Design and Implementation of an ICC Profile Validator

Tim Kohler
Canon Information Systems, Cupertino, California

Abstract

The International Color Consortium (ICC) has developed a useful tool for communicating color characteristics between different computers and devices. This tool is the ICC profile, a cross-platform data file format designed for flexibility and portability of color transformation data. The ICC Profile Format Specification[1] is the reference for this format. However, the ICC format is complex and there is a lot of room for errors and misunderstanding when creating an ICC profile. Since many companies and institutions are creating Color Matching Modules (CMMs), and even more are creating profiles, it is important that each of these be interoperable. Misunderstandings of the specification could lead to problems with compatibility between CMMs and profiles created by different vendors.

To address this issue, the ICC has formed a working group to design some software to verify compliance with the specification. One result of this work is the ICC Profile Validator. This paper describes the motivation, design considerations, and limitations of an ICC profile validator.

Motivation

Shortly after the ICC released the first ICC Profile Specification, version 3.0, many of the eager users of the profile format were confused about numerous issues. Several users created profiles with uncertainty of their compatibility with other systems. Also, since the vendors creating CMMs had the same questions, users had no way to test the compatibility of the profiles they had created without actually trying them on every system.

Many people contacted the ICC or individual member companies, asking questions about specifics or asking if there was some way to test their profiles for compliance. This volume of inquiries led to the formation of a conformance testing working group within the ICC. However, there were many questions to be resolved about what the compliance test should entail. The working group needed to decide what level of testing should be done. Should quality be tested? Should a profile that had the correct data structures but produced only black output be deemed a valid ICC profile? Furthermore, how was the code to be written and who was to write it? Finally, since the ICC had prided itself on the cross-platform cooperation in designing the profile format, how was the validator code to be created and maintained for at least four different operating systems?

Early this year, the author began to develop a solution and, in conjunction with other ICC members, a profile validator became a reality. A recent breakthrough in software technology and it's rapid acceptance by the computer industry made it possible to solve the cross-platform issue. The Java programming language,[2] created by Sun Microsystems, allowed easy development of cross-platform software such as this project. The use of Java made it possible for a profile validator to become a reality, since only one set of source code needs to be maintained.

Design

Three aspects of the design of the profile validator will be discussed; the functional design, the software design, and the user interface design. The functionality was based on work by ICC members who helped design tests for each element of the profile format. The software design was guided by the object oriented design requirements of the Java language, and the user interface was designed to be simple yet useful for many types of users.

Functional Design

Many testing modalities were considered for validation, including testing profile syntax, profile content, and CMM functionality. For this project, it was decided that only profiles would be tested and that no testing of their content or quality would be done. The reason for this decision was there were many differing opinions on what high quality meant. Since the ICC profile has a very broad usage, from inexpensive desktop printers to offset presses, the quality expectations were very different. Therefore, quality was left up to the profile creator.

The validator returns one of three results, pass, fail, or warning. The user has control over how much information is given for a profile that passes. The user may view all of the profile attributes as they are tested or they may view only the final result. A "pass" result indicates that no problems were found in the profile. A "warning" indicates that the profile's color data and functionality are unimpaired, however, some data fields were found to have errors. An example of a warning is a creation date before 1992. A "fail" result is due to any other error that would impact the normal use of the profile. A description of each warning or failure is always given.

There are five categories of tests performed by the validator: global structure tests, header data tests, required tag tests, "per-tag" tests, and data type tests. The global structure tests check file-related attributes and overall data structure. For example, the actual file size must match the file size stated in the header field, the file must be at least a minimum size, and the tag table must not be corrupted. Since global structure errors often indicate a corrupted data

structure, a failure of a global structure test ends the validation since the remaining tests rely on structure information to find data. The header data tests are fairly straight-forward. The tests validate the data in the header fields, for example, the "magic number", ('ascp') must appear in bytes 36-39. Some of these tests, such as the device manufacturer and device model signature fields, are not currently tested. In the future, the ICC will provide a computer readable list of registered manufacturers and model names that will be used to check these fields.

The required tag tests are based on the profile and device class. This test checks for the existence of certain tags in the tag table. All profile classes are required to contain a copyright tag ('cprt') and a profile description tag ('desc'). Each profile class and device class have specific public tags which must be present. If any required tags are not present, the profile fails the validation.

The per-tag tests make up the bulk of the validator tests. These tests review each public tag, reading the tag type and performing tag specific tests on the data. Some tag tests are very simple and others are more involved. All public tags are tested for a valid tag signature, the tag type and the tag signature must match, and the reserved bytes must be empty. Typical tests are for tag size, verifying valid ASCII strings, and verifying that numerical values fall into the appropriate range.

Finally, there are data type tests. These are used by the per-tag tests. These tests check for valid data ranges in often used data types. The three data types tested are dateTimeNumber, XYZNumber, and ASCII strings. These tests check for valid number types, valid numerical ranges of values, and valid ASCII characters.

Software Design

The validator was written in the Java programming language in order to allow for one set of source code to produce executable software on many computing platforms. One of the features of Java is the strict object oriented paradigm used for the language. Everything in Java is derived from a class. There are four main Java classes used in the validator: Valid, ValidUIFrame, Profile, and Tag. There are also several miscellaneous classes for dialog boxes, and other user interface elements. The Valid class is the main thread, it launches the user interface and handles cleaning up when the validator is done. ValidUIFrame creates the user interface frame and handles events within it. The Profile class represents one profile, it's methods test the global structure, tag table, and required tags. Each Tag class represents one tag. The methods within the Tag class perform individual per-tag tests. Tag objects are instanciated by Profiles. Profile objects are instanciated by the user interface when a profile is opened for testing.

User Interface

The user interface was designed to be simple and to give a quick method of checking a profile. The user is presented with a control panel containing a field for selecting a profile. Once selected, the user clicks the "Go" button and the profile is tested. A panel lights up with the result, a green panel for pass, an orange panel for warning, and a red panel for fail. There is also a text area that displays specific warnings or error messages that allows the user to diagnose problems. The user can select an option that causes the validator to display detailed information about the profile in the text area as well as warnings and errors. The text report may be saved to a file if desired.

Limitations

The profile validator is designed to test an ICC profile as thoroughly as possible. However, there are some limitations with the present implementation which are due to either practical restraints or work yet to be done by the ICC. The first limitation is that the validator only checks the syntax of profiles. There are no tests of content or quality. There are currently no tests for CMMs. Signatures registered by the ICC, such as device manufacturer and device model, are not tested. Private tags are not tested. The ProfileSequenceDescType is not tested because of the complex nested structure. The purpose of this validator is only to aid the builder of a profile. The validator will help point out and debug problems, however, it is the final responsibility of the profile builder to check the profile before use or distribution.

Future Directions

The Profile Validator will be presented to ICC for review and acceptance as an ICC approved validation tool. This approval must be based on extensive testing by the ICC. Testing entails a rigorous process whereby a test profile containing a specific error is generated for each error tested for by the validator. These profiles must give the appropriate error for each test case. Future versions will include the ability to test a whole directory of profiles, testing of registered signatures such as device manufacturer, and testing of the ProfileSequenceDescType. Also, each time a new version of the ICC Profile Format Specification is released, a new version of the validator will be released.

Acknowledgments

The author would like to acknowledge the invaluable assistance of Michael Vigneau of Polaroid Corporation for his help in designing the functionality. Also appreciated is the work done by the other members of the ICC conformance testing working group: Dirk DeBaer, Agfa-Gevaert, N.V. and Todd Newman, Silicon Graphics. And of course, the ICC, whose members contributed to this project.

References

1. *ICC Profile Format Specification*, Version 3.2, International Color Consortium, November 20, 1995.
2. J. Gosling, B. Joy, and G. Steele, *The Java Application Programming Interface*, Vol. **1**, Core Packages, Addison Wesley, 1996.