International Color Consortium®

White Paper 21

Level: Intermediate
Date: Feb 2006

Profile Compliance Testing – SampleICC Implementation Notes

1. INTRODUCTION

The ICC profile format specification defines an open file format that acts as an exchangeable container for data that is used to perform color transformations. The file format defines data elements, order and meaning. Implementation of systems that provide reading and/or writing of ICC profile files can be made for a variety of reasons. Determining correctness of a profile's generation/modification as outlined by the profile specification sometimes goes outside the information provided directly by the profile specification. This is because the exact tests to perform, the order they should be performed, and what the results of those tests mean is not clearly defined by the profile specification.

Providing a testing specification is a bit tenuous since to some degree the tests and meaning of the results can be context specific because not all implementations will use the data in the profiles for the same reasons. Additionally any changes to the profile specification would require changes to the testing specification. Keeping them in sync and determining ultimate authority becomes a problem.

Rather than provide a complete profile testing specification, this document will endeavor to describe some of the issues of profile conformance testing that were identified and addressed as part of implementing profile conformance capabilities into the SampleICC project.

The SampleICC project (see http://sampleicc.sourceforge.net) is an open source object oriented C++ development effort that was written to provide an example of how various aspects of color management can be implemented. It is maintained by the Architecture Working Group of the ICC.

This document provides overviews of test types and specifics of some important tests that are implemented in SampleICC's IccProfLib. It is hoped that this discussion along with the code in SampleICC can be used as a guide to understanding issues related to determining profile compliance. This document along with the code in SampleICC should NOT be considered as a profile conformance testing specification.

Notice: NOT ALL POSSIBLE TESTS RELATED TO PROFILE COMPLIANCE ARE PRESENTED IN THIS DOCUMENT. Such tests can be determined through a careful study of the ICC profile specification. Even though the tests in SampleICC are extensive, this document and the SampleICC code do not pretend to replace the ICC Profile Specification. The ICC Profile Specification should always be considered the ultimate source in determining profile compliance.

2. Profile Compliance Levels

In general there are three questions that can be asked in relation to profile compliance:

1. Is the profile legible? (Can the profile be read?)
2. Does the profile conform to the specification?
3. Is the profile useable?

The first question relates to whether the ICC profile can be parsed.

The second question assumes the first to be true. Once one can parse the file then there is a question whether the data within the file makes logical sense (as defined by the profile specification).

The third question is actually the question that usually begs to be asked. This question often goes beyond simple profile conformance testing. However, in some cases the answer to question number two may be negative, and the answer to question number three affirmative. A profile might not comply with the specification in some area outside the scope where the profile will be used, thus the profile is usable.

The issue of profile usability can partially be addressed by introducing levels of compliance. This document considers four levels of compliance:

1. Compliant – The profile is completely compliant.
2. Warning – Possible problems exists, but the profile is compliant with the specification.
3. Non-Compliant – The profile does not strictly follow the ICC Profile Format Specification.
4. Critical-Error – The non-compliance of the profile makes the profile unusable.

Notice there is a distinction between levels three and four. Both levels indicate that the profile does not strictly follow the ICC Profile Format Specification. For example there are Non-Compliant issues that may not effect whether a CMM can successfully understand and apply the profile under specific conditions. The distinction between levels three and four in many cases can be argued one way

or another. For example: A Non-Compliant issue for a CMM might be a Critical-Error to a profile manager and vice-versa.

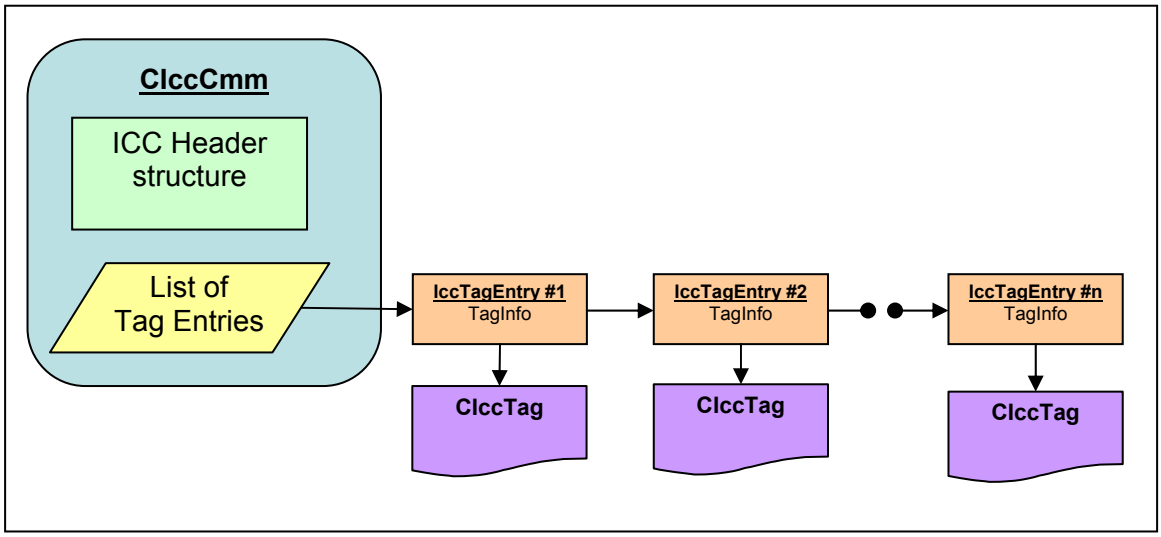
Note: The use of compliance levels in this document will pertain to the successful application of a profile within SampleICC.

The distinction of compliance levels also allows for a discussion of robustness to allow for future extensions of the profile specification. For example: At some future time an informational only field could be added using part of the reserved portion of the profile header. This would make the profile non-compliant in terms of an older specification, but since the field is informational it would not be a Critical-Error from a profile application point of view.

There is a clear distinction between profile compliance and validity. This document only considers issues of compliance. It is possible for a profile to be completely compliant to the ICC Profile Specification and not be valid for any constructive use. For example: A profile that turns all images black could be labeled as compliant if it passes all the types of tests involved in this document.

3. Profile Compliance Testing in SampleICC

The Profile Object Hierarchy of a profile loaded into memory is represented in the following figure:



Object Hierarchy of a Profile in memory

The **CIccProfile** class is responsible for defining operations on profiles. A **CIccProfile** object essentially maintains two data members - the profile header data, and a list of **IccTagEntries**.

Each **IccTagEntry** contains a pointer to a **CIccTag** derived object as well as a **TagInfo** structure that contains the tag signature, size, and location of the tag in a loaded file.

All tag types defined by the ICC specification are represented by **CIccTag** derived classes.

The SampleICC project splits answering the profile compliance questions into two parts. These two parts are implemented through the use of two public member functions of the **CIccProfile** class:

- **CIccProfile::ReadValidate()**

  The **CIccProfile::ReadValidate**() member function directly answers the first of the profile compliance questions. It is an alternative version of the **CIccProfile::Read**() function which parses an ICC profile file and generates the object structure in memory. While parsing the ICC profile it contributes to a profile compliance report and returns the maximum compliance level of all the tests that are performed.

  The following operations are performed in **ReadValidate()**. Each of these operations involves making calls to protected member functions of **CIccProfile**. The tests/operations are:

  1. The header is read and the Tag directory is parsed and read.
  2. The actual file size is compared to the profile size specified in the header.
  3. The profile ID is calculated and compared to that found in the header.
  4. Each Tag defined by the profile Tag directory is parsed and read in using the Tag classes in SampleIcc's IccProfLib.

- **CIccProfie::Validate()**

  The **CIccProfile::Validate**() member function tries to answer the last two profile compliance questions. Issues of usability (expressed through compliance level reporting) are limited to the known ability of the **CIccCmm** class to apply the profile. Whether a profile is useable for any other specific purpose goes beyond the scope of this function.

  The **CIccProfile::Validate**() member function acts separately from the **CIccProfile::ReadValidate**() member function and can be called to determine profile conformance issues of profiles that are being generated in memory using SampleIcc's IccProfLib before they are written to file.

  The **CIccProfile::Validate**() function also contributes to a profile compliance report and returns the maximum compliance level of all the tests that are performed.

The following operations are performed in **CIccProfile::Validate**(). Each of these operations involves making calls to protected member functions of CIccProfile. The tests/operations are:

1. The header is checked for compliance to the Profile specification. (See: **CIccProfile::CheckHeader()** in SampleIcc's IccProfLib).
2. Tags are checked for uniqueness. (See: **CIccProfile::AreTagsUnique()** in SampleIcc's IccProfLib).
3. Required tags are checked for the profile type (based upon profile type information stored in the header). (See: **CIccProfile::CheckRequiredTags()** in SampleIcc's IccProfLib).
4. The tag types are checked. (See: **CIccProfile::CheckTagTypes()** in SampleIcc's IccProfLib).
5. The virtual **CIccTag::Validate**() member function for each of the **CIccTag** derived objects associated with the profile's Tag directory is called. Within IccProfLib, each **CIccTag** class is responsible for providing its own specification compliance testing.

SampleIcc's IccProfLib also provides a single static global function (IE: **ValidateIccProfile**) to simplify profile compliance testing. This function performs the following operations:

1. It initiates profile file I/O.
2. It allocates a new **CIccProfile** object.
3. It calls **CIccProfile::ReadValidate**().
4. It calls **CIccProfile::Validate**().
5. Steps 3 & 4 generate a profile file report and compliance level for the profile.
6. It returns the loaded profile object structure.