

J-CAC Overview



Bob Hairfield

**Deputy Product Manager,
Secure Electronic Transactions -
Devices (PM SET-D)
(703) 769-4500**

Purpose



- What is J-CAC
- What Benefits J-CAC Provides to DoD
- How J-CAC works

J-CAC's Goal



Drive adoption of CAC and PKI as a fundamental Information Assurance enabler and Business Modernization Platform.

CAC/PKI Adoption Obstacles



- From our experience, we've identified three key obstacles:
- Navigating "Last Mile" to the CAC is complex and challenging
 - Many sub-systems must work together (e.g. PC/SC, CAPI)
 - There are thousands of interdependencies, variables, and decision paths a material developer must deal with
- Knowledge Base
 - There are very few software engineers that know PKI, CAPI, PC/SC, CAC, BSI, web application architectures, and the DOD Architecture
- Every Material Developer faces the same problem
 - If an application is going to sign, verify, encrypt, decrypt, or read CAC data, they will all end up building similarly capable modules

What is J-CAC



- “Open source” tools for material developers for:
 - PK- Enable- accessing cryptographic services (sign, verify...)
 - CAC-Enable-using CAC data and applets (Manifesting, food service...)
- *Designed* to support Java/web-based applications
 - J-CAC (short for Java- CAC)
 - Light-weight
 - Architected to support the broadest array of DoD Applications
- J-CAC deliverables:
 - Source Code
 - Documentation
 - Libraries
 - Reference Implementations

J-CAC Objectives



- Reduce complexity of last mile
- Modular, extensible architecture that insulates enabled applications from CAC/PKI changes
- Support portability and technology insertion
- Transfer CAC/PKI knowledge and best practices to material developers

J-CAC: Source Code



- Written using C++ and Java
- Java “Bindings” allow Applets to access the C++ components
 - (C/C++, Visual Basic, and .NET bindings projected)
- Freely available to any DoD Component
 - J-CAC source code use may be limited to DoD Applications
 - Code distribution details are TBD

J-CAC: Documentation



- Freely available - like source code
- Three Categories of Documentation
 - J-CAC System
 - Typical IEEE and DoD Documents for a software project
 - SRS, SDD, IDL, DoDAF (OV, SV, TV), SVVP
 - Reference Implementations
 - Documents the integration of J-CAC in the reference apps
 - Developer's Guide
 - Tips and Tricks for using J-CAC to PKE and CAC-E
 - Debugging and Logging tools

J-CAC: Libraries



- Compiled and 'tested' binaries
 - No need to customize or recompile
- Fast, efficient way to get rolling quickly and prototype J-CAC's capabilities
- 'Tested' =
 - We black box tested the libraries
 - We used the same libraries in our reference application
 - The SVVP is the definitive word on what was tested
 - Your mileage may vary
 - J-CAC code and libraries won't undergo security testing or accreditation

J-CAC: Reference Implementations



- Illustrate J-CAC's capability in a relevant, useful environment

Locked: Chat (pure Java Client)

Very Likely: Digital Signature (web forms) and
CAC-E via a browser based applet

Likely: Collaboration
(authentication, signature, encryption)

- Provide holistic view of CAC and PK enabling
 - Great education tool for uninitiated developers
 - Highlights other areas for the material developer to consider
 - Certificate Handling/Validation

J-CAC Benefits to DoD



- Encourages adoption of CAC and PKI
- Built once – everyone benefits
- Can customize, change, fix components to meet unique requirements
- Covers broad array of applications and uses
- Open access to code facilitates trust
 - No magic 'box'
- Educates developer
 - Insight into how it all fits and works together

J-CAC: How it works



Reference Implementations

Java Client Chat

Digital Signature Web Forms

Collaboration VMOC

Source Code/
Developers Guide

J-CAC Language Interface Bindings

Java

C/C++

VB

COM

.NET

Source Code

Interface Specification

Cryptographic Functions	Token Utility Functions
Encrypt(), Decrypt() Verify(), Sign() GetPubicKey(), Get Certificate()	GetReaderList(), Connect(), Disconnect(), SendCommand()

Interface Definition Language

J-CAC Core

Cryptographic Functions	Token Helper Functions	BSI Functions
Encrypt(), Decrypt() Verify(), Sign() GetPubicKey(), Get Certificate()	GetReaderList(), Connect(), Disconnect(), SendCommand()	F1, F2...

Source Code/
Libraries/
Documentation

CAPI

PC/SC



J-CAC Road Ahead



- J-CAC completion is scheduled for 2QFY05
- Identify early adopters
- Sponsorship / Distribution
- Determine long term viability and supportability