
- To set cryptographic parameters
 - » type, length, validity, source of randomness, key usage, extended key usage, storage...
- Secure Storage
 - » file, smartcard or HSM (Hardware Security Module based)
- Certificate Storage Information
 - » LDAP directory location
- Security servers that should be used
 - » OSCP, LDAP, PKI, Archive Server, Time Stamping Sever ...
- Policy refresh management by end-entities

Baltimore Technologies is in the process of extending the range of security policy elements that can be centrally managed and providing a new Policy Authority that will be responsible for distributing new policies to end-entities. Baltimore KeyTools will enforce these new security policy rules and hence deliver policy enforcement to PKI enabled server and desktop applications.

Certificate Authority Support – The open standards approach

KeyTools Lite/Pro is designed to work with all popular CAs. Clients using KeyTools can work with both private CA systems and public CA systems (Trusted Third Parties).

CA systems range from simple, single user programs to modular, highly scalable systems, which can support millions of certificates. Each CA has its own design and configuration but, to ensure interoperability, CAs should support the open, industry wide PKI standards.

KeyTools provides all necessary functionality for a client to communicate with a CA. With KeyTools Pro, developers also get a set of built-in transporter mechanisms (email, http, etc) to communicate with a CA.

Baltimore recommends the use of the Baltimore UniCERT CA system. KeyTools offers enhanced compatibility with UniCERT through its support of a PKI Policy Enforcement system. Full details are available at www.baltimore.com/products/unicert.

Certificate Request Message Syntax

KeyTools supports certificate requests in PKCS#10 and PKIX formats. Certificates can be retrieved in BER, DER and .p7c (PKCS #7) format.

KeyTools is designed to support encrypted certificate requests where the certificate may contain sensitive or confidential data. This will be based on an encrypted form of the standard request syntax.

KeyTools fully implements the Certificate Request Syntax (CRS) protocol, enabling KeyTools to fully interoperate with the VeriSign OnSite system.

Revocation

Efficient certificate revocation is essential to the correct working of a PKI. KeyTools supports full client certificate lifecycle controls.

With the PKIX Revocation Request Message standard, KeyTools provides the following:

- Revocation Request

Certificate Revocation Lists (CRL) Version 2 are fully supported:

- Read CRL
- Lookup CRL for specific certificate status (based on certificate serial number)
- Correct parsing of CRL Distribution Point information from X.509 certificates

The Online Certificate Status Protocol (OCSP) is also supported in KeyTools:

- Send OCSP request for certificate status
- Interpret OCSP response

Certificate Processing

KeyTools security logic eliminates the need for developers to work with certificates at a detailed level. Most operations automatically perform implicit certificate parsing based on standard requirements.

KeyTools includes a high-level set of functions that provide intelligent interpretation of X.509 certificates including:

- Certificate Validity Checking
- Extraction of public-key from certificate
- Generation of self-signed certificates
- Extraction of certificate extensions

Software-based Cryptography - seamless replacement of cryptographic libraries

KeyTools provides Baltimore's own full-strength cryptographic libraries as standard. However it is designed to allow a developer to use crypto libraries from other vendors. Because of KeyTools object oriented design this is very straightforward and requires no code changes at the application level.

In KeyTools Lite/Pro the cryptographic services are not directly accessed by the developer. Instead they are handled by a high level programming interface called the Vault. The Vault also works in tandem with the security policy to provide a simple, intuitive development environment. The Vault API is identical regardless of whether it is implemented in hardware or software devices, hence making the source of the cryptography transparent to the developer.

Algorithms provided include:

- RSA
- DSA
- Diffie-Hellman
- DES
- Triple-DES
- RC2
- RC4
- SHA-1
- MD2
- MD5

All algorithms are available with full-strength keys.

At the API level, KeyTools supports:

- Pseudo Random Number Generation
- Key Generation
- Highly secure private key storage (PKCS #1, #5, #8, #12)
- Sign, Encrypt, Decrypt, Verify functions
- Message Authentication Codes (keyed hashes)

Private Key Security

A PKI must provide a highly secure means to protect the private keys of users. If a private key is compromised, then all security functions involving that user are suspect.

KeyTools supports the abstract concept of a Vault. A Vault is a secure area in which cryptographic operations are performed and private keys and associated certificates are stored. Within this concept, private keys and other sensitive data are held securely and can only be accessed or altered by the authorized client. A Vault may be realized in many different forms within KeyTools and the toolkit supports both disk and smartcard based Vaults.

For disk based systems, the private key is encrypted with a user supplied passphrase. For smartcard systems KeyTools uses the cards in-built security to protect the private key, which is usually accessed by a PIN.

Plug-in Crypto

KeyTools plug-in design allows easy incorporation of third-party crypto libraries if required. KeyTools will support software libraries that are accessed through either a PKCS #11 interface or Baltimore's own cryptographic interface. KeyTools will, in future, support the CAPI architecture, enabling CAPI modules to be seamlessly plugged in as a cryptographic provider.

Smartcard and Token support - hardware based security

Note: Smartcard and Token support is only available in KeyTools Pro

Smartcards offer a highly portable and secure method for storage of data. Additionally, cryptographic smartcards can securely perform key generation, signing and encryption of data.

For application developers, it is important to offer support for the widest range of smartcards possible. KeyTools Pro fully abstracts cryptographic and secure storage functionality allowing programmers to rapidly prototype applications using software-based systems and to seamlessly change to smartcards when required.

In KeyTools Pro cryptographic services implemented within smartcards and hardware tokens are accessed in exactly the same way as software-based cryptographic services i.e. through the Vault interface. The Vault works in tandem with the security policy and is identical regardless of whether it is realized by hardware or software devices. Thus the source of the cryptography is transparent to the developer and allowing seamless migration from software to hardware based systems and vice-versa.

Within KeyTools Pro, smartcards are supported through the PKCS #11 (Cryptoki) interface for tokens. PKCS #11 promises compatibility across a range of smartcards and smartcard readers.

KeyTools Pro has successfully interoperated with a number of smartcards and tokens including Baltimore SureWare, and those offered by third-party vendors including: GemPlus, Chrysalis-ITS, DataKey and nCipher.

Directory Support – the LDAP interface

Directories provide an efficient means for certificate storage and retrieval within a PKI system. As part of its certificate control system, CAs populate directories with certificates and Certificate Revocation Lists (CRLs). Client applications can then use the directory to retrieve the certificate based on a parameter such as name or email address. Additionally, clients can check the CRL to determine whether an individual certificate is revoked or not.

The X.500 Directory standard has been dominant in the messaging arena to date, with major directory vendors supporting the Lightweight Directory Access Protocol (LDAP). X.500 directories are offered by a number of vendors worldwide, including Isocor, Novell, Netscape, CDC, Siemens and Platinum.

KeyTools Lite/Pro provides full support for LDAP directories. KeyTools is tested for correct operation with most of the leading directory systems. For specific details on directory support, visit our web site at www.baltimore.com or contact Baltimore at info@baltimore.com.

In conjunction with UniCERT, KeyTools can restrict client access to a set of directories so that an enterprise security policy can be enforced.

KeyTools provides the following directory support:

- Full support for LDAP v3
- Directory Lookup
- Directory Certificate Write
- Supports CRLs

Transport Mechanism Support - communicating with remote entities

Note: A file based transport mechanism is available in KeyTools Lite, all other transport mechanisms are only available in KeyTools Pro

A Public Key Infrastructure is comprised of multiple entities, each of which must be capable of talking to each other. Therefore, any PKI-enabled application must support mechanisms for communicating with other applications and CAs. A number of different transport mechanisms exist, however for each mechanism separate code normally needs to be written by the application developer.

KeyTools Pro negates the need to write separate code for each transport mechanism. All communication is handled by an abstract Transporter class. This class presents a generic interface that is independent of the transport mechanism used. This results in consistent coding standards and means that the code base is unchanged if, for example, an organization decides to migrate to web based certificate request system from an email based system in an application built with KeyTools Pro.

KeyTools provides the following transport mechanisms:

- SMTP/POP3
- HTTP
- CRS (Certificate Request Syntax)
- SCEP
- Sockets
- File

KeyTools Components

KeyTools SSL

KeyTools SSL is Baltimore's e-security component that enables you to write SSL and TLS-secured Internet and Intranet applications. Secure Sockets Layer (SSL) has become an industry standard with a multitude of vendors employing it to protect their communications. The KeyTools SSL libraries make it easy to add full-strength encryption, integrity and authentication to all on-line communications. KeyTools SSL is fully compliant with version 3.0 of SSL and version 1.0 of TLS.

KeyTools SSL toolkit allows the developer to integrate SSL and TLS data encryption capabilities into any online networked application. KeyTools SSL will provide confidentiality, integrity and authentication for many different types of applications, including:

- Internet communications – client / server environments
- Secure Internet Banking - funds transfer, access bank account details online
- Online Shopping - secure exchange of credit card details
- Online Brokerage Services - secure financial dealings
- Secure Web Casting - ensures security on web push technologies
- Secure Database Connectivity - remotely access & update a central database
- Telnet - secure remote log-in to online servers
- Distributed computing models – CORBA / RMI

KeyTools SSL provides the ability to configure the security parameters to be used for authentication and data security, and to initiate and receive SSL and TLS-secured connections. KeyTools SSL is a component of either KeyTools Lite or KeyTools Pro and utilizes a common set of core libraries. This means that functionality such as certificate handling, smartcard access, revocation checking and LDAP connectivity are fully available for use within SSL-enabled applications. It also ensures that a common key and certificate store is used both by KeyTools Lite/Pro and the SSL component. This highly integrated infrastructure results in a highly flexible and consistent code base for all security components within an application.

KeyTools SSL includes fully configurable support for session caching, security re-negotiation, and temporary key reuse. For non-TCP/IP environments, KeyTools SSL also supports user-defined record layers, allowing SSL security to be layered on top of alternative transports such as serial links, X.25 connections, etc.

The KeyTools SSL API is powerful, flexible and easy-to-use. KeyTools SSL allows the developer to define the parameters under which the SSL or TLS session will be established. Among these parameters are the cipher-suites to use, the level of client authentication to require, and containers for user identities (private key and certificate chain) and certificates (trusted CA certificates). In addition, session caches and temporary key stores can be installed and configured.

Once these parameters are defined, they are used to establish an SSL or TLS session. A KeyTools SSL class is provided for easily establishing socket connections, with an additional class provided for accepting socket connections, allowing the component to function in both client and server modes. Alternatively, you can use an existing communications medium; for example, a serial port, and then attach a KeyTools SSL session to this medium. Information can then be transferred across the resulting SSL or TLS pipe using standard API input and output stream facilities. Additional functions are provided for querying and upgrading the security parameters on a session and for ultimately closing it down. The Java version of KeyTools SSL also fully supports the Javax SSL API.

KeyTools SSL supports all mandatory and common optional cipher suites, ensuring full interoperability with third party SSL implementations, including SSL implementations in the Netscape and Microsoft browsers and all popular web servers.

With KeyTools Lite/Pro and KeyTools SSL developers have the most comprehensive policy based package available to developers for full on-line security based on industry standard protocols. Further details of KeyTools SSL features can be found in the section entitled 'KeyTools Snap-in Components at a Glance'.

KeyTools S/MIME

KeyTools S/MIME is a high-level e-security tool which allows developers to add full strength security to any messaging application. KeyTools S/MIME provides developers with advanced yet easy to use tools to add strong security features to applications which involve sensitive or monetary based messaging. Its API offers high-level functions to offer confidentiality, integrity, authentication and non-repudiation.

KeyTools S/MIME offers security based on the Internet Engineering Task Force (IETF) S/MIME standard, the de-facto standard for Internet email security which has been adopted by most common email clients and servers, including Microsoft and Netscape. S/MIME utilizes advanced cryptography techniques to offer a platform and transport independent standard for a wide variety of security requirements. Software developers utilizing KeyTools S/MIME can add security without prior knowledge of cryptographic systems or S/MIME protocols. Security features such as digital signatures, message hashing and key generation are all handled transparently by KeyTools S/MIME.

Standard email applications are inherently insecure, particularly over the Internet, as it is possible for network administrators, ISP's and others to intercept, read and alter messages. Software developers who are delivering client and server email applications can use the KeyTools S/MIME component to protect and to enhance their email services as follows:

- To ensure that mail is confidential and can only be read by the receiver
- To ensure that mail cannot be altered in transit
- To provide sender-generated digital signature, guaranteeing the receiver that an incoming email genuinely came from its stated source.

Baltimore's KeyTools S/MIME is a library of routines which allows a programmer to design applications that can create and interpret S/MIME compatible messages. KeyTools S/MIME snaps into either KeyTools Lite or KeyTools Pro and utilizes a common set of core libraries. This means that functionality such as certificate handling, smartcard access, revocation checking and LDAP connectivity are fully available for use within S/MIME-enabled applications. As with KeyTools SSL, it also ensures that a common key and certificate store is used both by KeyTools Lite/Pro and the S/MIME component. This highly integrated infrastructure results in a highly flexible and consistent code base for all security components within an application.

KeyTools S/MIME was designed to be easily integrated into email and messaging products. It allows end users to benefit from the security features provided by the S/MIME protocol its standards-based approach facilitates interoperability between applications. These applications include:

- Secure Email
- EDI
- Electronic Commerce Services
- Healthcare Applications
- Content Delivery and Internet Push
- Secure Messaging for Legacy Applications

The KeyTools S/MIME API extends from KeyTools Lite/Pro. With KeyTools S/MIME developers can enable any message, including email, to be secured, using digital signatures and encryption. An S/MIME class is provided for easily manipulating messages, abstracting detailed security processing away from the developer. The Java version of KeyTools S/MIME also fully supports the standard JavaMail API.

With KeyTools Lite/Pro and KeyTools S/MIME developers can add messaging based security and full PKI functionality to any application, and all with full policy control. Further details of KeyTools S/MIME features can be found in the section entitled 'KeyTools Snap-in Components at a Glance'.

KeyTools XML

KeyTools XML was the worlds first commercial XML security tool. KeyTools XML offers both XML digital signatures and XML encryption in one complete package. Extensible Markup Language (XML) is rapidly becoming the language of e-business. XML is being used to create standard document types that represent the electronic equivalent of physical documents such as purchase orders and invoices. As with all e-business and e-commerce systems security is of paramount importance.

To date, the security features required for sending XML documents over the Internet have not been included in XML applications. Utilizing state-of-the art cryptography, digital certificates and digital signatures, KeyTools XML solves this by providing the four core security elements: confidentiality, integrity, authentication, non-repudiation.

KeyTools XML requires minimum cryptographic or digital certificate knowledge on the part of the developer or integrator. It offers an API, which can be readily integrated into new and legacy applications without difficulty. Unlike other PKI products, the digital signatures and encryption provided by KeyTools XML result in pure XML documents.

KeyTools XML offers the following benefits:

- Allows secure e-business XML systems to be created
- Provides complete confidentiality of XML documents
- Prevents tampering of documents during either transmission or storage
- Unique and irrefutable digital signatures can be applied to part or whole documents
- Full Public Key Infrastructure (PKI) support
- No prior knowledge of cryptography is required

KeyTools XML represents the fusion of PKI and XML. It allows developers to build full Public Key Infrastructure based security into XML applications. It offers APIs for digitally signing XML documents, verifying digitally signed XML documents, and for decrypting encrypted XML documents. Along with these basic security operations it also provides, through KeyTools Lite or Pro, full support for manipulating certificates and private keys in industry standard formats. Full PKI support is provided, including certificate revocation support, LDAP lookup of certificates, etc.

Although digitally signing an XML document is a complex process, the powerful and flexible API abstracts much of this complexity away from the developer. XML signatures can have two forms:

- Complete documents, known as Detached Signatures, which verify the authenticity of external documents, as well as text and data elements that can be embedded within the signature document
- XML structures that are embedded within larger XML documents, known as embedded signatures verifying the authenticity of elements of the larger structure as well as external documents.

Both of these forms are supported in KeyTools XML. As with all KeyTools components, KeyTools XML enjoys full policy control capabilities. Further details of KeyTools XML features can be found in the section entitled 'KeyTools Snap-in Components at a Glance'.

KeyTools Telepathy m-Sign

The growth in the wireless market is being driven by the immense, universal popularity of mobile phones, personal digital assistants (PDA) and handheld PCs (HPC). Services and applications for these devices are increasing rapidly. In order to deliver these services and applications in a secure, scalable and manageable way, new architectures and protocols are being designed. The Wireless Application Protocol (WAP) is a result of continuous work to define an industry-wide specification for developing applications that operate over wireless communication networks. The WAP specification is developed and supported by the wireless telecommunication community so that the entire industry and, most importantly, its subscribers can benefit from a single, open specification.

With this rapid growth of m-commerce, application developers now require secure solutions for mobile transactions. An intrinsic aspect of any security mechanism is the ability to digitally sign and verify data. KeyTools Telepathy m-Sign is a developer toolkit designed to provide the full range of features required to process mobile-generated digital signatures. Leading the way in security for mobile devices, KeyTools Telepathy m-Sign is the world's first digital signature toolkit to support WAP (Wireless Application Protocol) digital signatures. It is the enabling technology to allow developers to gear their applications for the next phase of m-commerce and wireless services.

KeyTools Telepathy m-Sign is an essential tool for anyone creating a mobile commerce solution that requires digital signatures to be used for authentication, confidentiality, integrity and non-repudiation. The internationally recognized WAP 1.2 standard specifies digital signatures in WML (Wireless Mark-up Language). KeyTools Telepathy m-Sign implements that specification and allows content providers to receive signatures sent from a mobile device and verify their validity using a PKI.

KeyTools Telepathy m-Sign offers the following benefits:

- Allows secure mobile commerce solutions to be created
- Provides complete confidentiality of data transmitted from mobile devices
- Prevents tampering of documents during either wireless transmission or storage
- Full Public Key Infrastructure (PKI) support
- No prior knowledge of cryptography is required

KeyTools Telepathy m-Sign is designed to fully integrate with other KeyTools modules to give the complete security solution. Further details of KeyTools Telepathy m-Sign can be found in the section entitled 'KeyTools Snap-in Components at a Glance'.

Note: Baltimore Telepathy is the family of Baltimore products for the wireless environment. To learn about other products in the wireless range visit <http://www.baltimore.com/telepathy/>

KeyTools Standalone Modules

KeyTools Crypto

KeyTools Crypto is a powerful cryptographic component enabling developers to build strong cryptography into applications, based on state-of-the-art techniques. Using KeyTools Crypto, almost any application can be developed to include any the most popular and trusted cryptographic algorithms, such as RSA, DSA, Diffie-Hellman, DES, Triple-DES, RC2 and RC4 etc. KeyTools Crypto also provides access to a range of powerful modules including passphrase based encryption, hashing algorithms, pseudo random number generation, key handling, etc. KeyTools Crypto provides unlimited cryptographic key lengths with no restriction on the strength of security.

KeyTools Crypto supports RSA key pairs of any length. Keys are generated with a number of desirable security-related properties, including to resistance to iterated encryption attacks and several factoring algorithms. Both forms of PKCS#1 padding (for public keys and private keys) are supported, along with programmer supplied padding schemes. Private keys can be saved to disk using the PKCS #5 passphrase protection standard. The Digital Signature Algorithm has also been fully implemented. Combined with Diffie-Hellman key exchange this allows exchange of authenticated and validated Public Keys.

KeyTools Crypto supports ECB, CBC, CFB and OFB mode 56-bit DES and 112-bit Triple-DES encryption. Key generation includes suppression of weak and possibly weak keys. PKCS #5 compliant padding can be automatically performed or left to the application programmer. An extremely fast implementation of the RC4 stream cipher is provided.

Further details of KeyTools Crypto can be found in the section entitled 'KeyTools Standalone Modules at a Glance'.

Note: KeyTools Crypto is a standalone module and will not act as a snap-in component to KeyTools Lite/Pro. It is designed for applications that require an e-security library with a low footprint. All cryptographic functionality available in KeyTools Crypto is also available in KeyTools Lite and KeyTools Pro.

Sample Code

Example #1: Requesting a certificate via email using KeyTools Pro

The following sample code shows how KeyTools Pro can be used to request and retrieve a certificate from a Certificate Authority using the in-built email transporter and standard PKCS #10 / PKCS #7 formats.

```
// An EMailTransporter is set with details specific to your mail server...

EMailTransporter transporter( mailServer,
                              yourEMailAddress,
                              RA_gatewayAddress,
                              yourEMailIdentity );

// The policy determines which security operations are permissible when
// using KeyTools Pro

SmartPointer<Policy> policy(new Policy, &SmartDelete<Policy>);
policy->load(_T("ExamplePolicy.xml"));

Vault vault(*policy);

// Open the vault. A PKCS #12 Vault will be used. The PKCS #12 object
// cannot be opened without a password

vault.openVault(PKCS#12deviceID, password);

// Now all that remains to be done is to apply for the certificate. This // method
// creates a PKCS#10 certification request, signs it and submits
// it the CA system via the EmailTransporter.

DistinguishedName dName(_T("c=ie,o=Baltimore Technologies,
                            ou=PKI-Plus Team,cn=Lnr Foley,
                            email=efoley@baltimore.com"));
PKIServicesClient pkiServiceClient(*policy);

cout << "Applying for certification ..." << endl;
pkiServiceClient.applyForCertificate(transporter, vault, dName);

// The reply from the CA is identified by the subject line on the email
// which the CA sends back. The value of this subject line is
// controlled by the CA.

ByteArray subjectLine("Unicert Certification Authority Reply");
```

Sample Code

Example #1: Requesting a certificate via email using KeyTools Pro

```
// Now we can retrieve a response from the CA. This method on the policy
// will automatically insert the retrieved certificate into the Vault.

bool found = false;
while (!found)
{
    found = pkiServiceClient.retrieveCertificate(transporter,vault,
                                                subjectLine);

    if (!found)
    {
        cout << "no response yet" << endl;

        // sleep for 2.0 sec
        System::sleepMillis(2000);
    }
    else
    {
        cout << "Got a certificate" << endl;
    }
}

// The Vault is now closed using a pin/passphrase to protect it.
vault.closeVault(password);
```

Sample Code

Example #2: Signing and verifying an XML document using KeyTools XML

The following sample code shows how KeyTools XML can be used to sign and verify an XML document by using the KeyTools Vault and Policy.

```
// The first few lines are generic PKI-Plus setup code. Create and load
// the policy. Create the Vault. Open the Vault.

SmartPointer<Policy> policy(new Policy, &SmartDelete<Policy>);

policy->load(_T("../DerFiles/ExamplePolicy.xml"));

Vault vault(*policy, mycallback);
vault.openVault(PKCS#12deviceID, password);

// Now create a few content references for adding to signature.
// An external file reference placed in the manifest in a manifest called //
policyManifest.

XMLContentReference policyRef1(_T("http://forehead/ExamplePolicy1.xml"));
policyRef2.putInManifest(_T("policyManifest"));

// Another external xml file placed directly in the reference list.

XMLContentReference policyRef2(_T("http://forehead/ExamplePolicy2.xml"));

// Create a list of the references to sign.

SignableXML::XMLContentList stuffToSign;
stuffToSign.push_back(policyRef1);
stuffToSign.push_back(policyRef2);

// Create a SignableXML object with its target location.

SignableXML xmlToSign(_T("../\\DerFiles\\testSig.xml"));
```

Sample Code

Example #2: Signing and verifying an XML document using KeyTools XML

```
// Initialise the signing operation with the list of stuff to sign, a
// CryptoProvider, the DTD for the XML Signature, Signature is to be
// placed as root element, and validation turned on. (no namespaces)

xmlToSign.initSign(stuffToSign, vault.primaryCryptoProvider(),
    _T("xmldsig-core-schema.dtd"), _T(""), true);

// Use the Vault to sign the XML

vault.sign(xmlToSign);

// Verification. Pass a digesting CryptoProvider, and indicate XML
// validation required.

xmlToSign.initVerify(vault.primaryCryptoProvider(), true);

// Retrieve the certificate of the signer

SmartPointer<X509Certificate> pCert = xmlToSign.getSignersCertificate();

// Pass it to the Vault with the signable object for verification.

vault.verify(xmlToSign, *pCert);

if (!fRet)
{
    return fail;
}

return ok;
```

About Baltimore Technologies

About Baltimore Technologies plc

Baltimore Technologies plc is a leading global supplier of e-commerce and enterprise security solutions and is a market leader in the fast-growing area of Public Key Infrastructure (PKI) technology.

The Baltimore product and service portfolio includes:

- Award-winning Certificate Authority systems
- Toolkits to PKI-enable applications
- Secure e-mail, Web forms and file transfer
- Secure messaging toolkits
- Cryptographic hardware devices
- Consultancy
- Education and training
- Solutions development and implementation
- Global support

Baltimore Technologies is committed to providing global access and support to its worldwide customer base. With sales and support offices throughout the Americas, Europe, Asia and Australia, development centers in Europe and Australia, and alliances worldwide with Baltimore TrustedWorld Partners, Baltimore offers truly global e-security.

Baltimore is also an active promoter of open industry standards. PKI World, Baltimore's Technology Alliance Program, unites industry-leading organizations that provide a choice of complementary standards-based technologies needed to deploy flexible, PKI-secured applications. Baltimore's client list covers many of the world's leading financial institutions, telecommunications and postal companies and government organizations, including ABN Amro Bank, Australian Tax Office, Bank of England, Bank of Ireland, Belgacom, Deutsche Bank, European Commission, Home Office (UK), NHS, PTT Post (Netherlands), Tradelink (Hong Kong) and Ulster Bank.

For further information please visit the Baltimore Web site:

<http://www.baltimore.com>

mail to: info@baltimore.com

Appendix A: Public Key Infrastructure Systems

A scalable security framework for open, distributed environments

Public Key Infrastructure (PKI) provides the core framework for a wide variety of components, applications, policies and practices to combine and achieve the four principal security functions for commercial transactions:

- Confidentiality – to keep information private
- Integrity – to prove that information has not been manipulated
- Authentication – to prove the identity of an individual or application
- Non-repudiation – to ensure that information cannot be disowned

Lack of security is often cited as a major barrier to the growth of e-commerce, which can only be built on the confidence that comes from knowing that all transactions are protected by these core functions.

The Components of a PKI

A Public Key Infrastructure is a combination of hardware and software products, policies and procedures. It provides the basic security required to carry out electronic business so that users, who do not know each other, or are widely distributed, can communicate securely through a chain of trust. PKI is based on digital IDs known as "digital certificates" which act like "electronic passports", and bind the user's public key to his or her identity.

A PKI should consist of:

Security Policy

Since a PKI plays a central role in security, it must have a defined policy, which can be developed and applied to all computer-based systems within the organization. A security policy sets out and defines an organization's top-level direction on information security, as well as the processes and principles for the use of cryptography. Typically it will include statements on how the organization will handle keys and valuable information, and will set the level of control required to match the levels of risk.

Certificate Authority

The CA system is the trust basis of a PKI, as it manages public key certificates for their whole life cycle. The CA will:

- Issue certificates by binding the identity of a user or system to a public key with a digital signature
- Schedule expiry dates for certificates
- Ensure certificates are revoked when necessary by publishing Certificate Revocation Lists (CRLs)

When implementing a PKI, the CA can exist within many different domains, but essentially the CA can be one of two types: Private and Public. A Private Certificate Authority is typically deployed by a corporation for securing transactions between its business and any related parties (e.g. customers, suppliers). Public Certificate Authorities (Trusted Third Parties) are available on open networks, such as the Internet, and facilitate security between previously unrelated parties.

Certificate Distribution System

Certificates can be distributed in a number of ways depending on the structure of the PKI environment, for example, by the users themselves, or through a directory service. A directory server may already exist within an organization or one may be supplied as part of the PKI solution. Directories offer a scalable and efficient mechanism to store and maintain a variety of different data types.

PKI-enabled Applications

E-commerce and enterprise systems require security at many different levels, many of which can be addressed through use of a PKI. The users of a PKI range from network users to devices, computers, applications, content etc. In order to use the security features available within a PKI, these elements must be able to communicate with other entities within the PKI and use cryptography and associated technologies, thus achieving the level of security required. Examples of applications are:

- Communications between Web servers and browsers
- Email
- Credit card transactions over the Internet
- Virtual Private Networks (VPN)

Appendix B: PKCS Standards

A framework for ensuring interoperability in PKI-based systems

The Public-Key Cryptography Standards (PKCS) are a set of specifications designed to ensure that multiple applications and systems designed to work within a PKI will interoperate with each other. The PKCS standards focus on cryptographic techniques, digital certificate use and smartcard interfaces.

PKCS #1

PKCS #1 defines a method of encrypting and signing data using the RSA public key cryptosystem. It describes a syntax for RSA public and private keys and three signature algorithms for signing certificates.

PKCS #3

PKCS #3 describes a method for implementing Diffie-Hellman key exchange.

PKCS #5

PKCS #5 describes a method for encrypting messages with a secret key derived from a password. The method is intended primarily to encrypt private keys when transferring them between systems but also can be used to encrypt messages.

PKCS #7

PKCS #7 describes a standard syntax for data that may be encrypted or signed, such as digital envelopes or digital signatures. It allows other attributes, such as timestamps, to be authenticated along with the message content. The syntax is recursive so that envelopes can be nested, or someone can sign some previously encrypted data.

PKCS #8

PKCS #8 describes a syntax for private key information - including a private key and a set of attributes - and a syntax for encrypted private keys. PKCS #5 can be used to encrypt the private key information.

PKCS #9

PKCS #9 defines selected attribute types for PKCS #6 extended certificates, PKCS #7 digitally signed messages, and PKCS #8 private-key information.

PKCS #10

PKCS #10 defines a syntax for issuing certificate requests to a CA.

PKCS #11

PKCS defines communication methods with cryptographic devices such as smart cards.

PKCS #12

PKCS #12 describes the syntax for storing a user's public keys, protected private keys, certificates, and other related cryptographic information in software. The goal is to standardize on a single key file for use among a variety of applications.

PKCS #15

PKCS #15 specifies a storage format for keys and certificates on smart cards.

Glossary

Access control

The prevention of unauthorized use of a resource, such as a computer system or an application.

API

Application Programming Interface. An interface used by developers to incorporate a toolkit's functionality into an application's code.

Authentication

A guarantee that a message really has come from the person who claims to have sent it.

Certificate (or public key certificate)

A digitally signed data structure defined in the X.509 standard that binds the identity of a certificate holder to a public key.

Certification Authority (CA)

A third party that registers users and certifies their identities by signing their public-key certificate.

Certificate Extension

Mechanism added in version 3 of the X.509 standard whereby certificates can be "extended", in a standardized and generic fashion, to include additional information. An extension consists of three fields: type (type of data in extension value), criticality (single bit flag defining the importance of extension), and value (data for extension).

Certification Path

An ordered sequence of certificates which, together with the public key of the initial object in the path (e.g. CA public key), can be processed to obtain that of the final object in the path (e.g. user's public key).

Certificate Revocation List (CRL)

A list of revoked but unexpired certificates issued by a CA.

Confidentiality

Assurance that the contents of the message have not been disclosed to third parties.

CRL Distribution Point

The location from which a CRL or partial CRL can be obtained.

Cross-Certificate

A certificate authority certificate issued by a CA in one domain for a CA in another domain thus allowing a certificate path to involve more than one certificate authority domain.

DES

Data Encryption Standard. A symmetric algorithm that encrypts blocks of binary data using a 56-bit key.

Diffie-Hellman

Originally invented in 1974, Diffie and Hellman reinvented this key-agreement algorithm in 1976. It is used to create a shared secret random number that can then be used as a symmetric algorithm's session key.

Digital Signature

A block of data created by hashing a file and combining the resulting hash with the sender's private key. It protects against forgery, even by the recipient.

DSA

Digital Signature Algorithm which forms part of the Digital Signature Standard. An asymmetric algorithm used for digital signatures, providing security by discrete logarithm intractability.

Encryption

Encoding human readable text in such a way that it is unreadable.

Hash Function

A function which transforms any data to a fixed-length output, satisfying the following two properties:

- i) it is computationally infeasible to find, for a given output, an input which transforms to that output;
- ii) and it is computationally infeasible to find for a given input a second input which maps to the same output.

IETF

Internet Engineering Task Force - an Internet Standardization group. The IETF is a large, open international community of network designers, operators, vendors and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet.

Integrity

Proof that the message contents have not been altered, deliberately or accidentally, during transmission.

Key Pair

In a public key cryptosystem, the set of related keys which consists of a public key and a private key that are associated with an entity.

LDAP

Lightweight Directory Access Protocol. A client-server protocol for accessing a directory service.

Non-repudiation

Certainty that the sender of the message cannot later deny having sent it.

OCSP

Online Certificate Status Protocol. OCSP may be used by client applications that want to validate certificates and digital signatures.

PKCS

Public-Key Cryptography Standards. A set of standards that have been developed to aid compatibility between different cryptographic products. For more information see Appendix B.

PKI

Public Key Infrastructure - the architecture, organisation, techniques, practices, procedures and applications that collectively support the implementation and operation of a certificate based public key cryptographic system. A PKI is used to establish trust within electronic communities.

PKIX

Extended Public Key Infrastructure - an industry standard endorsed by the IETF for the application of a public key infrastructure.

Private Key

A cryptographic key kept secret, which enables you to sign data.

Public Key

A cryptographic key that is publicly known, which enables you to verify data signed with the corresponding private key.

Public-Key Certificate

A packet of data consisting of your public key and your basic identification details, all signed with the Certification Authority's private key to verify that it is authentic.

Registration Authority (RA)

An entity that is responsible for identification and authentication of certificate subjects, but that does not sign or issue certificates (i.e., an RA is delegated certain tasks on behalf of a CA).

RSA

An asymmetric algorithm used primarily to create digital signatures, and more rarely for encryption. It is named after its creators: Rivest, Shamir and Adleman. It provides security by factorization intractability.

Smart card

A plastic card with a microprocessor that can store information and perform certain calculations and cryptographic functions.

S/MIME

Secure MIME. A specification for secure electronic mail designed to add security to e-mail messages in MIME format via authentication (using digital signatures) and privacy (using encryption).

SSL

Secure Sockets Layer protocol. A security protocol that prevents eavesdropping, tampering, or message forgery with HTTP transmissions and provides server-side, and optional client-side, authentication.

TCP/IP

The basic communication standard protocol within the Internet.

TLS (Transport Layer Security)

An enhanced version of SSL version 3.0

Triple-DES

Based on the DES algorithm. A symmetric cipher that encrypts blocks of binary data using a 112-bit key.

Verification

The process of ensuring that the sender of an electronic document is who they claim to be.

VPN

Virtual Private Network – a closed network of protected communication channels using an open network as a communication medium.

W3C

The World Wide Web Consortium, an organization set up to create standards of interoperability between Web software vendors.

WAP

Wireless Application Protocol. A universal open standard for bringing Internet content and services to mobile phones and other wireless devices.

WPKI

Wireless Public Key Infrastructure.

WTLS

Wireless Transport Layer Security. The wireless equivalent to SSL.

X.509

The X.500 directory service standard relevant to public key infrastructures describing two authentication methods: simple authentication based on password usage and strong authentication based on public key cryptography. Version 3 added certificate extensions to the X.509 standard.



BALTIMORETM
www.baltimore.com

To secure your applications,
call the KeyTools hotline:

1-877-228-9754

Outside the US **+353 1 881 6300**

<http://keytools.baltimore.com>

Americas

USA
1-877-2-BUY-PKI (2-289-754)

Boston, MA
77 A Street
Needham Heights
MA 02494
Tel: (781) 455 3333
Fax: (781) 455 4005

San Mateo, CA
155 Bovet Road, Suite 400
San Mateo, CA 94402
Tel: (650) 372 5270
Fax: (650) 372 5283

Regional Offices

E-mail: info@baltimore.com
Web: www.baltimore.com/contact_us/usa

Europe

Canada
1-877-2-BUY-PKI (2-289-754)

Kanata, ON
444 Hazeldean Road
Unit 2
Kanata, ON
K2L 1V2
Tel: (613) 836 4736
Fax: (613) 836 7716

Latin America
Tel: (978) 694 6114

Ireland
Parkgate Street
Dublin 8
Tel: +353 1 881 6000
Fax: +353 1 881 7000

United Kingdom
The Square
Basing View
Basingstoke
Hampshire RG21 4EG
Tel: +44 1256 818 800
Fax: +44 1256 812 901

Amsterdam
Tel: +31 20 5 829 829

Madrid
Tel: +34 91 458 9840

Munich
Tel: +49 89 5405 2360

Paris
Tel: +33 1 53 43 63 24

Stockholm
Tel: +46 8 5091 4320

Asia-Pacific

Australia
5th Floor
1 James Place
North Sydney
NSW 2060
Tel: +61 2 9409 0300
Fax: +61 2 9409 0301

Japan
New Otani Garden Court 8F
4-1, Kioi-cho
Chiyoda-ku, Toyko
102-0094 Japan
Tel: +81 3 5212 3770
Fax: +81 3 5212 3786

Canberra
Tel: +61 2 6260 5577

Hong Kong
Tel: +852 2168 9500

Singapore
Tel: +65 236 0510

keytools@baltimore.com

© 2000 Baltimore Technologies plc. All rights reserved. Global e-security, the Baltimore Logo, Baltimore KeyTools and Baltimore product names including Baltimore Telepathy, Baltimore SureWare, KeyTools Lite, KeyTools Pro, KeyTools SSL, KeyTools S/MIME, KeyTools XML, KeyTools Telepathy Edition are trademarks of Baltimore Technologies plc. All other trademarks used throughout this publication are the property of their respective owners. Users should ensure that they comply with all national legislation regarding the export, import and use of cryptography.

USPO800-TK004