Java Specification Request 321: Trusted Computing API for Java™

Tutorial on the Early Draft Review

Ronald Toegl, Werner Keil

Expert Group
JSR-321

Agenda

This is an overview of the upcoming Trusted Computing API for Java. It has recently finished early draft review!
To be released under an open source license.

- Java & TC
- Lessons learned from TSS
- API Design
- Short Example
- Possibilities to influence the specification

Goal: Trusted Java Applications

Java is a natural choice for security critical software

- type-safe
- bounds-checking
- access control checks
- automated memory management
- rich network and cryptographic libraries

A number of use cases for Trusted Computing in Java

- Grid-Computing: policy enforcement, IP protection, data protection
- Web-Services
- DRM etc.
- Remote Attestation Service
- PrivacyCA
- TPM Management Tools

Needed: Java Standard

- So far there has been no standard integration of Trusted Computing in Java.

- JSR321 is a Java Specification Request in the Java Community Process for a Trusted Computing API for the Java SE platform.
  It is aimed to develop a Trusted Computing API for Java providing selected functionality the TCG Software Stack offers to the C world, while following the conventions of modern Java APIs.

The Expert Group

- Specification Lead: Ronald Toegl

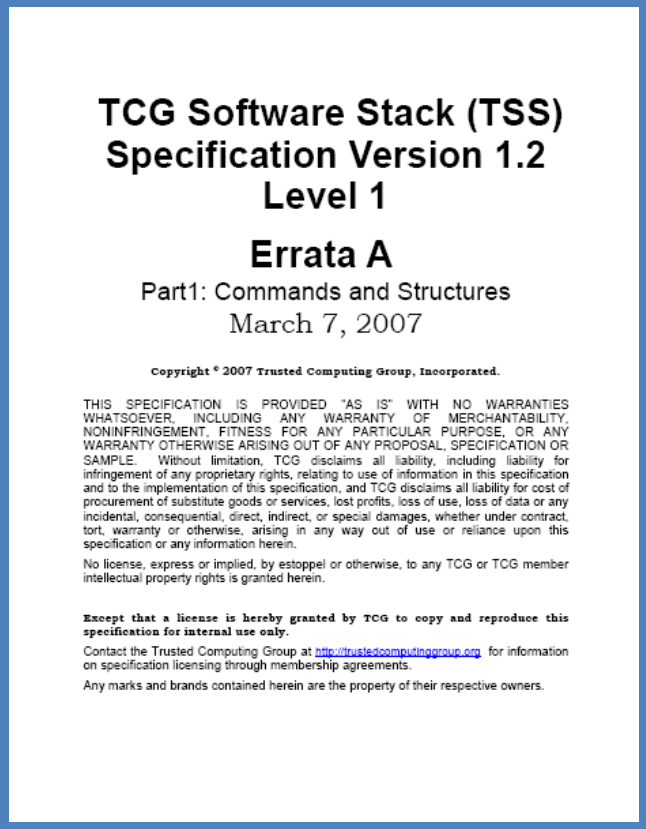
The members of the JSR 321 Expert Group are

- Ronald Toegl and Peter Lipp, Institute for Applied Information Processing and Communications (IAIK), Graz University Of Technology
- Nauman, Mohammad, Institute of Management Sciences, Pakistan
- Kenneth M. Graf, Intel Corp.
- Jeff Nisewanger, Sun Microsystems, Inc.
- Deepak Dasaratha Rao, Samsung Electronics Corporation
- Winkler, Thomas, University of Klagenfurt, Austria
- Werner Keil, Creative Arts & Technologies (Individual), Austria
- Gungoren, Bora, Portakal Teknoloji, Turkey

Informal members of the Expert Group are

- Hong, Theodore, University of Cambridge

A look back at the TSS

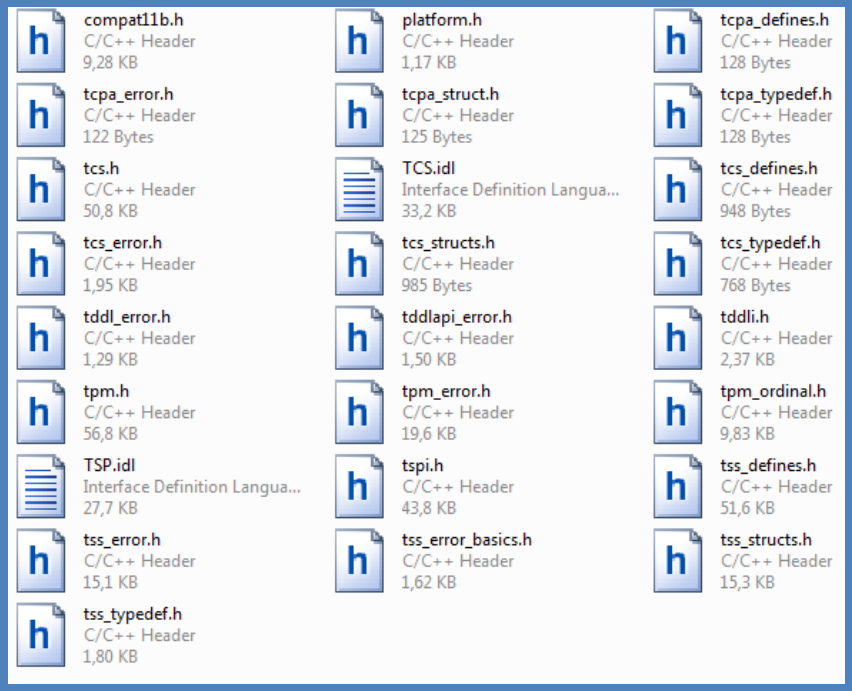


TCG Software Stack (TSS) Specification Version 1.2 Level 1

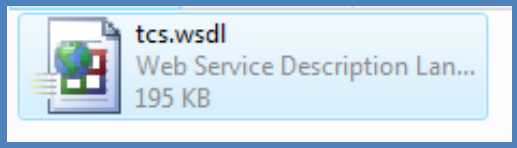
Errata A

Part1: Commands and Structures

March 7, 2007



compat11b.h C/C++ Header 9,28 KB	platform.h C/C++ Header 1,17 KB	tcpa_defines.h C/C++ Header 128 Bytes
tcpa_error.h C/C++ Header 122 Bytes	tcpa_struct.h C/C++ Header 125 Bytes	tcpa_typedef.h C/C++ Header 128 Bytes
tcs.h C/C++ Header 50,8 KB	TCS.idl Interface Definition Langua... 33,2 KB	tcs_defines.h C/C++ Header 948 Bytes
tcs_error.h C/C++ Header 1,95 KB	tcs_structs.h C/C++ Header 985 Bytes	tcs_typedef.h C/C++ Header 768 Bytes
tddl_error.h C/C++ Header 1,29 KB	tddlapi_error.h C/C++ Header 1,50 KB	tddli.h C/C++ Header 2,37 KB
tpm.h C/C++ Header 56,8 KB	tpm_error.h C/C++ Header 19,6 KB	tpm_ordinal.h C/C++ Header 9,83 KB
TSP.idl Interface Definition Langua... 27,7 KB	tspi.h C/C++ Header 43,8 KB	tss_defines.h C/C++ Header 51,6 KB
tss_error.h C/C++ Header 15,1 KB	tss_error_basics.h C/C++ Header 1,62 KB	tss_structs.h C/C++ Header 15,3 KB
tss_typedef.h C/C++ Header 1,80 KB		



tcs.wsdl
Web Service Description Lan...
195 KB

A look back at the TSS

TCG Software Stack (TSS) is the core software component specified by the TCG for interaction with the TPM

TSS design is provided and standardized by the TCG

TSS is specified (amongst others) to

- supply one single (exclusive) entry point to the TPM functionality
- synchronize TPM access
- TPM resource management (key slots, authorization sessions, …)
- building of TPM commands messages according to TPM specification
- manage user secrets
- perform authentication protocols
- handle event log
- Provide APIs for application programmers

It covers **all** operation scenarios: OS, system administration, middleware and applications.

Lessons Learned for Java API Design

- Existing TSS-based infrastructure can be used to support a high-level library
- Usability needs to be improved to lower initial threshold for developers
- Scope needs to be limited to what (Java application) developers need
- Reference Implementations are more than just helpful – especially as Open Source
  Indeed: *required* in the JCP!

➔ JSR 321 is a high-level TC API for Java

Filtering Functionality

- TPM 1.2 only
- Remove features needed only by the BIOS, OS, system service,..
- Many TSS functions are simply not needed in Java™:
  - Management of memory and other resources can and should be hidden from application developers.
  - Object initialization and destruction are natural features of object-oriented languages.
  - Cryptographic primitives like hash functions are already well-supported in the Java™ Cryptography Extension (JCE).
- *Tested* TSS implementations of functionality must be available

Filtering Functionality

- As heard JSR-104 with IAEK Graz an EG member has just been withdrawn.
- No Spec Lead
- The JSRs shared a few common ideas
- Motorola cutting back JCP involvement may have influenced, too
- Allows IAEK to focus more on active 321!
- While no immediate plans exist to integrate ideas from 104, people with interest there are welcome to JSR-321.

TSS C-Function Name	Description	Reason for Removal or Implementation	Visible in API	JSR 321 Object that will handle the functionality
`Tspi_GetAttribUint32`	Find out the value of an integer attribute of an object.	Access to basic information on TSS	No	-
`Tspi_GetAttribData`	Get a non-integer attribute of an object.	Access to basic information on TSS	No	-
`Tspi_GetPolicyObject`	Find out the current authorization policy associated with the context.	Essential for processing commands	Yes	Hidden. Configured using `Secret` object
`Tspi_Context_Close`	Close a context.	Context Sessions are essential to TPM	Yes	`TPMContext`
`Tspi_Context_Connect`	Connect to a context after it is created.	Context Sessions are essential to TPM	Yes	`TPMContext`
`Tspi_Context_Create`	Create a context.	Context Sessions are essential to TPM	Yes	`TPMContext`
~~`Tspi_Context_FreeMemory`~~	~~Free memory allocated by a Tspi-level function.~~	Java hides Memory Management	-	-
`Tspi_Context_GetDefaultPolicy`	Use the default authorization policy for the creation of an object.	Essential	No	Hidden. Configured using Secret object
`Tspi_Context_CreateObject`	Create an object, such as a key object. After creating the object, the fields in the object need to be set.	TPM object live in Contexts	Yes	`TPMContext`
~~`Tspi_Context_CloseObject`~~	~~Destroy an object.~~	Java manages resources	No	-
`Tspi_Context_GetCapability`	Get the current capabilities of the context.	Configuration of Context	No	`TPMContext`

Mapping TSS TSP API to Java Classes

Package Layout

The namespace assigned to JSR 321

Package Summary	
javax.trustedcomputing	This package and its sub packages provide for integration of Trusted Computing in Java.
javax.trustedcomputing.tpm	This package allows to connect to a Trusted Platform Module (TPM).
javax.trustedcomputing.tpm.keys	This package allows the creation, storage, loading and unloading of hierarchies of TPM keys.
javax.trustedcomputing.tpm.structures	This package contains helper classes for interaction with various other classes from the javax.trustedcomputing.tpm package.
javax.trustedcomputing.tpm.tools	This package allows using various core concepts of Trusted Computing.

javax.trustedcomputing

Exception Summary	
TrustedComputingException	The default Exception used in the `javax.trustedcomputing` package.

- The default Exception used in the javax.trustedcomputing package.
- It covers all unexpected behaviors on all levels of the trusted platform.
- This includes also the errors raised in lower layers of the TCG architecture such as error codes returned from the TPM, and the TSS and its sub-layers.
- Returns human-readable error messages and TCG compatible error codes

javax.trustedcomputing.tpm

Class Summary	
TPMContext	The Context class is the centerpiece of the API. This package allows to connect to a Trusted Platform Module (TPM). The Context class is the centerpiece of the API. It serves as central object factory. All TPM-depending objects are created here. While there may exist several TPMContexts at the same time, all derived Objects (such as keys) are only valid within one Context session instance.

Interface Summary	
TPM	This represents the hardware TPM and the basic functionalities it offers. It allows to query the status and capabilities of the hardware TPM and provides access to the random number generator. It also provides access to the Platform Configuration Registers (PCRs).

javax.trustedcomputing.tpm.keys

Class Summary	
KeyManager	Provides management functionality for TPM-based cryptographic keys.

Interface Summary	
BindingKey	Binding keys protect data which is bound to a specific platform.
IdentityKey	IdentityKeys perform signatures on data that originates within the TPM.
LegacyKey	LegacyKeys are the only TPM-based keys that are allowed to perform both signing and encryption operations.
SigningKey	Signing keys sign arbitrary data.
StorageKey	Storage keys wrap other keys or sealed data.
StorageRootKey	The Storage Root Key (SRK) is the highest key in the TPM key hierarchy.
TPMKey	Provides common functionality for all types of TPM -based keys, as created by the KeyManager.
TPMRSAKey	Provides access to the public parts of the RSA keys used by version 1.2 TPMs.

javax.trustedcomputing.tpm.structures

Class Summary	
Digest	Provides a container for SHA-1 hash values.
PCREvent	Holds the data to be extended into PCRs, together with event information that will be stored in the systems Stored Measurement Log (SML).
PCRInfo	The contents of the Platform Configuration Registers (PCR) of a TPM can be used to report the configuration of a system.
Secret	Provides conversion of password strings into the hashed binary format expected by the TPM.
ValidationData	Holds all information necessary to validate that an operation that returns it was properly performed by an authentic TPM.

This package contains helper classes for interaction with various other classes from the javax.trustedcomputing.tpm package. The classes in this package are passive, i.e. do not communicate with the hardware TPM directly.

javax.trustedcomputing.tpm.tools

Class Summary	
Binder	Provides all services for performing the TPM-bind operation on user data.
Sealer	Provides all services for performing TPM_SEAL on user data.
Signer	This class allows to sign user data or files using a SigningKey or a LegacyKey.
~~TickStamper~~	~~Allows to read the current tick counter of the TPM and to perform time stamping operations based on it.~~

This package allows using various core concepts of Trusted Computing.
- Small set of required tools.
- Designed to be extendible with additional features.

Example: How to seal a secret

Imports the API declarations

The implementation of the API is selected by specifying a specific implementation of TPMContext.

We open a connection to the TPM services of the local platform.

KeyManager and all key derived from it are bound to a TPMContext

```
package demo.jsr321;

import javax.trustedcomputing.tpm.TPMContext;
import javax.trustedcomputing.tpm.keys.KeyManager;
import javax.trustedcomputing.tpm.keys.StorageKey;
import javax.trustedcomputing.tpm.keys.StorageRootKey;
import javax.trustedcomputing.tpm.structures.Digest;
import javax.trustedcomputing.tpm.structures.PCRInfo;
import javax.trustedcomputing.tpm.structures.Secret;
import javax.trustedcomputing.tpm.tools.Sealer;

import junit.framework.TestCase;

/**
 * Performs tests on the Sealing functionality
 * @author rtoegl
 *
 */
public class SealingTests extends TestCase {

  /**
   * Performs a straightforward functional test on the Sealer class.
   */
  public void testSealer() throws Exception {

    /**
     * First choose an implementation of TPMContext.
     */
    TPMContext context = TPMContext
        .getInstance("cordelia.tpm.CordeliaTPMContext.java");

    context.connect();

    KeyManager keyManager = context.getKeyManager();
```

..example continued.

TPM Authentication secrets are easily constructed with default encoding.

Now we can instruct the TPM to create a cryptographic key with a specific policy.

We define a platform configuration as a set of PCR values.

Get a Sealer tool instance.

Use it to seal our little secret to this platform-bound key and configuration.

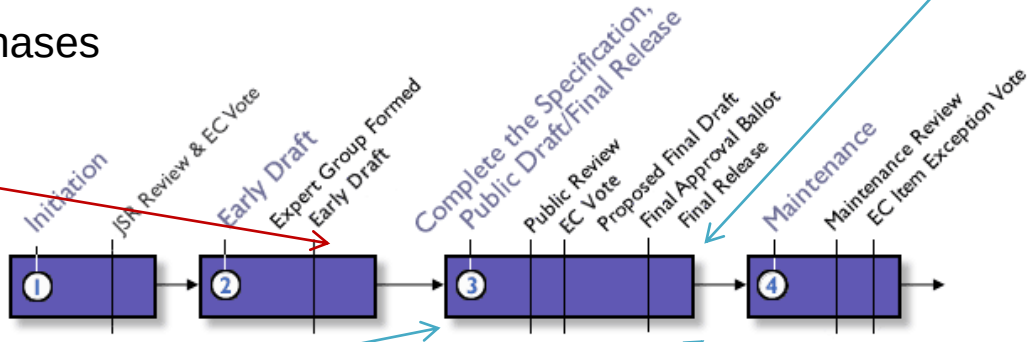
Finally, close the context.

```
Secret keyUsageSecret = context
    .getSecret("TheSecretPassword4SealingAndUnsealing");

StorageRootKey srk = keyManager
    .loadStorageRootKey(Secret.WELL_KNOWN_SECRET);

/**
 * We create a Key of type Storage key, which is non-migratable,
 * non-volatile and need authorization. Using the key is not bound to a
 * specific platform state. sealingKEy is wrapped by the Storage Root Key.
 */
StorageKey sealingKey = keyManager.createStorageKey(srk,
    keyUsageSecret, null, false, false, true, null);

byte[] dataToProtect = "My little secret".getBytes();

PCRInfo targetState = context.getPCRInfo();

Digest pcr12 = context.getDigest(new byte[] { 0xc, 0xa, 0xf, 0xe, 0x0,
    0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
    0x0, 0x0, 0x1 });
targetState.setPCRValue(12, pcr12);

Sealer sealer = context.getSealer();

byte[] sealedData = sealer.seal(dataToProtect, targetState, sealingKey);

// Done sealing data to the platform and its state!


context.close();
```

Java Community Process

- The JCP is an process established to develop and publish Java industry standards.
- It requires a formal sequence of phases
- **We are here**

2010 ?



- An Expert Group collaborates in defining the standard.

- Written Spec

- Updated Written Spec
- Reference Implementation (RI)
- Technology Compatibility Kit (TCK)

RI and TCK MUST cover complete API specification!

How to participate

Get the Draft.

➔ http://jsr321.dev.java.net

Read the specifications and comment on it!

➔ jsr-321-comments@jcp.org

The Expert Group will then discuss your contributions.

Of course, you are also (more than) welcome to join as an Expert or, to provide implementations!

How to contact the Expert Group

Spec Lead:

Ronald Toegl

Institute for Applied Information Processing and Communications (IAIK)
Graz University of Technology, Austria
ronald.toegl@iaik.tugraz.at

3,2,1 GO!