May 28, 2003

A Plan for Testing Conformance and Interoperability of Implementations of the GSC-IS

Eric Dalci
Elizabeth Fong
Alan Goldfine

National Institute of Standards and Technology
Information Technology Laboratory
Software Diagnostic and Conformance Testing Division

1 Introduction

In 1999, the National Institute of Standards and Technology (NIST) agreed to lead development of specifications and standards related to the Government Smart Card program. This effort has resulted in the publication of the Government Smart Card Interoperability Specification (GSC-IS) [NISTIR 6887]. NIST also agreed to develop a comprehensive interoperability/conformance test program for Government Smart Card implementations.

1.1 Purpose of This Paper

The purpose of this paper is to:

- present the requirements for the interoperability/conformance testing of implementations of the GSC-IS
- demonstrate how these requirements are fulfilled by NIST's conformance test program.

1.2 Interoperability vs. Conformance

We say that an implementation of a specification conforms to that specification when it fulfills the requirements of the specification. A conformance clause is a section of a specification that states all the requirements or criteria that must be satisfied for an implementation to claim conformance to the specification. Conformance testing is a way

of verifying this condition, i.e., of determining whether or not the implementation exhibits deviations from the specification.

When two implementations of a given specification behave the same to client applications, then the two implementations can interoperate. For the purposes of this document, we consider two GSC-IS implementations to be interoperable if they have each passed the NIST conformance test system.

2 Architecture of the GSC-IS

The Government Smart Card Interoperability Specification describes an architectural model that includes the following components:

- The Basic Services Interface (BSI), a core set of smart card services at the Application Programming Interface (API) layer between client applications and the smart card service provider modules.

- The Extended Services Interfaces (XSI), which define additional API functions specific to particular implementations. These additional functions are non-interoperable, and are not specified in the GSC-IS.

- The Card Edge Interface (CEI), which defines standard, default sets of interoperable Application Protocol Data Units (APDUs). The ADPUs are a subset of those specified in the ISO 7816 standards. These default sets of ADPUs support both virtual machine and file system smart cards.

- The GSC Data Models, which define the set of containers and data elements within each container on cards supporting that data model. The GSC-IS defines two data models: the GSC data model and the Department of Defense Common Access Card (CAC) data model. The GSC data model was developed for version 1.0 of the GSC-IS.

- The Card Capabilities Container (CCC), one of the required containers present on GSC-IS smart cards. The purpose of the CCC is to describe the differences between the given card's native APDU set and the default set defined by the GSC-IS CEI.

3 GSC-IS Conformance Clause

The conformance clause specified in the GSC-IS specifies conformance conditions for smart card service provider modules (SPMs) and smart cards themselves:

- An SPM implementation that claims conformance to the GSC-IS must implement each of the following:

- The Architectural Model, as defined in Chapter 2
- The Access Control Model, as defined in Chapter 3
- The Basic Services Interface, as defined in Chapter 4
- The Virtual Card Edge Interface, as defined in Chapter 5
- The Card Capabilities Container, as defined in Chapter 6
- Container Naming, as defined in Chapter 7
- Support for both of the Container Data Models defined in Chapter 8 and the appropriate Appendices
- At least one language binding for BSI Services, as defined in the Appendices.

• A smart card that claims conformance to the GSC-IS must support each of the following:

- The Architectural Model as it relates to smart cards, i.e., as defined in sections 1, 4, 5, and 6 of Chapter 2
- The Access Control Model, as defined in Chapter 3
- Either the file system card edge interface or the VM card edge interface, as defined in Chapter 5
- The Card Capabilities Container, as defined in Chapter 6
- Container Naming, as defined in Chapter 7
- One of the Container Data Models defined in Chapter 8 and the appropriate Appendix.

4 Conformance Requirements for GSC-IS

The conformance requirements for implementations of the GSC-IS are defined at two levels: the service call level and the card command (APDU) level.

The service call level is concerned with functional calls required to obtain various services from the card. The GSC-IS addresses interoperability at this level by defining the Basic Service Interface API.

The card command level is concerned with the exact APDUs that are sent to the card to obtain the required service. The GSC-IS addresses interoperability at this level by defining the Card Edge Interface API. The CEI consists of a card-independent, standard set of APDUs implemented by the Service Provider Module,

The conformance requirements for smart cards themselves include the mandatory Card Capability Container. The CCC is a file which contains rules and procedures for translating between the smart card's native APDU set and the standard APDU set defined by the CEI.

4.1 Requirements for the BSI Interface

The Basic Services Interface is a high level API specifying standard smart card services that can be used by client applications. The BSI defines 21 functions grouped into three functional modules: the utility provider module, the generic container provider module, and the cryptographic provider module.

An SPM must provide a BSI as defined in Chapter 4 of the GSC-IS.

Conformance testing requirements for the BSI are:

- All GSC-IS conformant Service Provider Modules must accept each of the defined BSI function calls.

- BSI functions exposed by the SPM must be callable from external applications.

- For every conformance test scenario, the SPM will behave correctly. That is, the SPM must give the expected results regarding access to and data on the smart card, and return the proper return code, as specified in the GSC-IS.

4.2 Requirements at the Card Edge Interface Level

The Card Edge Interface (CEI) is a lower-level, APDU interface between an SPM and a smart card. The GSC-IS specifies 15 default APDU commands, each of which, for the most part, conforms to the ISO 7816-4 protocol for the formatting of APDU commands.

The default APDU set can be implemented directly by both file system and virtual machine cards. Any differences between a vendor’s native APDU set and the GSC-IS default APDU set are accommodated by the rules and procedures documented in the mandatory Card Capability Container.

An SPM must provide a CEI as defined in Chapter 5 of the GSC-IS.

The GSC-IS requires that all Government Smart Cards must be able to exchange data with all conformant SPMs that include a Card Acceptance Device (CAD), e.g., a card reader. The GSC-IS does not specify a particular reader driver.

4.3 Requirements for Conformant Government Smart Cards

The requirements for a GSC-IS conformant card are:

- The card must have a Card Capability Container containing data and structure as defined in Chapter 6 of the GSC-IS.
- The card must support the CEI as defined in Chapter 5.

5 Satisfying the Requirements for Testing

Based upon the above requirements for interoperability, a GSC-IS implementation must be tested for conformance at both the BSI and CEI levels.

5.1 BSI Conformance Testing

The GSC-IS specification of the BSI includes two language bindings, C and Java. A conformant SPM only needs to support one of these.

Conformance test systems have been developed for both the C and Java BSI bindings. Each system consists of :

- test assertions for the BSI commands
- test scenarios that instantiate the assertions

.

- test cases—the actual code that implements the scenarios.

5.1.1 Conformance Test Assertions

There are 21 sets of BSI test assertions, corresponding to the 21 commands defined in the GSC-IS.

Each assertion is a statement of the behavior expected of the candidate implementation as it processes the respective command under different conditions. Each assertion is written in narrative form, and consists of:

- a starting state,
- a function (C) or method (Java) call with a set of generic legal or illegal input parameters, and

- the return code, generic return values, and return conditions that are anticipated of the candidate implementation, based on the input parameters.

There are generally many test assertions corresponding to one BSI function (method).

5.1.2 Conformance Test Scenarios

The BSI test scenarios attempt to bridge the gap between the high-level test assertions and the actual test code. Each scenario includes:

- a detailed starting state, including specific values for input parameters,

- a detailed scenario specifying how return conditions are verified. For example, a DataCreate BSI command is followed by a ReadValue to verify that the DataCreate command did indeed store the correct data in the target container.

The test scenarios are written in narrative form, and are then manually translated to XML.

There is at least one test scenario corresponding to each test assertion.

5.1.3 Conformance Test Code

The actual test code (in C for the C binding, or Java for the Java binding) is generated from the XML translation of the scenarios by applying a Style Sheet Language XSLT. This approach for generating source code is described in "Conformance Testing Architecture for the Basic Service Interface of the Government Smart Card Specification."

The C or Java test code is then compiled. A data constants file, which includes initial values for parameters as well as globally defined parameter values, is created.

At this point, the compiled executable code is ready to be executed against a candidate implementation.

5.2 CEI Conformance Testing

CEI testing includes testing for each operation in the default GSC APDU set. The default GSC APDU set is applicable to both file system and virtual machine cards. To satisfy the requirement that a conformant card must contain the mandatory CCC, the card edge interface test will first find the CCC and check that the mandatory field structure of the CCC is conformant. The registered data model number is then read, to ensure that one of

the two defined data model (GSC or CAC) is specified. If the above tests are passed, then the tests corresponding to the individual APDUs are executed.

As with BSI testing, APDU testing includes tests with either legal or illegal parameters. The response from the candidate card is compared with the expected response to produce a pass/fail condition.

Each file system card APDU test implicitly tests the correctness of the mapping information, contained in the CCC, that translates the card's native APDU into GSC default APDU.

6 Conformance Test Suite Reports

The test reports for the three test suites (C binding, Java binding, and CEI) are each constructed using HTML format. Each report contains a header information section specifying the software and hardware platform upon which the suite was executed, identification of the card reader and tested card, the date, and the name of the test operator. The body of the report consists of an account of the execution of each test case. For each test case,

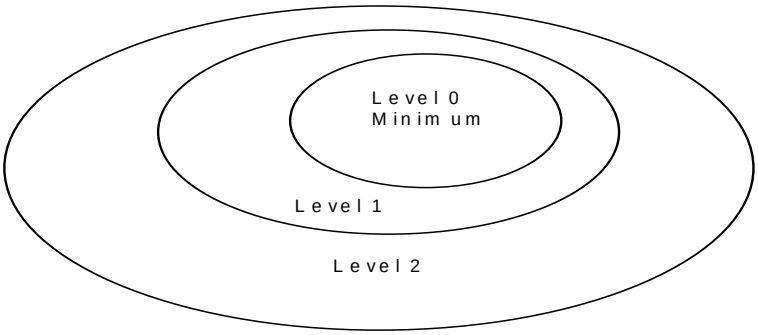
- starting conditions and input parameters
- expected return codes and values, and
- the actual codes and values returned by the candidate implementation

are noted. If the actual results are the same as the expected ones, the implementation is considered to have passed the test case, and a "test passed" notice is highlighted in green. Otherwise, the implementation is considered to have failed, and a "test failed" notice is highlighted in red.

7 Levels and Profiles of Conformance

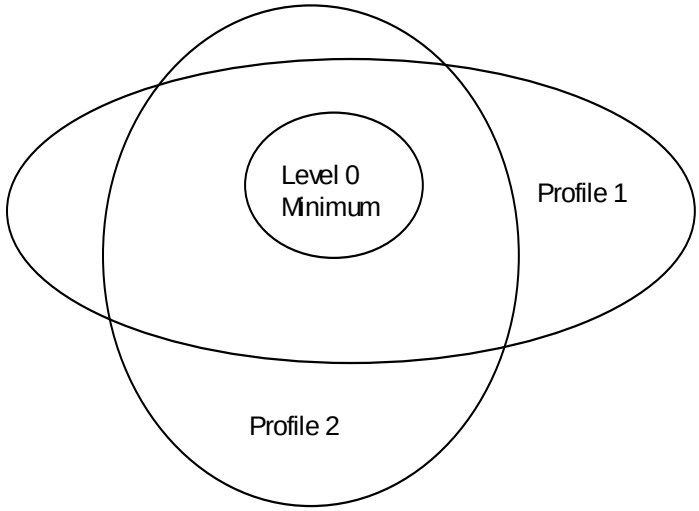
There is a requirement that the conformance criteria partition the functionality specified in the GSC-IS into levels and/or profiles.

Levels of conformance reflect a partitioning into increasing supersets of functionality, beginning with a minimal set that is applicable to all implementations, and ending with the set consisting of the entire functionality specified in the GSC-IS.



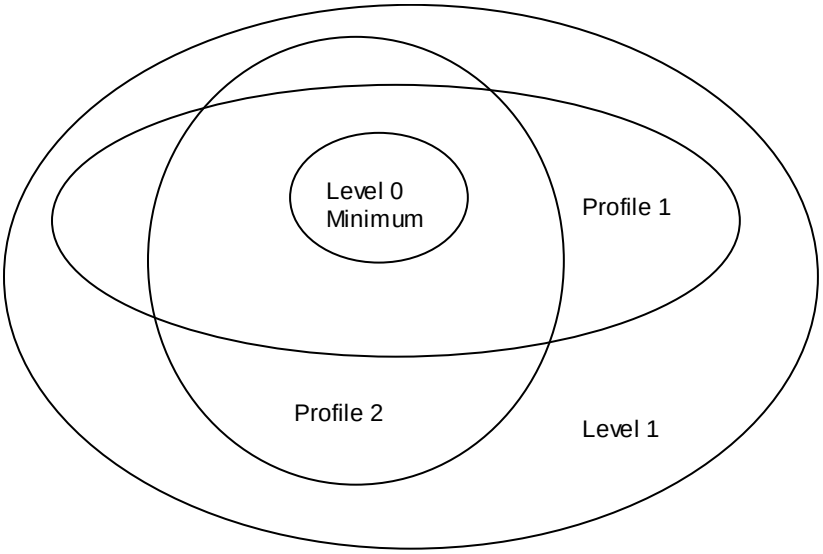
Example of a Levels structure

Profiles constitute a less rigidly constructed collection of sets of functionality. Each profile must contain the minimal set, but otherwise can partially intersect, or be disjoint from, other profiles. Profiles may be designed for particular applications or user communities.



Example of Profile Structure

It is possible that a profile structure can be combined with a level structure:



Example of a combined Levels and Profiles structure

A separate paper on GSC-IS conformance levels and profiles will be developed.